

Introduction au Deep Synergy Learning

Mohammed Bouayoun

Chapitre 1 : Introduction au Deep Synergy Learning (DSL)

1.1.	Contexte et Motivation	5
1.1.1.	Bref Historique de l'IA.....	5
1.1.2.	Émergence des Réseaux Profonds (Deep Learning) et Limites.....	6
1.1.3.	Des Approches Conventionnelles à la Synergie Informationnelle	7
1.1.4.	Positionnement du DSL dans le Panorama de l'IA	7
1.1.5.	Rôle Potentiel du DSL vers l'IA Forte (IAG)	8
1.1.6.	Objectifs et Contributions Clés du DSL.....	9
1.1.7.	Plan Général du Chapitre.....	11
1.2.	Définitions et Concepts Préliminaires	17
1.2.1.	Qu'est-ce qu'une "Entité d'Information" ?.....	17
1.2.2.	Notion de "Synergie Informationnelle"	18
1.2.3.	Différence entre Interaction, Synergie et Corrélation	20
1.2.4.	Approche Hiérarchique vs Approche Auto-Organisée	22
1.2.5.	Réseaux Neuronaux Traditionnels vs Réseaux Synergiques.....	25
1.2.6.	Terminologies Employées dans le DSL	27
1.2.7.	Exemples Illustratifs de la Synergie dans la Nature	30
	Problèmes.....	33
1.3.	Importance de l'Auto-Organisation.....	37
1.3.1.	Inspirations Biologiques et Cognitives	37
1.3.2.	Concepts Clés : Émergence, Auto-Régulation, Feedback	39
1.3.3.	Comparaison avec les Méthodes d'Apprentissage Classiques	41
1.3.4.	Le Rôle des Flux d'Information Multimodaux	43
1.3.5.	Évolution Dynamique et Adaptation en Continu	45
1.3.6.	Impacts sur la Robustesse et la Résilience des Modèles	48
1.3.7.	Perspectives pour une Approche plus Globale de l'Apprentissage	50
1.4.	Architecture Générale du DSL.....	61
1.4.1.	Principe de Base : Entités et Liens Synergiques	61
1.4.2.	Présentation du Synergistic Connection Network (SCN)	62
1.4.3.	Notion de Cluster et de Macro-Cluster	64
1.4.4.	Fonctions de Synergie : Distance, Similarité et Co-Information	67
1.4.5.	Pondérations Adaptatives et Évolution Temporelle	69
1.4.6.	Interactions Directes et Indirectes	72
1.4.7.	Synergie binaire et n-aire : au-delà des relations deux à deux.....	74
1.5.	Pourquoi une Approche Synergique ?	78

1.6. Applications Pressenties et Domaines Impactés.....	113
1.6.1. Vision Artificielle et Reconnaissance d'Objets Complexes	113
1.6.2. Analyse Audio et Traitement du Langage Naturel.....	116
1.6.3. Robotique et Systèmes Intelligents Évolutifs	119
1.6.4. Recommandation Personnalisée et Systèmes de Décision.....	123
1.6.5. Surveillance, Diagnostic Médical et Anomalies	126
1.6.6. Planification et Optimisation dans l'Industrie 4.0	129
1.6.7. Perspectives pour la Recherche Fondamentale en IA Forte	133
1.7. Défis, Contraintes et Ouvertures.....	137
1.7.1. Complexité Computationnelle et Scalabilité	137
1.7.2. Qualité et Disponibilité des Données.....	140
1.7.3. Développement d'Algorithmes d'Optimisation Appropriés	143
1.7.4. Contrôle et Stabilité des Processus Auto-Organisés	146
1.7.5. Interprétabilité et Explicabilité pour l'Humain.....	149
1.7.6. Considérations Éthiques et Réglementaires	152
1.7.7. Comparaisons Expérimentales avec d'Autres Approches	155
1.8. Positionnement du DSL dans l'Évolution de l'IA	159
1.8.1. IA Symbolique vs IA Sub-symbolique : Intégration Potentielle.....	159
1.8.2. DSL et Apprentissage Profond : Collaboration ou Substitution ?	162
1.8.3. Approches Hybrides : DSL, RL (Reinforcement Learning) et Logique.....	164
1.8.4. Rôle de la Mémoire et de l'Attention dans le DSL	167
1.8.5. Tendances Futures : Vers une IA Forte, Consciente ?	170
1.8.6. Effet Sur la Recherche Interdisciplinaire.....	173
1.8.7. Exemple de Convergence : DSL & Neurosciences	175
1.9. Méthodologie, Ressources et Outils	179
1.9.1. Approche de Recherche : Théorique, Expérimentale, Hybride	179
1.9.2. Bases de Données et Plates-formes de Test pour le DSL	182
1.9.3. Frameworks de Développement : Python, C++, Librairies spécialisées	186
1.9.4. Environnements de Simulation et d'Évaluation.....	190
1.9.5. Protocoles de Validation : Qualitatifs et Quantitatifs	195
1.9.6. Collaboration et Partage de Ressources (Open Source)	200
1.9.7. Gestion du Cycle de Vie d'un Projet DSL.....	205

1.1. Contexte et Motivation

1.1.1. Bref Historique de l'IA

1.1.2. Émergence des Réseaux Profonds (Deep Learning) et Limites

1.1.3. Des Approches Conventionnelles à la Synergie Informationnelle

1.1.4. Positionnement du DSL dans le Panorama de l'IA

1.1.5. Rôle Potentiel du DSL vers l'IA Forte (IAG)

1.1.6. Objectifs et Contributions Clés du DSL

1.1.7. Plan Général du Chapitre

1.1. Contexte et Motivation

1.1.1. Bref Historique de l'IA

L'intelligence artificielle (IA) constitue un domaine de recherche qui s'inscrit dans une longue tradition, remontant à la première moitié du XXe siècle. Son évolution a été marquée par des jalons fondamentaux qui ont façonné la discipline telle qu'elle se présente aujourd'hui. Pour mieux appréhender l'émergence de l'Apprentissage Profond Synergétique (APS), ou **Deep Synergy Learning (DSL)** en anglais, il est essentiel d'examiner les principaux courants qui ont successivement vu le jour ou cohabité.

Les premières réflexions autour de l'intelligence artificielle trouvent leur origine dans les travaux d'Alan Turing, notamment à travers son article de 1950, *Computing Machinery and Intelligence*. Dans ce texte fondateur, il introduit le célèbre test de Turing, conçu pour évaluer la capacité d'une machine à simuler l'intelligence humaine dans le cadre d'une interaction textuelle.

C'est lors de la conférence de Dartmouth en 1956 que l'IA est officiellement définie comme un champ scientifique. Cette rencontre, organisée par John McCarthy, Marvin Minsky, Claude Shannon et Nathan Rochester, marque l'acte fondateur de la discipline. Elle établit les bases conceptuelles et méthodologiques nécessaires à la création de systèmes intelligents.

Les premières approches de l'IA reposaient sur la manipulation de symboles et de règles logiques pour modéliser la pensée humaine. Ces travaux, regroupés sous l'appellation **IA symbolique**, cherchaient à reproduire des processus cognitifs à travers des systèmes de règles formelles. Des programmes tels que *Logic Theorist* (1956) et *General Problem Solver* (1959) illustrent cette démarche en appliquant des algorithmes de recherche pour résoudre des problèmes mathématiques ou généraux.

Cependant, bien que performante dans des environnements fermés, l'IA symbolique montrait des limites lorsqu'il s'agissait de traiter des situations complexes ou incertaines. Cette incapacité à généraliser au-delà de scénarios spécifiques a révélé la nécessité d'approches complémentaires.

Parallèlement à l'IA symbolique, le courant connexioniste, basé sur les réseaux de neurones artificiels, gagne en popularité. Des chercheurs comme Frank Rosenblatt, Bernard Widrow et Marcian Hoff ont exploré le potentiel des réseaux d'unités élémentaires capables d'apprendre à partir de données, comme en témoigne le Perceptron (1958). Cependant, les travaux de Marvin Minsky et Seymour Papert, publiés en 1969, ont mis en lumière les limites des réseaux de neurones monocouches, incapables de traiter des problèmes non linéaires. Cette critique, combinée à l'absence de méthodes efficaces pour entraîner des réseaux multicouches, a conduit au déclin temporaire de l'intérêt pour le connexionisme, marquant le premier "hiver de l'IA".

Le milieu des années 1980 voit la renaissance des réseaux de neurones grâce à la découverte de la rétropropagation, une méthode permettant d'entraîner efficacement des réseaux multicouches. Ce renouveau est accompagné d'une montée en puissance des approches statistiques de l'apprentissage automatique, telles que les modèles graphiques probabilistes et les machines à vecteurs de support (SVM). Ces outils élargissent les applications de l'IA à des domaines variés, notamment la vision par ordinateur et le traitement du langage naturel.

La fin des années 2000 marque l'avènement du Deep Learning, rendu possible par la convergence de trois facteurs majeurs. L'abondance des données massives issues d'Internet et des technologies numériques a offert un terrain fertile pour l'entraînement des modèles. L'émergence de ressources de calcul performantes, telles que les GPU et les TPU, a permis de traiter des volumes de données colossaux. Enfin, des progrès méthodologiques, incluant de nouvelles fonctions d'activation et des stratégies d'optimisation, ont renforcé l'efficacité des réseaux de neurones profonds. Ces avancées ont conduit à des percées significatives dans des domaines comme la reconnaissance d'images, la traduction automatique et les jeux stratégiques, illustrées notamment par le succès d'AlphaGo.

Malgré ces réussites, le Deep Learning présente certaines limites, notamment une forte dépendance aux données annotées, un manque d'interprétabilité et des difficultés à généraliser. Ces contraintes mettent en évidence la nécessité d'approches complémentaires.

Face aux limitations des méthodes actuelles, de nouveaux paradigmes ont émergé, mettant l'accent sur l'auto-organisation et les interactions dynamiques entre sources d'information. Le **Deep Synergy Learning (DSL)** s'inscrit dans cette dynamique en intégrant des notions de coopération et de coévolution au sein des systèmes intelligents. Cette approche vise à dépasser les modèles existants en favorisant une intelligence plus adaptable et autonome, ouvrant ainsi la voie à l'intelligence artificielle générale (IAG).

1.1.2. Émergence des Réseaux Profonds (Deep Learning) et Limites

L'émergence des réseaux profonds, ou **Deep Learning**, constitue un tournant fondamental dans l'histoire de l'intelligence artificielle. Bien que les réseaux de neurones artificiels aient été introduits dès les années 1960, ce n'est qu'au début des années 2010 qu'ils se sont imposés comme une technologie centrale grâce à plusieurs avancées majeures. Ces progrès incluent l'accroissement massif des données numériques, le développement de ressources matérielles puissantes comme les GPU, et l'introduction de méthodologies innovantes. Cependant, malgré ces succès, le Deep Learning continue de présenter des limitations importantes qui suscitent l'exploration de nouvelles approches.

L'avènement du Deep Learning repose sur plusieurs éléments déterminants. D'une part, l'ère numérique a produit une quantité sans précédent de données issues de diverses sources, telles que des images, des vidéos, des textes ou des signaux. Ces ensembles de données volumineux, comme le dataset ImageNet, ont permis d'entraîner des réseaux plus profonds et plus complexes. D'autre part, l'utilisation des GPU, conçus initialement pour le traitement graphique, a permis d'accélérer les calculs massivement parallèles nécessaires à l'entraînement des modèles. Ces avancées matérielles ont été accompagnées de progrès méthodologiques, comme l'introduction de fonctions d'activation avancées, la régularisation par dropout ou encore la normalisation de lots (batch normalization). Enfin, les succès visibles dans des applications telles que la reconnaissance d'images, la traduction neuronale et les jeux stratégiques ont renforcé l'intérêt pour cette technologie.

Les réseaux de neurones profonds ont apporté des avantages significatifs dans le traitement des données complexes. L'une de leurs principales forces réside dans leur capacité à extraire automatiquement des caractéristiques pertinentes à partir des données brutes, ce qui simplifie considérablement les tâches d'ingénierie des features. De plus, leur performance dans des domaines tels que la classification d'images ou la reconnaissance vocale dépasse de loin celle des méthodes traditionnelles. Lorsqu'ils sont entraînés avec suffisamment de données pertinentes, ces réseaux montrent une certaine aptitude à généraliser en identifiant des motifs complexes que d'autres approches ne peuvent pas détecter.

Malgré leurs performances impressionnantes, les réseaux profonds présentent plusieurs limites. Tout d'abord, leur dépendance aux données annotées est problématique, car l'annotation de grands volumes de données est coûteuse et parfois impraticable dans certains contextes. Ensuite, leur nature de boîte noire rend difficile l'interprétation des décisions prises, ce qui pose des problèmes d'explicabilité, notamment dans des secteurs critiques comme la santé ou la justice. Les réseaux profonds sont également vulnérables aux attaques adversariales, où des perturbations mineures peuvent induire des erreurs majeures. De plus, leur capacité de généralisation reste limitée lorsqu'ils sont confrontés à des contextes différents de ceux rencontrés durant l'apprentissage. Enfin, le coût énergétique élevé de l'entraînement de ces modèles soulève des préoccupations en termes de durabilité et d'accessibilité.

Face à ces limitations, de nouvelles approches visent à repousser les frontières du Deep Learning. L'apprentissage peu supervisé ou auto-supervisé, qui réduit la nécessité de données annotées, est en plein essor. Par ailleurs, l'intégration du raisonnement symbolique avec les réseaux de neurones, dans une approche neuro-symbolique, pourrait permettre une intelligence plus transparente et explicable. Enfin, des recherches explorent des architectures auto-organisées où les informations interagissent de manière dynamique, rompant avec la rigidité des modèles actuels. Ces évolutions pourraient ouvrir la voie à des systèmes plus robustes, adaptatifs et proches d'une véritable intelligence artificielle générale.

1.1.3. Des Approches Conventionnelles à la Synergie Informationnelle

L'histoire de l'intelligence artificielle témoigne de la succession de grandes tendances méthodologiques, chacune cherchant à reproduire ou modéliser certains aspects de l'intelligence humaine ou animale. Parmi ces approches, les approches symboliques et les approches statistiques ou connexionnistes ont dominé. Les premières s'appuient sur la formalisation explicite de la connaissance à travers des règles et des faits, tandis que les secondes reposent sur l'apprentissage à partir de grandes quantités de données via des modèles tels que les réseaux de neurones artificiels. Bien que ces paradigmes aient permis des avancées considérables, ils montrent leurs limites lorsqu'il s'agit de traiter des systèmes complexes, dynamiques ou émergents. La synergie informationnelle s'inscrit dans une tentative de dépasser ces restrictions en proposant une approche inspirée des systèmes naturels.

Les systèmes naturels, tels que les écosystèmes, les colonies d'insectes ou le cerveau humain, illustrent un principe fondamental selon lequel l'ensemble dépasse la somme de ses parties. Ces systèmes se caractérisent par l'auto-organisation, où des structures émergent sans planification globale, la robustesse grâce à la redondance des composants, et la capacité d'adaptation en réponse à des variations environnementales. Les entités locales interagissent dynamiquement à travers des signaux ou des mécanismes simples, conduisant à des comportements collectifs complexes. Transposé à l'intelligence artificielle, ce principe propose de considérer les informations comme des entités actives capables d'interagir, de coopérer et de s'adapter en permanence.

Les approches classiques du Deep Learning, bien qu'efficaces, restent enfermées dans une structure hiérarchique rigide où les données circulent de manière linéaire ou avec des boucles limitées. Cela restreint leur capacité à modéliser des interactions complexes ou des phénomènes émergents. En particulier, dans des scénarios multi-modaux combinant des flux hétérogènes comme la vision, le texte ou l'audio, l'intégration repose souvent sur des mécanismes de fusion statiques qui ne permettent pas une réelle interaction entre les modalités. Ces architectures peinent également à s'adapter à des environnements dynamiques ou partiellement inconnus.

La synergie informationnelle propose une vision radicalement différente. Les entités d'information, au lieu d'être des entrées passives dans un modèle prédéfini, deviennent des acteurs autonomes capables d'interagir et d'évoluer. Elles peuvent évaluer leur synergie avec d'autres entités, c'est-à-dire mesurer leur capacité à produire ensemble des informations plus riches que la somme de leurs contributions individuelles. En fonction de cette évaluation, elles peuvent nouer, rompre ou modifier des connexions, favorisant l'émergence de structures auto-organisées et dynamiques. Ce processus dépasse la simple adaptation des pondérations traditionnelles en permettant des changements topologiques dans les réseaux, où de nouvelles entités peuvent apparaître, fusionner ou disparaître.

Le **Deep Synergy Learning (DSL)** s'inscrit dans cette continuité en cherchant à unifier et enrichir les paradigmes existants. Alors que le Deep Learning met l'accent sur la profondeur des représentations, le DSL valorise la richesse et la dynamique des interactions entre les entités d'information. Les bénéfices attendus incluent une meilleure intégration des modalités multiples, une résilience accrue face à l'incertitude et une capacité d'apprentissage continu où le réseau peut évoluer en réponse à de nouvelles données sans nécessiter un réapprentissage complet. En favorisant des interactions flexibles et adaptatives, le DSL ambitionne de jeter les bases d'une intelligence artificielle plus proche des capacités cognitives humaines.

1.1.4. Positionnement du DSL dans le Panorama de l'IA

Le **Deep Synergy Learning (DSL)** s'inscrit dans la continuité des avancées en intelligence artificielle, tout en introduisant une approche novatrice fondée sur la coopération dynamique et l'auto-organisation des entités d'information. Pour mieux comprendre son originalité, il convient de le situer par rapport aux grandes tendances historiques et actuelles. L'IA symbolique, le connexionnisme incarné par le Deep Learning, ainsi que les approches hybrides ou émergentes offrent un cadre permettant d'analyser sa place et son évolution dans le paysage des modèles d'intelligence artificielle.

Le DSL se distingue des systèmes classiques d'IA symbolique, qui reposent sur la formalisation explicite de règles logiques et de bases de connaissances, en permettant l'émergence spontanée de structures et d'organisations adaptées au contexte. Contrairement à la rigidité des approches symboliques, il s'appuie sur des mécanismes auto-adaptatifs qui modifient les interactions en fonction des données disponibles. Par ailleurs, il prolonge l'héritage connexionniste en mettant l'apprentissage au centre de son fonctionnement. Alors que le Deep Learning repose sur des architectures

statiques et hiérarchiques comme les CNN ou Transformers, le DSL introduit une dynamique d’interaction entre les entités, permettant une réorganisation continue des connexions en fonction de leur synergie.

Le DSL puise son inspiration dans les systèmes naturels complexes tels que les écosystèmes ou les réseaux neuronaux biologiques. Ces systèmes se caractérisent par des propriétés globales qui émergent de l’interaction locale d’éléments simples, sans planification centrale. Dans cette perspective, chaque entité d’information dans le DSL peut établir, rompre ou renforcer des connexions en fonction de la synergie qu’elle partage avec d’autres entités. Cette dynamique dépasse les schémas hiérarchiques classiques en introduisant des boucles de rétroaction et en intégrant des structures internes évolutives. Contrairement aux approches traditionnelles où les flux d’information sont souvent indépendants et combinés à un stade tardif, le DSL favorise un tissage constant des flux, améliorant ainsi la robustesse et l’adaptabilité.

Le DSL se distingue également des approches neuro-symboliques qui intègrent explicitement des règles logiques dans des systèmes neuronaux. Bien qu’il ne repose pas directement sur des formalismes logiques, il permet l’émergence de règles implicites via des interactions coopératives et des clusters auto-organisés. De plus, il participe à la quête d’un apprentissage continu en permettant l’évolution permanente des connexions et en favorisant une réorganisation constante en réponse aux nouveaux contextes. Cela en fait une solution prometteuse pour surmonter le problème de l’oubli catastrophique souvent rencontré dans l’apprentissage automatique.

L’IA forte, ou intelligence générale, se caractérise par sa capacité à s’adapter de manière flexible à des environnements variés et non balisés. Le DSL offre une contribution unique à cet objectif en introduisant le principe de coévolution des entités d’information. Cette adaptabilité intrinsèque repose sur l’auto-organisation et la synergie, qui sont considérées comme des éléments essentiels pour l’émergence d’une intelligence créative et autonome. Plutôt que de remplacer le Deep Learning, le DSL peut s’y intégrer en tant que niveau supplémentaire, enrichissant les représentations neuronales classiques par une interactivité et une plasticité accrue.

1.1.5. Rôle Potentiel du DSL vers l’IA Forte (IAG)

L’**Intelligence Artificielle Forte** (IAG), également désignée par intelligence artificielle générale, se définit comme la capacité d’une machine à réaliser des tâches cognitives humaines de manière autonome, flexible et adaptative, en s’appuyant sur une compréhension et un apprentissage comparable à ceux d’un être humain. Cet objectif ultime de l’intelligence artificielle reste encore largement théorique, malgré les avancées impressionnantes des systèmes actuels, notamment ceux basés sur le **Deep Learning**. Ces modèles, bien que performants dans des tâches spécialisées, peinent à atteindre le niveau de polyvalence et de plasticité requis pour prétendre à une véritable IA générale. Le **Deep Synergy Learning (DSL)** se positionne comme une approche novatrice et prometteuse, offrant un cadre conceptuel pour dépasser les limites des modèles traditionnels.

Une intelligence générale ne se limite pas à exceller dans une tâche unique, mais doit pouvoir évoluer dans des environnements variés et imprévisibles. Le DSL se distingue par son aptitude à favoriser l’apprentissage multi-contextuel grâce à ses mécanismes d’**auto-organisation** et de **synergie informationnelle**. Ces mécanismes permettent au système de s’adapter en temps réel à des situations inédites, en reconfigurant les relations entre entités d’information. Contrairement aux systèmes conventionnels, qui restent figés dans leurs architectures ou nécessitent une ingénierie lourde pour évoluer, le DSL peut ajuster sa structure de manière dynamique.

La richesse des interactions entre les entités est une autre caractéristique essentielle du DSL. Plutôt que de juxtaposer des modules spécialisés sans réelle interaction, il propose une approche où chaque flux d’information, qu’il soit visuel, textuel ou sensoriel, interagit et co-évolue avec les autres. Ce type d’interconnexion fluide permet l’émergence de représentations conceptuelles globales, essentielles pour traiter la complexité du monde réel.

Enfin, le DSL intègre une dimension d’**adaptabilité évolutive**, permettant au système de non seulement apprendre, mais également désapprendre, réviser et transformer ses structures en fonction de nouvelles expériences. Cette adaptabilité dépasse celle des architectures classiques, qui, bien qu’elles ajustent leurs pondérations, conservent généralement une topologie statique.

L’un des principes fondamentaux du DSL est l’**auto-organisation**, inspirée des systèmes biologiques. À l’image du cerveau humain, il privilégie une architecture ouverte, où les entités d’information ne suivent pas une trajectoire

linéaire prédéfinie, mais interagissent selon des schémas dynamiques et adaptatifs. Cette capacité à reconfigurer les flux internes permet au système de s'auto-réguler et d'évoluer en fonction des besoins contextuels.

La **plasticité dynamique** du DSL constitue un autre levier crucial pour atteindre l'IA forte. Elle dépasse l'ajustement traditionnel des pondérations en introduisant la possibilité de créer de nouvelles connexions, d'en supprimer d'obsoletes ou de réorganiser des clusters entiers. Cette plasticité ouvre la voie à une exploration continue et à une innovation constante, des caractéristiques indispensables pour développer une intelligence véritablement générale.

Les systèmes d'apprentissage profond ont démontré leur capacité à exceller dans des domaines spécifiques, comme la reconnaissance d'images ou la traduction automatique. Cependant, cette spécialisation s'accompagne d'une incapacité à transférer efficacement les connaissances d'un domaine à un autre. Le DSL propose une alternative en favorisant une **coopération dynamique** entre différentes modalités ou tâches, permettant une synergie globale entre sous-systèmes.

Ce paradigme ouvre la possibilité d'un transfert de connaissances fluide, où les acquis d'une modalité, comme l'analyse visuelle, enrichissent l'apprentissage dans une autre, telle que la compréhension du langage. Cette capacité de réutilisation et de réorganisation constitue une avancée majeure vers une **flexibilité généralisée**, essentielle pour le développement de l'IAG.

Le cerveau humain se distingue par son haut degré d'interconnexion et sa capacité à faire émerger des significations à partir de l'interaction entre différentes aires sensorielles et associatives. Le DSL s'inspire de ce modèle en proposant une architecture où les interactions dynamiques entre les entités d'information jouent un rôle central. Cette approche favorise non seulement l'émergence de comportements complexes, mais également celle de **schémas cognitifs** riches et adaptatifs, proches de ceux observés dans les processus humains.

De plus, le DSL vise à dépasser la simple corrélation statistique en permettant l'émergence de concepts et de significations. Les interactions entre entités d'information, structurées par des mécanismes d'auto-organisation, contribuent à la construction de **nœuds sémantiques**, amorçant ainsi une véritable compréhension du monde environnant.

Bien que le DSL représente une avancée prometteuse, plusieurs défis doivent être relevés pour qu'il puisse concrétiser son potentiel en tant que moteur de l'IAG. L'**efficacité algorithmique** constitue un premier enjeu, car les processus d'auto-organisation et de réorganisation dynamique peuvent être coûteux en termes de calcul. Trouver des solutions d'optimisation est donc essentiel.

L'**équilibre entre stabilité et plasticité** est également un aspect critique. Le système doit éviter de se réorganiser de manière excessive, ce qui pourrait nuire à la cohérence de ses apprentissages, tout en maintenant une capacité suffisante d'adaptation.

Enfin, les questions d'**interprétabilité** et de **sécurité** demeurent cruciales. Une architecture aussi dynamique et complexe pourrait poser des problèmes de traçabilité et de contrôle, notamment dans des domaines sensibles comme la santé ou la finance.

1.1.6. Objectifs et Contributions Clés du DSL

Le Deep Synergy Learning (DSL) s'inscrit comme une évolution fondamentale dans le domaine de l'**intelligence artificielle (IA)**, mettant en avant une **approche dynamique** et hautement **adaptative** pour surmonter les limites des architectures hiérarchiques classiques, telles que celles du **Deep Learning**. Cette nouvelle méthodologie repose sur l'idée centrale de **synergie informationnelle**, où les entités d'information interagissent et s'organisent de manière autonome pour produire des représentations plus riches et efficaces. L'objectif principal du DSL est d'intégrer les principes issus du **connexionnisme** et des **systèmes complexes**, tout en introduisant des mécanismes avancés d'**auto-organisation** et d'**apprentissage évolutif**.

A. Cadre théorique et objectifs fondamentaux

Le DSL repose sur une modélisation qui privilégie les interactions **synergiques** entre les entités d'information. Contrairement aux architectures fixes, où les flux d'information sont généralement dirigés et figés, le DSL permet une

reconfiguration continue des liens internes. Cette flexibilité est rendue possible grâce à des mécanismes d'**évaluation dynamique** de la synergie, notée $S(i, j)$, entre deux entités i et j . Mathématiquement, la synergie entre deux entités peut être décrite comme une fonction dépendant de leurs représentations respectives x_i et x_j :

$$S(i, j) = \Phi(x_i, x_j),$$

où Φ est une fonction d'évaluation qui peut être basée sur la corrélation, l'entropie conjointe, ou toute autre mesure adaptée au contexte.

L'objectif fondamental est de maximiser cette synergie globale dans le système, définie comme une somme pondérée des synergies individuelles :

$$\text{Synergie totale} = \sum_{i \neq j} w_{ij} S(i, j),$$

où w_{ij} représente les poids ajustables des connexions entre i et j . En optimisant cette fonction, le DSL favorise des **interactions pertinentes** et élimine les connexions inutiles ou redondantes, améliorant ainsi l'efficacité computationnelle et la robustesse des représentations générées.

L'un des objectifs clés du DSL est également d'offrir une plateforme pour l'**intégration multimodale**. En incorporant des données issues de différentes modalités, comme le texte, les images ou les signaux temporels, le système est capable d'unifier ces informations de manière **contextuelle**. Chaque modalité contribue activement à l'apprentissage global, tout en bénéficiant des enrichissements apportés par les autres, dans un cadre coopératif.

B. Contributions principales et innovations méthodologiques

Le DSL propose une contribution notable en introduisant un mécanisme d'**auto-organisation** et de **plasticité structurelle**. Contrairement aux approches classiques où les architectures sont déterminées à l'avance, le DSL permet une **reconfiguration dynamique** des connexions entre entités, suivant des critères d'efficacité synergiques. Cette adaptabilité repose sur un processus d'optimisation continue où chaque entité ajuste ses liens en fonction de la pertinence des informations échangées. Ce mécanisme peut être formalisé comme suit :

$$\frac{\partial w_{ij}}{\partial t} = \alpha \cdot \nabla S(i, j),$$

où α est un facteur d'apprentissage, et $\nabla S(i, j)$ désigne le gradient de la synergie entre i et j .

Une autre innovation majeure réside dans la gestion des **clusters auto-organisés**, qui sont des regroupements spontanés d'entités hautement synergiques. Ces clusters peuvent émerger, se fusionner ou se dissoudre en fonction des interactions observées, offrant une **modularité naturelle** au système. Cette capacité de création et de dissolution des clusters renforce la **résilience** et l'**évolutivité**, deux qualités cruciales pour les environnements complexes et dynamiques.

Enfin, le DSL se distingue par sa capacité à capturer des **représentations sémantiques profondes**, en favorisant l'émergence de **patrons globaux** à partir d'interactions locales. Cette propriété rapproche le DSL des processus cognitifs humains, où la compréhension découle souvent d'une synthèse d'informations contextuelles et émergentes.

C. Avantages du DSL

Le DSL présente des avantages significatifs par rapport aux paradigmes d'apprentissage traditionnels. L'un des principaux bénéfices est sa capacité à **maximiser la synergie informationnelle**, ce qui améliore la richesse et la robustesse des représentations apprises. Cette optimisation synergique permet au système d'exploiter pleinement la diversité des données, y compris dans des environnements multi-modaux. Par ailleurs, la **plasticité dynamique** du DSL garantit une adaptabilité exceptionnelle face à des scénarios évolutifs ou imprévisibles, tout en réduisant les besoins en supervision humaine. En outre, son cadre auto-organisé favorise une meilleure **résilience** aux perturbations, grâce à la redondance et à la flexibilité des connexions internes.

D. Limites et défis à surmonter

Malgré ses nombreux atouts, le DSL n'est pas exempt de défis. L'un des obstacles majeurs est la **complexité computationnelle** inhérente à l'évaluation continue des synergies et à la reconfiguration des connexions. Cette complexité pourrait limiter son application à grande échelle, en particulier dans des contextes où les ressources en calcul sont contraintes. De plus, la nature hautement dynamique et adaptative du DSL pose des **problèmes d'interprétabilité**, car il devient difficile de tracer ou d'expliquer précisément les décisions prises par le système. Enfin, le maintien d'un **équilibre entre stabilité et plasticité** est une question délicate, trop de plasticité peut entraîner une instabilité, tandis qu'une structure trop rigide pourrait nuire à l'efficacité globale.

Conclusion

Le **Deep Synergy Learning** se présente comme une avancée majeure dans le domaine de l'IA, en introduisant des mécanismes d'interaction et d'adaptation qui rompent avec les approches classiques. Grâce à sa capacité à maximiser les synergies informationnelles et à favoriser une auto-organisation dynamique, il ouvre de nouvelles perspectives pour la création de systèmes intelligents, flexibles et robustes. Cependant, pour atteindre son plein potentiel, il sera nécessaire de surmonter les défis liés à sa complexité computationnelle et à son explicabilité, tout en explorant des applications pratiques dans des environnements réels.

1.1.7. Plan Général du Chapitre

La présente section a pour objectif de donner une vision d'ensemble de la **structure** et de la **logique** qui sous-tendent le Chapitre 1 dans son intégralité. Après avoir introduit le contexte et la motivation du Deep Synergy Learning (DSL) dans la section 1.1, nous poursuivrons en explorant les différents volets nécessaires pour comprendre et situer le DSL dans l'écosystème de l'intelligence artificielle.

1.2. Définitions et Concepts Préliminaires

- **1.2.1. Qu'est-ce qu'une “Entité d'Information” ?**
- **1.2.2. Notion de “Synergie Informationnelle”**
- **1.2.3. Différence entre Interaction, Synergie et Corrélation**
- **1.2.4. Approche Hiérarchique vs Approche Auto-Organisée**
- **1.2.5. Réseaux Neuronaux Traditionnels vs Réseaux Synergiques**
- **1.2.6. Terminologies Employées dans le DSL**
- **1.2.7. Exemples Illustratifs de la Synergie dans la Nature**

Cette partie clarifie les notions de base liées au **DSL**. Elle définit les **entités d'information**, explicite la distinction entre une **simple interaction** et une **synergie**, et propose des **exemples concrets** afin d'illustrer la pertinence de cette approche.

1.3. Importance de l'Auto-Organisation

- **1.3.1. Inspirations Biologiques et Cognitives**
- **1.3.2. Concepts Clés : Émergence, Auto-Régulation, Feedback**
- **1.3.3. Comparaison avec les Méthodes d'Apprentissage Classiques**

- **1.3.4. Le Rôle des Flux d'Information Multimodaux**
- **1.3.5. Évolution Dynamique et Adaptation en Continu**
- **1.3.6. Impacts sur la Robustesse et la Résilience des Modèles**
- **1.3.7. Perspectives pour une Approche plus Globale de l'Apprentissage**

Nous soulignerons ici pourquoi l'auto-organisation est cruciale pour le DSL, en nous inspirant notamment des systèmes biologiques et cognitifs. Nous verrons en quoi elle se distingue des méthodes d'apprentissage habituelles et comment elle améliore robustesse et résilience.

1.4. Architecture Générale du DSL

- **1.4.1. Principe de Base : Entités et Liens Synergiques**
- **1.4.2. Présentation du Synergistic Connection Network (SCN)**
- **1.4.3. Notion de Cluster et de Macro-Cluster**
- **1.4.4. Fonctions de Synergie : Distance, Similarité et Co-Information**
- **1.4.5. Pondérations Adaptatives et Évolution Temporelle**
- **1.4.6. Interactions Directes et Indirectes**
- **1.4.7. Cas Particuliers : Synergie binaire et n-aire**

Cette section décrit la structure interne du DSL. Nous y découvrirons notamment le **Synergistic Connection Network (SCN)**, ainsi que les mécanismes de mise à jour et de formation des clusters d'entités d'information.

1.5. Pourquoi une Approche Synergique ?

- **1.5.1. Avantages par Rapport aux Réseaux Neuronaux Profonds**
- **1.5.2. Gestion Naturelle de la Multi-modalité**
- **1.5.3. Flexibilité vis-à-vis des Données Incomplètes ou Bruitées**
- **1.5.4. Potentiel d'Auto-Évolution et d'Adaptation Continue**
- **1.5.5. Réduction de la Dépendance à la Supervision Humaine**
- **1.5.6. Crédit de Représentations Riches et plus Interprétables**
- **1.5.7. Intégration de Dimensions Symboliques ou Cognitives**

Ici, nous discutons des bénéfices concrets du DSL par rapport aux approches traditionnelles. Nous verrons comment la synergie facilite la fusion de modalités, l'adaptation aux données bruitées et la création de représentations plus expressives.

1.6. Applications Pressenties et Domaines Impactés

- **1.6.1. Vision Artificielle et Reconnaissance d'Objets Complexes**
- **1.6.2. Analyse Audio et Traitement du Langage Naturel**
- **1.6.3. Robotique et Systèmes Intelligents Évolutifs**

- **1.6.4. Recommandation Personnalisée et Systèmes de Décision**
- **1.6.5. Surveillance, Diagnostic Médical et Anomalies**
- **1.6.6. Planification et Optimisation dans l'Industrie 4.0**
- **1.6.7. Perspectives pour la Recherche Fondamentale en IA Forte**

Nous dresserons un panorama des domaines susceptibles de bénéficier du DSL, en soulignant pour chacun d'entre eux comment la synergie et l'auto-organisation peuvent apporter des solutions nouvelles ou plus performantes.

1.7. Défis, Contraintes et Ouvertures

- **1.7.1. Complexité Computationnelle et Scalabilité**
- **1.7.2. Qualité et Disponibilité des Données**
- **1.7.3. Développement d'Algorithmes d'Optimisation Appropriés**
- **1.7.4. Contrôle et Stabilité des Processus Auto-Organisés**
- **1.7.5. Interprétabilité et Explicabilité pour l'Humain**
- **1.7.6. Considérations Éthiques et Réglementaires**
- **1.7.7. Comparaisons Expérimentales avec d'Autres Approches**

Malgré son **potentiel**, le **DSL** rencontre plusieurs **obstacles**, notamment la **complexité des calculs**, les **difficultés de collecte de données** et les **incertitudes éthiques**. Cette section propose une **synthèse** de ces problématiques et explore des **pistes** permettant d'y remédier.

1.8. Positionnement du DSL dans l'Évolution de l'IA

- **1.8.1. IA Symbolique vs IA Sub-symbolique : Intégration Potentielle**
- **1.8.2. DSL et Apprentissage Profond : Collaboration ou Substitution ?**
- **1.8.3. Approches Hybrides : DSL, RL (Reinforcement Learning) et Logique**
- **1.8.4. Rôle de la Mémoire et de l'Attention dans le DSL**
- **1.8.5. Tendances Futures : Vers une IA Forte, Consciente ?**
- **1.8.6. Effet Sur la Recherche Interdisciplinaire**
- **1.8.7. Exemple de Convergence : DSL & Neurosciences**

Nous reviendrons sur la relation du DSL avec les grands axes de l'IA (symbolique, connexionniste, hybride) et discuterons de sa place actuelle et future dans l'avancée de la discipline vers l'IA générale.

1.9. Méthodologie, Ressources et Outils

- **1.9.1. Approche de Recherche : Théorique, Expérimentale, Hybride**
- **1.9.2. Bases de Données et Plates-formes de Test pour le DSL**
- **1.9.3. Frameworks de Développement : Python, C++, Librairies spécialisées**

- **1.9.4. Environnements de Simulation et d'Évaluation**
- **1.9.5. Protocoles de Validation : Qualitatifs et Quantitatifs**
- **1.9.6. Collaboration et Partage de Ressources (Open Source)**
- **1.9.7. Gestion du Cycle de Vie d'un Projet DSL**

Cette partie expose les moyens pratiques pour mettre en œuvre et évaluer le DSL : choix des plateformes logicielles, types de données disponibles, protocoles de validation... Elle servira de guide méthodologique pour les projets s'appuyant sur la synergie informationnelle.

1.10. Structure du Livre et Lecture Conseillée

- **1.10.1. Vue d'Ensemble des Chapitres Suivants**
- **1.10.2. Lien entre Chapitres et Cohérence Globale**
- **1.10.3. Logique Pédagogique : Progression des Concepts**
- **1.10.4. Conseils de Lecture en Fonction des Profils (Débutants, Experts)**
- **1.10.5. Renvois vers des Ressources Complémentaires**
- **1.10.6. Planification du Lecteur : Stratégie d'Étude**
- **1.10.7. Perspectives de Recherche à Long Terme**

Ici, nous présenterons la trame globale de l'ouvrage, en expliquant comment chaque chapitre s'articule pour apporter une compréhension complète du DSL, que l'on soit débutant ou spécialiste confirmé.

1.11. Conclusion du Chapitre

- **1.11.1. Synthèse des Points Clés**
- **1.11.2. Contributions du Chapitre à la Compréhension du DSL**
- **1.11.3. Liaison avec le Chapitre 2 : Fondements Théoriques Avancés**
- **1.11.4. Points de Discussion et Questions Ouvertes**
- **1.11.5. Réflexion sur la Place du DSL dans l'IA Généralisée**
- **1.11.6. Mot de Fin et Transition vers la Suite**

Enfin, nous dresserons le bilan des notions abordées, mettrons en évidence les contributions majeures de ce premier chapitre, et préparerons la transition vers le chapitre suivant, qui approfondira les fondements théoriques et mathématiques du **Deep Synergy Learning**.

Ce **Plan Général du Chapitre 1** doit permettre au lecteur de repérer rapidement les différentes thématiques abordées et de comprendre la **progression logique** qui va du contexte initial de l'IA (sections 1.1 à 1.2) aux **concepts structurels** du DSL (sections 1.3 à 1.5), puis à leurs **applications** (1.6), défis (1.7), positionnement (1.8) et **méthodologies** (1.9), avant de conclure sur l'organisation globale de l'ouvrage (1.10) et de refermer le chapitre (1.11).

1.1. Contexte et Motivation

Cette section introduit les raisons d'être du Deep Synergy Learning. Elle part d'un historique de l'IA (1.1.1) pour montrer l'émergence du Deep Learning (1.1.2) et ses limites, puis explique la transition vers l'idée de synergie informationnelle (1.1.3). Le **positionnement** du DSL dans l'écosystème de l'IA (1.1.4), son **rôle potentiel** dans la quête d'une IA forte (1.1.5) et ses **objectifs et contributions clés** (1.1.6) y sont détaillés. La sous-section 1.1.7 conclut en résumant la structure globale de cette partie introductive.

1.2. Définitions et Concepts Préliminaires

Après avoir cerné les motivations du DSL, on aborde ici les **fondations conceptuelles**. L'accent est mis sur la définition des "entités d'information" (1.2.1), la notion de "synergie informationnelle" (1.2.2), et les différences entre interaction, synergie et corrélation (1.2.3). On compare également l'**approche hiérarchique** traditionnelle aux mécanismes **auto-organisés** (1.2.4), en distinguant les **réseaux neuronaux classiques** des **réseaux synergiques** (1.2.5). Un point terminologique (1.2.6) clarifie le vocabulaire clé, avant de donner quelques **exemples illustratifs** de synergie dans la nature (1.2.7) pour mieux ancrer ces concepts.

1.3. Importance de l'Auto-Organisation

Le **DSL** met fortement l'accent sur l'**auto-organisation**, un principe fondamental de son fonctionnement. Cette section en précise la **portée** ainsi que ses **inspirations** (1.3.1), en mettant en lumière des **concept clés** tels que l'**émergence** et l'**auto-régulation** (1.3.2). On y compare la démarche auto-organisée avec les méthodes d'apprentissage classiques (1.3.3), en insistant sur le rôle crucial des **flux multimodaux** (1.3.4). L'**évolution dynamique** (1.3.5) et l'impact sur la **robustesse** (1.3.6) figurent parmi les arguments forts pour justifier ce choix d'organisation. La section se termine sur une réflexion quant à l'adoption d'une approche plus globale de l'apprentissage (1.3.7).

1.4. Architecture Générale du DSL

On explore ensuite la **structure interne** du Deep Synergy Learning. On démarre par les **principes de base** (1.4.1), puis on décrit le **Synergistic Connection Network (SCN)** (1.4.2), pivot du DSL, mettant l'accent sur la **notion de cluster** (1.4.3). Les **fonctions de synergie** et les mesures de distance ou similarité (1.4.4) sont présentées, suivies des **pondérations adaptatives** (1.4.5) qui évoluent au fil du temps. On souligne ensuite la distinction entre **interactions directes et indirectes** (1.4.6), et on conclut en mentionnant les **cas particuliers** (1.4.7) comme la synergie binaire ou n-aire, ouvrant la voie à des extensions plus avancées.

1.5. Pourquoi une Approche Synergique ?

Cette section développe les avantages potentiels du DSL. Elle souligne par exemple sa plus-value par rapport aux **réseaux neuronaux profonds** (1.5.1), sa gestion naturelle de la **multi-modalité** (1.5.2), et la **flexibilité** qu'il apporte face aux données bruyantes ou incomplètes (1.5.3). Viennent ensuite des points sur l'**auto-évolution** (1.5.4), la réduction de la **dépendance à la supervision** (1.5.5), l'émergence de **représentations plus riches** (1.5.6) et, enfin, l'éventuelle intégration de **dimensions symboliques** (1.5.7).

1.6. Applications Pressenties et Domaines Impactés

Ici, on brosse un **tableau des champs d'application** où le DSL peut exceller. On commence par la **vision artificielle** et la **reconnaissance d'objets** (1.6.1), l'**analyse audio** et le **traitement du langage** (1.6.2), la **robotique** (1.6.3), les **systèmes de recommandation** (1.6.4), la **surveillance** et le **diagnostic médical** (1.6.5), puis on aborde les problématiques de **planification** et **d'optimisation** dans l'industrie (1.6.6). La section s'achève en évoquant les implications pour l'**IA forte** (1.6.7).

1.7. Défis, Contraintes et Ouvertures

Même si le DSL est porteur de promesses, il soulève nombre de **défis**. On évoque notamment la **complexité computationnelle** (1.7.1), la qualité des **données** (1.7.2), la nécessité de **nouveaux algorithmes** (1.7.3), et les questions de **stabilité** dans les processus auto-organisés (1.7.4). Des enjeux d'**interprétabilité** (1.7.5) et d'**éthique** (1.7.6) sont également soulevés, avant de proposer des pistes de comparaison expérimentale (1.7.7) avec d'autres approches.

1.8. Positionnement du DSL dans l'Évolution de l'IA

On approfondit ici la **place** du DSL dans la grande histoire de l'IA, soulignant les possibilités d'**intégration** entre IA symbolique et sub-symbolique (1.8.1), les relations entre DSL et **Deep Learning** (1.8.2), et les perspectives d'**approches hybrides** (1.8.3). Le **rôle de la mémoire** et de l'**attention** (1.8.4), les pistes vers l'**IA forte et consciente** (1.8.5), ainsi que l'influence sur la **recherche interdisciplinaire** (1.8.6) sont passés en revue. Enfin, un **exemple de convergence** (1.8.7) montre comment le DSL peut s'enrichir des neurosciences.

1.9. Méthodologie, Ressources et Outils

Cette section vise à orienter le lecteur sur la façon de **mettre en œuvre** le DSL. On y aborde les approches de recherche (1.9.1), les **bases de données** disponibles (1.9.2) et les **frameworks** courants (1.9.3), ainsi que les environnements de **simulation** et d'**évaluation** (1.9.4). Des **protocoles de validation** (1.9.5), la **collaboration open source** (1.9.6), et la **gestion de projet** (1.9.7) sont enfin exposés, pour accompagner toute personne souhaitant expérimenter avec le DSL.

1.10. Structure du Livre et Lecture Conseillée

Avant de clore le chapitre, on donne ici un **aperçu global** de l'enchaînement des chapitres du livre (1.10.1), en expliquant la **cohérence** d'ensemble (1.10.2) et la **logique pédagogique** adoptée (1.10.3). On propose également des **conseils** selon les profils de lecteurs (1.10.4), indique des **ressources complémentaires** (1.10.5) et suggère une **stratégie d'étude** (1.10.6). La section se termine par quelques **perspectives de recherche** (1.10.7).

1.11. Conclusion du Chapitre

Enfin, la conclusion du chapitre (1.11) dresse une **synthèse** (1.11.1) des points essentiels et met en évidence la **contribution** (1.11.2) de ce chapitre à la compréhension du DSL. On y établit une **liaison** claire avec le chapitre suivant (1.11.3), en soulignant les **questions ouvertes** (1.11.4) et les **réflexions** (1.11.5) sur la place du DSL dans l'IA généralisée. Un **mot de fin** (1.11.6) assure la transition vers la suite du livre, où seront approfondis les fondements théoriques et les premières implémentations concrètes.

1.2. Définitions et Concepts Préliminaires

Dans cette partie, nous allons introduire les notions fondamentales qui sous-tendent le **Deep Synergy Learning (DSL)**. Alors que la section précédente (1.1) portait sur le contexte historique, les motivations et la place du DSL dans l'écosystème de l'IA, le présent segment (1.2) est consacré à la **structure conceptuelle** du DSL et aux **fondements** qui permettent d'en comprendre les mécanismes profonds.

Nous débuterons par la définition de ce qu'est une “**Entité d'Information**” (1.2.1), pierre angulaire du paradigme DSL. Nous poursuivrons en expliquant le concept de “**Synergie Informationnelle**” (1.2.2) et en clarifiant la différence entre interaction, synergie et simple corrélation (1.2.3). Nous verrons ensuite comment le DSL se démarque d'une **approche strictement hiérarchique** (1.2.4) et en quoi il diverge des **réseaux neuronaux traditionnels** (1.2.5). Enfin, nous préciserons la **terminologie** spécifique (1.2.6) et illustrerons les principes de synergie par des **exemples concrets** (1.2.7).

1.2.1. Qu'est-ce qu'une “Entité d'Information” ?

Le concept d'**entité d'information** constitue l'un des piliers fondamentaux du **Deep Synergy Learning**. Dans le DSL, une entité d'information n'est pas un simple point de données isolé, mais plutôt un “objet d'apprentissage” possédant :

- Une **représentation interne** (généralement un vecteur, un tenseur ou une fonction).
- Les **caractéristiques dynamiques** d'une entité lui permettent d'**évoluer**, de se **modifier** et d'**entretenir** des **liens synergiques** avec d'autres entités.
- Un **historique** (ou une mémoire partielle) de ses interactions antérieures, pouvant impacter son comportement futur.

Pour formaliser une entité d'information \mathcal{E} , on se place dans un espace vectoriel (ou parfois un espace de Hilbert plus général) :

$$\mathcal{E}_k \in \mathbb{R}^d, \quad \text{ou} \quad \mathcal{E}_k \in \mathbb{R}^{n_1 \times n_2 \times \dots \times n_p},$$

suivant la nature des données (vecteur, matrice, tenseur, etc.). Par exemple, une image peut être encodée en tant que tenseur 3D (hauteur \times largeur \times canaux de couleur), tandis qu'un signal audio pourra être représenté sous forme de séries temporelles dans \mathbb{R}^d .

Dans certains cas, la représentation peut également être probabiliste. Ainsi, \mathcal{E}_k peut être décrite par une distribution de probabilité $\mathcal{P}_k(\mathbf{x})$ sur un certain espace $\mathbf{x} \in \mathcal{X}$. L'important est de conserver la possibilité de **mesurer** la distance, la similarité ou la divergence entre deux entités :

$$\text{dist}(\mathcal{E}_k, \mathcal{E}_m) \quad \text{ou} \quad \text{sim}(\mathcal{E}_k, \mathcal{E}_m).$$

Outre la représentation brute, une entité peut avoir des **paramètres internes** (poids, biais, etc.) qui se modifient selon le temps ou selon les interactions :

$$\theta_k = \{\theta_{k,1}, \theta_{k,2}, \dots, \theta_{k,\ell}\}.$$

Ces paramètres influent sur le “comportement” de l'entité, c'est-à-dire sa manière de calculer des **scores de similarité** ou des **fonctions de sortie**. On peut aussi décrire un **état** interne $\mathbf{s}_k(t)$ évoluant avec t , le temps (ou la phase d'apprentissage) :

$$\mathbf{s}_k(t) \in \mathbb{R}^d,$$

indiquant, par exemple, le niveau de confiance ou les caractéristiques discriminantes apprises jusqu'à l'instant t . Cet état peut servir de base à la mise à jour des **connexions synergiques** entre l'entité \mathcal{E}_k et d'autres entités \mathcal{E}_m .

Dans le cadre du DSL, nous pouvons associer à chaque entité \mathcal{E}_k un **ensemble** de composants :

$$\mathcal{E}_k = (\mathbf{x}_k, \mathbf{s}_k, \theta_k, \dots),$$

où :

- $\mathbf{x}_k \in \mathbb{R}^d$ est la représentation courante (ex. : vecteur de caractéristiques, image encodée).
- \mathbf{s}_k est l'état interne dynamique (optionnel ou modulable).
- θ_k représente des **paramètres d'ajustement** ou d'**apprentissage**.
- D'autres composants pourraient inclure la **mémoire** (historique), les **métadonnées**, etc.

Cette formulation a pour but de **généraliser** la notion de “neurone” ou de “vecteur de données” pour en faire une entité d’apprentissage **active** et **adaptative**, au cœur des mécanismes d’interaction synergiques.

Pour illustrer concrètement la notion d’entité d’information, considérons la tâche de **reconnaissance de situations** dans une vidéo associée à un flux audio. Dans ce scénario, l’entité $\mathcal{E}_{\text{visuelle}}$ peut représenter un **descripteur d’image** extrait par un réseau de neurones convolutionnel (CNN), enrichi de **paramètres** relatifs à la forme ou à la pose des objets détectés. De son côté, l’entité $\mathcal{E}_{\text{auditive}}$ peut regrouper une **carte d’intensité fréquentielle** (spectrogramme) et un **état** décrivant la tonalité ou le niveau de bruit ambiant.

Ces deux entités ne constituent pas de simples “blocs” de données isolés, elles sont conçues pour **interagir**, se **synchroniser** ou même **fusionner**, dès lors que leur **synergie** (au sens de la section 1.2.2) est suffisamment élevée. Autrement dit, si les informations issues des canaux visuel et auditif s’enrichissent mutuellement, elles ont la possibilité de renforcer leurs liens et, potentiellement, de s’intégrer au point de former une entité commune, apte à traiter des signaux audiovisuels de manière coordonnée. Cette démarche souligne la **plasticité** du Deep Synergy Learning, qui autorise une reconfiguration permanente des relations entre entités pour améliorer la représentation globale de la scène.

En définitive, l’entité d’information constitue le **nœud élémentaire** du DSL. C’est à **travers** elle et **par** elle que les mécanismes de synergie prennent forme, permettant l’émergence de structures d’apprentissage plus complexes. Le **design** même de chaque entité, qu’il s’agisse de sa représentation (vecteur, tenseur, distribution de probabilité), de son état ou de ses paramètres d’ajustement, détermine directement l’**expressivité** et l’**efficacité** de l’apprentissage au sein du réseau. Le choix judicieux de ces attributs, adapté à la modalité (vision, audio, texte, etc.) et à la tâche visée, facilite la formation de **liaisons synergiques** fructueuses et, par conséquent, contribue à la robustesse et à la performance globale du système DSL.

1.2.2. Notion de “Synergie Informationnelle”

L’un des concepts centraux du Deep Synergy Learning (DSL) est la **synergie informationnelle**. Il s’agit de la capacité de deux (ou plusieurs) entités d’information à produire, ensemble, un **contenu** ou une **performance** impossible à atteindre (ou significativement moins bonne) si elles agissaient de manière isolée. Dans un cadre mathématique, la synergie se formalise par une mesure qui évalue l’**apport mutuel** entre les entités. Plus cette mesure est élevée, plus les entités concernées s’enrichissent mutuellement, amplifiant leur pouvoir de représentation ou de décision.

Définition générale. Considérons deux entités d’information, \mathcal{E}_i et \mathcal{E}_j . En première approximation, on peut définir la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ comme une fonction qui quantifie à quel point la prise en compte conjointe de \mathcal{E}_i et de \mathcal{E}_j **améliore** un critère d’apprentissage (la prédiction d’une variable cible, la qualité d’une représentation, etc.).

$$S(\mathcal{E}_i, \mathcal{E}_j) = f(\mathcal{E}_i, \mathcal{E}_j),$$

où $f(\cdot)$ peut être :

- Une **mesure d’entropie conjointe** (ou de co-information) en théorie de l’information,
- Un **gain de performance** par rapport à une référence (p. ex. différence de log-vraisemblance),

- Une **fonction de similarité/distance** qui prend en compte des aspects non linéaires et adaptatifs.

Cette fonction f doit être conçue pour refléter la notion que “le tout est plus que la somme des parties”. Ainsi, il est d’usage de considérer qu’une **haute synergie** indique que l’association $\{\mathcal{E}_i, \mathcal{E}_j\}$ est nettement plus informative que chacune des entités prise isolément.

En théorie de l’information, on peut s’appuyer sur l'**entropie conjointe** et la **co-information**. Par exemple, si \mathbf{X}_i et \mathbf{X}_j sont les variables aléatoires (représentant respectivement les entités \mathcal{E}_i et \mathcal{E}_j), on définit :

$$I(\mathbf{X}_i; \mathbf{X}_j) = H(\mathbf{X}_i) + H(\mathbf{X}_j) - H(\mathbf{X}_i, \mathbf{X}_j),$$

où $H(\cdot)$ est l’entropie (de Shannon, ou d’autres formes d’entropie plus générales). Cette quantité $I(\mathbf{X}_i; \mathbf{X}_j)$ mesure l’**information mutuelle** entre \mathbf{X}_i et \mathbf{X}_j . Toutefois, l’information mutuelle standard ne distingue pas toujours la **synergie** de la **redondance**.

Pour caractériser la synergie stricto sensu, plusieurs travaux de théorie de l’information proposent des mesures de **co-information** plus élaborées, voire des “Partial Information Decomposition” (PID), qui visent à séparer la part de redondance et la part de synergie :

$$\text{Synergie}(\mathbf{X}_i, \mathbf{X}_j) = I_{\text{PID}}^{(\text{syn})}(\mathbf{X}_i; \mathbf{X}_j \mid \mathbf{Y}),$$

où \mathbf{Y} peut être une troisième variable (cible à prédire) ou un contexte. Dans le cadre du DSL, il est donc pertinent d’utiliser, lorsque c’est possible, des **métriques entropiques** pour quantifier la contribution **non triviale** de chaque couple d’entités.

Une autre approche consiste à définir la synergie comme le **gain** (en termes de fonction objectif ou de performance) obtenu lorsqu’on associe deux entités, par rapport à leur utilisation séparée :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \Delta(\mathcal{E}_i, \mathcal{E}_j) = \Phi(\{\mathcal{E}_i, \mathcal{E}_j\}) - [\Phi(\{\mathcal{E}_i\}) + \Phi(\{\mathcal{E}_j\})],$$

où $\Phi(\cdot)$ est un **score** ou une **mesure** de la qualité du système (ex. : taux de classification, log-likelihood, etc.). Dans ce cas :

- $S(\mathcal{E}_i, \mathcal{E}_j) > 0$ signifie qu’il y a véritablement une **valeur ajoutée** à combiner \mathcal{E}_i et \mathcal{E}_j .
- $S(\mathcal{E}_i, \mathcal{E}_j) < 0$ indique qu’il y a **inhibition** ou dégradation mutuelle.
- $S(\mathcal{E}_i, \mathcal{E}_j) = 0$ suggère une **indépendance** ou une simple addition sans synergie.

Cette formulation est souvent utilisée en pratique, car elle s’aligne directement sur un **objectif** (objectif supervisé, critère d’optimisation non supervisé, etc.). On peut de plus pondérer cette synergie par un facteur adaptatif, en tenant compte du **contexte temporel** ou des autres entités impliquées.

Contrairement aux approches linéaires ou statiques, le DSL prévoit que la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ soit **évolutive** au cours du temps. En d’autres termes, le réseau peut réévaluer en continu la contribution mutuelle de \mathcal{E}_i et \mathcal{E}_j . Matériellement, cela se traduit par la mise à jour d’une **pondération synergique** $\omega_{i,j}(t)$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \cdot [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \cdot \omega_{i,j}(t)],$$

où η est un taux d’apprentissage, τ un terme de régularisation (ou d’oubli). Plus la synergie entre deux entités est forte, plus leur lien s’intensifie. Au contraire, si ce lien n’apporte guère de valeur ajoutée (ou est carrément nuisible), sa pondération peut diminuer et aller jusqu’à **rompre** la connexion.

Définition Synergie binaire, ternaire, et n-aire. Dans sa version la plus simple, on considère la synergie entre **paires** d’entités $(\mathcal{E}_i, \mathcal{E}_j)$. Toutefois, nombreux de scénarios exigent d’évaluer la synergie entre plusieurs entités simultanément. Dans ce cas, on généralise S à un ensemble $\{\mathcal{E}_{k_1}, \dots, \mathcal{E}_{k_m}\}$. On parle alors de **synergie n-aire**, dont la mesure n’est pas

forcément la somme des synergies binaires. En effet, il se peut qu'**une triple** $\{\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c\}$ dégage une synergie supérieure à la somme des synergies de ses paires :

$$S(\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c) > S(\mathcal{E}_a, \mathcal{E}_b) + S(\mathcal{E}_b, \mathcal{E}_c) + S(\mathcal{E}_a, \mathcal{E}_c).$$

Ce phénomène traduit la nature **émergente** du DSL, où des ensembles d'entités peuvent coopérer de manière non triviale pour engendrer de nouvelles représentations ou actions.

Exemple scénario multimodal.

Soit $\mathcal{E}_{\text{visuelle}}$ (extraction de caractéristiques d'une image), $\mathcal{E}_{\text{auditives}}$ (traits de voix, intonation) et $\mathcal{E}_{\text{textuelles}}$ (mots-clés extraits d'une transcription).

Si $\mathcal{E}_{\text{visuelle}}$ et $\mathcal{E}_{\text{textuelle}}$ ont peu d'information en commun, leur **information mutuelle** peut être faible. Pourtant, prises ensemble, elles peuvent produire un **contexte** (ex. : "lieu de la scène + thèmes abordés verbalement") qui aide à l'interprétation des sons (détection d'émotion). Autrement dit, c'est l'**intersection** de ces informations qui devient cruciale, expliquant une **synergie** plus forte lorsqu'on combine ces trois entités plutôt que deux à deux.

Importance pour le DSL.

Le concept de synergie informationnelle est ce qui **diffrerencie** le DSL d'un système où l'on se contenterait de propager les données entre couches. Au contraire, dans le DSL :

Les entités **cherchent** activement des partenaires synergiques,

Les **pondérations** entre elles **s'ajustent** en fonction de la synergie,

Les **clusters** ou micro-réseaux émergent autour des synergies les plus fortes,

Les entités peuvent **fusionner** ou **évoluer** pour mieux exploiter la coopération (nous verrons ces points dans les chapitres suivants).

Ainsi, la synergie agit comme un **moteur** d'auto-organisation et de **dynamique adaptative**, permettant au réseau de se **restructurer** au fil du temps, en valorisant les combinaisons d'entités les plus porteuses d'information ou de gain de performance.

1.2.3. Différence entre Interaction, Synergie et Corrélation

Lorsqu'on étudie les relations entre différentes entités d'information, il est essentiel de faire la distinction entre **interaction**, **corrélation** et **synergie**. Ces notions sont parfois utilisées de façon interchangeable, mais elles renvoient à des réalités mathématiques et conceptuelles différentes. Comprendre ces nuances permet de mieux cerner l'originalité du Deep Synergy Learning (DSL) et la portée de son concept de « synergie informationnelle ».

1.2.3.1. Interaction : une relation générique

Le terme **interaction** désigne de manière générale l'influence mutuelle que peuvent exercer deux éléments (ou plus) l'un sur l'autre. D'un point de vue mathématique, on parle souvent d'**interaction** lorsque le comportement (ou la fonction) d'un système dépend de l'état de plusieurs variables de manière **non indépendante** :

$$f(\mathbf{x}_1, \mathbf{x}_2) \neq f_1(\mathbf{x}_1) + f_2(\mathbf{x}_2).$$

Par exemple, dans un modèle statistique de type régression, l'**effet d'interaction** entre deux variables se traduit par la présence d'un terme produit $\mathbf{x}_1 \times \mathbf{x}_2$.

Une interaction ne garantit pas nécessairement un effet bénéfique ou un « plus » collectif ; elle se borne à signaler que l'état ou la valeur prise par \mathbf{x}_2 modifie l'effet de \mathbf{x}_1 (et inversement).

Ainsi, dans le DSL, de simples **interactions** peuvent exister entre des entités d'information sans pour autant impliquer une **synergie** (cette dernière requérant un effet d'émergence véritable, voir plus bas).

1.2.3.2. Corrélation : dépendance statistique (souvent linéaire)

La **corrélation** (au sens commun) mesure le degré de **dépendance statistique** entre deux variables, souvent réduit à la **corrélation linéaire** de Pearson :

$$\rho(\mathbf{X}, \mathbf{Y}) = \frac{\text{cov}(\mathbf{X}, \mathbf{Y})}{\sigma_{\mathbf{X}} \cdot \sigma_{\mathbf{Y}}}$$

avec $\text{cov}(\cdot, \cdot)$ représente la covariance, $\sigma_{\mathbf{X}}$ et $\sigma_{\mathbf{Y}}$ désignent l'écart-type de \mathbf{X} et \mathbf{Y} .

Une corrélation élevée ($\rho \approx 1$ ou $\rho \approx -1$) signifie que deux variables évoluent de façon similaire (linéairement liées), tandis qu'une corrélation nulle ($\rho \approx 0$) indique l'absence de dépendance linéaire (mais pas forcément l'absence de dépendance tout court).

Remarque : Dans un cadre non linéaire, d'autres mesures (mutual information, distance correlation, etc.) peuvent s'avérer plus pertinentes que la simple corrélation linéaire.

Corrélation ≠ synergie.

Dans le cas d'une corrélation forte, on observe souvent une **redondance**. Si \mathbf{X} prédit bien \mathbf{Y} , alors \mathbf{Y} n'apporte pas nécessairement de nouvelle information. La corrélation peut aussi être trompeuse (corrélation de variables bruitées, effet de causalité inverse, variables cachées...).

Dans le DSL, deux entités très corrélées peuvent d'ailleurs être moins intéressantes (peu de gain) que deux entités faiblement corrélées, mais dont la **combinaison** génère un contenu supplémentaire.

Ainsi, une **corrélation forte** n'implique pas forcément une **synergie** ; et inversement, deux entités peuvent ne pas être corrélées mais créer, ensemble, un effet émergent.

La **synergie**, telle que définie dans le DSL, suppose un **gain** ou une **valeur ajoutée** lorsque les entités se combinent, au-delà de ce qu'elles apportent chacune de leur côté. Autrement dit :

$$S(\mathcal{E}_i, \mathcal{E}_j) > 0 \Rightarrow \text{La combinaison } \{\mathcal{E}_i, \mathcal{E}_j\} \text{ vaut plus que la somme séparée.}$$

- **Synergie ≠ simple interaction** : L'interaction signale simplement une dépendance réciproque, alors que la synergie suppose qu'un nouveau niveau de fonctionnalité ou d'information émerge.
- **Synergie ≠ redondance** : Deux variables très similaires (corrélées) ont peu de synergie, car prendre l'une ou l'autre n'ajoute pas grand-chose à la décision globale.
- **Synergie ≠ coïncidence** : Les coïncidences peuvent être éphémères et non reproductibles. La synergie implique un **effet régulier et réel** sur l'optimisation ou la représentation interne du système.

Matériellement, dans un modèle où la fonction de coût \mathcal{L} est à minimiser (ou la fonction de performance Φ à maximiser), la synergie entre \mathcal{E}_i et \mathcal{E}_j s'exprime souvent comme :

$$\Delta_{ij} = \Phi(\{\mathcal{E}_i, \mathcal{E}_j\}) - [\Phi(\{\mathcal{E}_i\}) + \Phi(\{\mathcal{E}_j\})].$$

- Si $\Delta_{ij} > 0$, on parle de **synergie positive** (la combinaison est plus utile que la simple juxtaposition).
- Si $\Delta_{ij} < 0$, il y a **inhibition** ou **redondance néfaste**, et le couplage des entités se révèle contre-productif.

- Si $\Delta_{ij} \approx 0$, cela signifie qu'elles n'apportent pas grand-chose l'une à l'autre au regard de la tâche considérée.

Pour mieux illustrer ces différences, on peut imaginer un **diagramme** représentant trois situations :

1. **Corrélation (redondance)** : Les deux entités (A et B) apportent presque la même information.
2. **Interaction** : A et B se modifient mutuellement, mais sans forcément créer une nouvelle dimension.
3. **Synergie** : La combinaison A + B engendre un nouveau potentiel (ex. : l'ajout d'un flux audio + flux visuel crée un contexte multimodal plus riche que n'importe lequel des flux pris isolément).

Cas concrets.

- **Cas de redondance sans synergie** : Deux capteurs de température placés au même endroit, fournissant des mesures quasi identiques. Ils sont très corrélés, mais en prendre un seul est aussi informatif que d'en prendre deux.
- **Cas d'interaction sans synergie** : En biologie, certaines protéines interagissent (l'une bloque l'autre, par exemple), mais cela ne produit pas nécessairement un comportement globalement plus efficace pour l'organisme.
- **Cas de synergie forte** : En traitement de la parole, combiner la lecture labiale (analyse des mouvements des lèvres, entité visuelle) et le signal acoustique (entité auditive) améliore considérablement la reconnaissance par rapport à l'utilisation du signal audio seul ou de l'image seule, surtout en environnement bruyant.

Dans le **Deep Synergy Learning**, on cherche précisément à **favoriser** la création de synergies positives et à **réduire** (voire éliminer) les liens qui relèvent de la simple redondance ou d'interactions stériles. Les règles de mise à jour des **pondérations synergiques** (voir plus loin dans les chapitres dédiés) sont conçues pour :

- **Renforcer** les liens entre entités ayant un $\Delta_{ij} > 0$.
- **Diminuer** ou rompre les liens entre entités dont la corrélation ne procure aucun gain ou, pire, engendre un effet négatif ($\Delta_{ij} < 0$).
- Permettre la **détection** de synergies n-aires, où plusieurs entités coopèrent pour former des micro-réseaux auto-organisés.

Cette démarche permet d'éviter l'explosion combinatoire (en évaluant toutes les combinaisons) grâce à un **mécanisme dynamique** où les liens se forment ou se défont au fil de l'apprentissage, suivant les feedbacks de performance ou des indicateurs entropiques.

Dans le DSL, c'est précisément cette notion de synergie, mesurée et mise à jour en continu, qui permet de construire des **clusters** d'entités coopératives, de faire émerger de **nouvelles représentations**, et de potentialiser la **résilience** du système face à la variabilité des données. Les sections ultérieures reviendront sur la façon dont ce mécanisme de synergie se met en place dans une **approche auto-organisée** (1.2.4) et comment il se distingue des **réseaux neuronaux** traditionnels (1.2.5).

1.2.4. Approche Hiérarchique vs Approche Auto-Organisée

Les approches hiérarchiques traditionnelles, largement utilisées dans le Deep Learning, reposent sur la succession de **couches** (layers) qui transforment progressivement les données d'entrée jusqu'à aboutir à une sortie (une prédiction, une classification, etc.). À chaque couche, on opère une **composition** de fonctions (le plus souvent linéaires, suivies de non-linéarités comme ReLU ou sigmoid). Par opposition, l'**approche auto-organisée** (telle qu'on la retrouve dans le Deep Synergy Learning, DSL) met l'accent sur la **capacité du réseau à reconfigurer ou réorganiser** sa structure interne en fonction des synergies détectées, plutôt que de s'en tenir à une architecture rigide et prédefinie.

Dans cette section, nous allons approfondir cette différence en examinant les principes fondamentaux de l'approche hiérarchique, pour mieux comprendre comment le DSL, en tant qu'approche auto-organisée, propose une alternative à la fois plus **dynamique** et plus **adaptative**.

Fondements de l'approche hiérarchique

Dans un **réseau hiérarchique** traditionnel (tel qu'un réseau de neurones profond), le **principe du traitement en cascade** impose aux données \mathbf{x} de traverser une succession de transformations linéaires et non linéaires :

$$\mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}), \quad \mathbf{h}^{(2)} = f^{(2)}(\mathbf{h}^{(1)}), \quad \dots, \quad \mathbf{h}^{(L)} = f^{(L)}(\mathbf{h}^{(L-1)}),$$

où $\mathbf{h}^{(\ell)}$ représente la “représentation cachée” extraite à la couche ℓ , et $f^{(\ell)}$ désigne un opérateur paramétrique (incluant poids et fonction d’activation). Dans ce schéma, l’information **circule** essentiellement **de bas en haut**, et les éventuelles boucles de rétroaction (feedback top-down) demeurent limitées ou spécialisées, comme dans les architectures RNN ou LSTM.

Cette organisation induit une **séparation des rôles** :

- Les **premières couches** traitent des descripteurs “bas niveau” (par exemple, repérer des bords pour une image, des phonèmes pour un signal audio, etc.).
- Les **couches intermédiaires** approfondissent la combinaison de ces descripteurs, extrayant des motifs plus complexes.
- Les **dernières couches** produisent la **décision finale** (classe, score, etc.).

Toutefois, cette hiérarchie s’accompagne d’une **certaine rigidité**. Une fois le nombre de couches, la taille de chacune, et la nature des connexions (dense, convolutionnelle, récurrente...) choisis, la **structure** du réseau reste figée pendant la phase d’apprentissage. Seuls les **poids** sont ajustés par descente de gradient ou par l’une de ses variantes, tandis que la topologie globale demeure invariable.

Les **réseaux hiérarchiques** classiques présentent plusieurs limites marquantes. D’abord, ils demeurent fortement **dépendants** à la supervision, ils requièrent souvent de larges quantités de données annotées pour “régler” leurs poids internes. Par ailleurs, au fur et à mesure que l’architecture croît, on assiste à une **prolifération** exponentielle du nombre de paramètres, alourdisant le coût en calcul et en mémoire, tout en rendant le réseau plus difficile à **interpréter** et à **déboguer**.

En outre, ces réseaux manifestent un **manque d’adaptabilité**. Face à un “domain shift” (changement de distribution des données), il s’avère nécessaire de procéder à un réapprentissage (ou un ajustement considérable), faute de mécanismes internes pour **reconfigurer** ou **réorganiser** dynamiquement la structure. Enfin, ils souffrent d’une **faible modularité**, bien que les couches s’empilent, elles ne peuvent guère “échanger” librement en dehors des cheminements établis dans l’architecture.

Dans une **approche auto-organisée**, caractéristique du DSL, la **plasticité topologique** occupe une place centrale. Plutôt que de fonctionner avec des couches fixes, les **entités d’information** peuvent spontanément **former des clusters, créer ou rompre** des liaisons, et même **fusionner** si la synergie s’avère suffisamment élevée (cf. sections 1.2.2 et 1.2.3). Ainsi, l’**architecture** ne se trouve plus **imposée**, mais se **construit** et se **reconstruct** au fil de l’**apprentissage**, en fonction des **besoins** et de l’**évolution** des **données**.

Cette **évolution dynamique** repose sur l’adaptation continue des pondérations synergiques $\omega_{i,j}$ suivant une règle telle que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η est un taux d’apprentissage, τ un coefficient de régularisation et $S(\mathcal{E}_i, \mathcal{E}_j)$ la synergie entre les entités \mathcal{E}_i et \mathcal{E}_j . Dans ce cadre, une synergie **positive** renforce la connexion, tandis qu’une synergie **négative** l’affaiblit. Ainsi, les entités dont la coopération est bénéfique sont encouragées à établir (ou consolider) leurs liens, tandis que les connexions moins productives s’éteignent naturellement.

Au fil de ce processus d'ajustement, des **micro-réseaux** ou **clusters auto-organisés** émergent dès lors que les synergies mutuelles s'élèvent entre certaines entités. Ces agrégats peuvent apparaître, se scinder ou disparaître, reflétant les évolutions des données ou l'arrivée d'interactions nouvelles. Par ailleurs, cette liberté structurelle favorise une **coopération multi-flux**, en autorisant les entités visuelles, auditives, textuelles, etc. à s'influencer **directement**, sans passer par un chemin prédéfini de "couches". Par exemple, une entité auditive peut détecter à l'instant t une forte synergie avec une entité visuelle et, de ce fait, former un cluster pour la durée nécessaire ; plus tard, si les conditions changent, elle peut s'éloigner de ce groupe pour établir d'autres coopérations plus pertinentes.

Dans une **architecture auto-organisée**, on ne définit plus un enchaînement linéaire $\mathbf{h}^{(1)} \rightarrow \mathbf{h}^{(2)} \rightarrow \dots$; à la place, on conçoit un **graphe** $G(t)$ dont les **nœuds** sont les entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ et dont les **arêtes** représentent les pondérations synergiques $\omega_{i,j}(t)$. L'évolution du réseau se décrit alors par une fonction de mise à jour :

$$G(t+1) = \mathcal{U}[G(t), S(\cdot, \cdot)],$$

où \mathcal{U} tient compte des **critères de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Le système prend ainsi la forme d'un **Système Dynamique Non Linéaire**, se réorganisant de manière à privilégier les **combinatoires** d'entités jugées les plus utiles ou performantes.

Il est fréquent d'ajouter un mécanisme d'**énergie libre** ou de **coût global** :

$$\mathcal{J}(G) = - \sum_{i,j} \omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j) + \alpha \|\boldsymbol{\omega}\|^2,$$

de façon à **régulariser** la taille du réseau et à éviter que le nombre de connexions ne "flambe" de manière excessive. Dans ce cadre, la mise à jour peut s'effectuer via une **descente de gradient** (ou un algorithme d'optimisation inspiré des systèmes complexes, comme un algorithme génétique ou un recuit simulé), conduisant progressivement à une **organisation** qui valorise les synergies tout en restreignant les liaisons redondantes.

Dans une **approche auto-organisée**, l'**adaptabilité** représente un atout majeur ; le réseau peut s'ajuster en continu face à l'arrivée de nouvelles données ou au changement d'une distribution, sans qu'il faille repenser entièrement son architecture. En outre, la **multi-modalité** y est gérée de façon native, les entités issues de différentes sources (audio, image, texte, etc.) ont la possibilité de s'**influencer** directement et de **former des clusters** multimodaux, ce qui facilite la **fusion** des divers flux et l'exploitation de leurs synergies. Par ailleurs, cette dynamique ouverte autorise un **potentiel créatif**, l'**émergence** de combinaisons inédites entre entités peut révéler des **patrons** jusque-là invisibles, que des architectures hiérarchiques classiques ne parviendraient pas à capturer aussi spontanément.

Néanmoins, cet avantage s'accompagne de plusieurs **défis** importants. D'abord, la **complexité de contrôle** peut s'avérer **élévée**. Sans **mécanismes de régulation** tels que le **facteur $\$\\tauau\$$** ou certaines **pénalités**, le réseau risque de basculer vers un **excès de connexions** ou de **boucles** pouvant engendrer des **oscillations**. Le **coût de calcul** constitue également un défi, car évaluer la **synergie** entre un grand **nombre d'entités** requiert souvent des **heuristiques** ou des **méthodes** parcimonieuses afin de rester **tractable** en **pratique**. Enfin, l'**interprétabilité** peut devenir une **problématique** importante. Bien que l'**auto-organisation** tende à faire émerger des **clusters** plus **significatifs**, l'**évolution** permanente de la **structure** rend plus **complexe** une **analyse en profondeur** du fonctionnement **interne** du réseau.

Comparaison synthétique

Caractéristique	Approche Hiérarchique	Approche Auto-Organisée (DSL)
Architecture	Fixe (définie a priori)	Flexible (graphe évolutif)
Propagation de l'info	Principalement feed-forward	Libre (coopération directe entre entités)
Formation des connexions	Statique (paramètres ajustés)	Dynamique (création / rupture de liens)
Apprentissage	Descente de gradient classique	Mise à jour des synergies (pondérations)
Multimodalité	Fusion tardive (généralement)	Intégration native, clusters multimodaux
Adaptation continue	Limité (fine-tuning, transfert)	Fort (reconfiguration à la volée)
Exemples	CNN, RNN, Transformers	Synergistic Connection Network (SCN)

Les sections suivantes (1.2.5 et suivantes) reviendront sur la comparaison plus directe entre les **réseaux neuronaux traditionnels** et les **réseaux synergiques**, tout en introduisant la terminologie spécifique (1.2.6) et des **exemples** (1.2.7) illustrant la pertinence de l'auto-organisation dans différents contextes naturels.

1.2.5. Réseaux Neuronaux Traditionnels vs Réseaux Synergiques

Dans les sections précédentes, nous avons présenté les notions de **synergie informationnelle**, de **corrélation** et de **plasticité** structurelle dans une approche **auto-organisée**. Il est maintenant temps de faire un **parallèle** entre, d'une part, les **réseaux neuronaux profonds** (ou traditionnels) tels qu'on les connaît en apprentissage profond (Deep Learning) et, d'autre part, les **réseaux synergiques** comme envisagés dans le DSL (Deep Synergy Learning). Cette comparaison aidera à mettre en lumière ce que le DSL apporte de différent par rapport aux architectures classiques (CNN, RNN, Transformers, etc.).

Dans les **réseaux neuronaux traditionnels**, la structure est conçue dès le départ : on détermine à l'avance le nombre de couches et leur type (convolutionnelles, récurrentes, fully-connected, etc.), ainsi que la façon dont elles s'enchaînent. Chaque couche s'appuie sur la représentation produite par la précédente, imposant une **hiérarchie explicite**. L'apprentissage se fait en ajustant les poids et les biais via une descente de gradient ou l'une de ses variantes (Adam, RMSProp, etc.), tandis que la **propagation de l'information** suit principalement un chemin **feed-forward** ; même lorsque des boucles internes existent (RNN, LSTM), elles demeurent cantonnées à la topologie imposée.

À l'inverse, dans les **réseaux synergiques** au sein du **DSL**, la **topologie** se veut **flexible** et **évolutive**. Il ne s'agit plus de **couches** définies de manière **stricte**, mais d'un **ensemble** d'**entités d'information** reliées entre elles par des **pondérations synergiques**. Ces **liens** ne sont pas figés et peuvent évoluer en fonction de la **dynamique** du réseau, comme évoqué dans les sections **1.2.2** et **1.2.4**.

Ces pondérations ne sont pas seulement ajustées ; elles peuvent aussi être créées, renforcées ou rompues, selon la synergie détectée. Cette **possibilité de reconfiguration** permanente marque une **différence** fondamentale. Le **réseau** ne se limite pas à une **simple superposition** de couches fixes, mais il s'**auto-organise** en **clusters** dès lors que la **synergie** entre entités le justifie. De plus, l'information ne circule pas selon une progression linéaire ; elle peut **transiter** entre toutes les entités jugées "synergiques", adoptant ainsi une **approche distribuée** plus proche d'un écosystème vivant que d'un pipeline hiérarchisé.

Dans les **objectifs classiques** du Deep Learning, on minimise une **fonction de coût** $\mathcal{L}(\theta)$ (par exemple, l'entropie croisée ou la MSE), à l'aide d'une **backpropagation** qui calcule les gradients. Les performances sont ensuite mesurées selon des métriques comme la **précision**, le **rappel** ou le **F1-score**, en fonction du type de tâche (classification, régression, etc.).

En revanche, dans une **approche DSL**, ces **objectifs traditionnels** (par exemple, la précision en classification) coexistent avec des **fonctions de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ qui orientent l'apprentissage. Les mises à jour des pondérations synergiques $\omega_{i,j}$ tiennent compte de ces scores de synergie, **favorisant** les liaisons dont la synergie s'avère significative, tout en permettant la **création**, le **renforcement** ou la **dissolution** de connexions. On peut également définir une **fonction globale** $J(G)$, laquelle agrège la **somme** (ou autre forme d'agrégation) des synergies et **pénalise** la multiplication de connexions redondantes. Le réseau se comporte alors comme un **système dynamique**, visant à concilier la **minimisation** d'une perte liée aux tâches classiques (p. ex. une fonction de classification) et la **maximisation** de la synergie informationnelle au sens large.

Dans les **réseaux neuronaux traditionnels**, la **multimodalité** est prise en charge en définissant à l'avance des **voies de traitement distinctes** pour chaque type de donnée, qu'il s'agisse d'**audio**, d'**image** ou de **texte**. Ces flux séparés sont ensuite **fusionnés** à un certain **stade** du traitement, soit à travers des **couches intermédiaires**, soit par une **fusion tardive**, généralement réalisée vers la **fin du pipeline**. L'architecture doit donc être **explicitement conçue** pour chaque canal (par exemple, un CNN dédié à l'image, un RNN ou un Transformer pour le texte, puis un module spécialisé pour agréger les différents flux). La synergie potentielle entre ces canaux se découvre **indirectement**, via la backpropagation, mais la structure globale du réseau — et la manière dont les canaux se croisent — reste imposée de l'extérieur.

À l'inverse, dans les **réseaux synergiques** du DSL, les **entités** associées à divers canaux (audio, visuel, textuel, etc.) ont la possibilité de se “découvrir” **mutuellement** au fil de l'apprentissage. Si l'audio et l'image présentent une forte synergie, elles peuvent **former un cluster multimodal** de manière autonome, sans qu'une couche de fusion spécifique ne soit paramétrée au préalable. Cette approche favorise la **co-évolution** des **représentations** : lorsqu'un flux, tel que le **visuel**, est perturbé par du bruit, l'entité correspondante peut alors s'appuyer davantage sur les **canaux texte ou audio**, à condition qu'une **synergie élevée** soit détectée entre ces modalités. Ainsi, la multimodalité se développe de façon **organique**, guidée par la recherche de gains effectifs de performance ou d'information.

Dans les **réseaux neuronaux profonds** classiques, la **rétrorépropagation** constitue le mécanisme standard pour ajuster les **poids** de couche en couche ; on dispose alors d'une **architecture** explicitement définie de bout en bout et d'un **objectif** scalaire unique qui oriente la descente de gradient (p. ex. l'entropie croisée). En revanche, dans un **DSL**, les **mises à jour** des pondérations synergiques se réalisent de manière **distribuée**, souvent selon des règles plus locales (inspirées, par exemple, d'approches “Hebbiennes généralisées” ou d'une évaluation directe des gains de performance obtenus). Un **objectif global** peut persister (comme un taux de reconnaissance), mais la **découverte** de synergies s'opère fréquemment dans des configurations plus indépendantes du gradient global.

La **rétrorépropagation** n'est pas pour autant exclue. Il est possible d'envisager un **système hybride** où la **backpropagation** s'applique à certains **sous-modules**, tandis que la **formation** et la **reconfiguration** du graphe **synergique** suivent des lois d'**auto-organisation** distinctes. Ainsi, on bénéficie d'une **flexibilité accrue**. Il est possible de continuer à **ajuster** finement certaines parties du réseau via le **gradient**, tout en permettant au système de **découvrir** et de **renforcer** localement des **liaisons synergiques** au-delà du cadre strict imposé par une **architecture fixe**.

Dans les **réseaux neuronaux traditionnels**, la **robustesse** dépend essentiellement de la qualité du jeu d'entraînement et de plusieurs mécanismes de **régularisation** (dropout, batch normalization, etc.). L'**adaptation** à un nouveau domaine s'effectue souvent par un transfert d'apprentissage (transfer learning), suivi d'un **fine-tuning** partiel ou complet des poids du réseau. Cependant, ce procédé peut exposer le système au risque de **catastrophic forgetting** lorsqu'on l'emploie pour apprendre de façon continue une succession de tâches. Les poids ajustés pour les nouvelles données tendent alors à **effacer** ce qui avait été **acquis** auparavant.

Dans les **réseaux synergiques**, au contraire, la **structure évolutive** permet au système d'**allouer** de nouvelles entités ou de **renforcer** certains liens pour absorber plus facilement un **changement** de données, sans exiger un réapprentissage complet de l'ensemble du réseau. Les **clusters** déjà formés pour des tâches précédentes peuvent coexister dans la nouvelle configuration, au lieu d'être remplacés ou écrasés. Ainsi, un réseau synergique peut mieux **retenir** l'expérience passée (réduisant d'autant le **catastrophic forgetting**) et faire preuve d'une plus grande **flexibilité** quand son environnement ou sa mission évoluent.

Les **réseaux synergiques** n'ont pas vocation à **remplacer** purement et simplement les architectures neuronales traditionnelles. Au contraire, divers **scénarios de cohabitation** sont envisageables. On peut, par exemple, adopter une **approche hybride**, dans laquelle un pipeline CNN (pour l'image) ou Transformer (pour le texte) extrait des **représentations** initiales ; ces représentations deviennent ensuite des **entités** au sein d'un réseau synergique, lequel peut alors coopérer et se reconfigurer de manière plus libre.

Dans certains **systèmes complexes**, on peut aussi instaurer une **transition progressive**, en commençant par des couches de feature extraction classiques, puis en insérant une **couche synergique** à un stade où les divers canaux se croisent. De cette façon, on préserve la puissance des modèles traditionnels pour l'extraction de caractéristiques tout en intégrant la logique auto-organisée et adaptative du DSL à un niveau plus élevé.

Enfin, il est possible de développer des **extensions spécialisées**, par exemple un composant auto-organisé dédié à la **fusion multimodale** ou à la **gestion de multiples contextes**, tandis que la classification finale demeure assurée par un réseau fully-connected ordinaire. L'essentiel est d'exploiter la **flexibilité** des réseaux synergiques dans les domaines où ils excellent — par exemple, l'émergence dynamique de clusters — tout en s'appuyant sur l'expérience accumulée des architectures neuronales traditionnelles.

Synthèse et perspectives

Aspect	Réseaux Neuronaux Traditionnels	Réseaux Synergiques (DSL)
Topologie	Fixe, pré-spécifiée	Évolutive, auto-organisée
Propagation	Hiérarchique, feed-forward	Dispersée, multidirectionnelle
Apprentissage	Backpropagation end-to-end	Règles locales + mise à jour synergie
Évolution temporelle	Nécessite du re-training pour s'adapter	Adaptation dynamique à la volée
Gestion multimodale	Fusion tardive ou intermédiaire, souvent manuelle	Fusion spontanée via synergie et création de clusters
Robustesse	Vulnérabilité à l'overfitting, besoin de régulariser	Auto-régulation via le feedback de synergie
Applications	Classification, régression, vision, NLP...	Idem, mais avec en plus la souplesse et l'auto-organisation

En conclusion, les **réseaux neuronaux traditionnels** et les **réseaux synergiques** diffèrent principalement par la **structure**, la **dynamique d'apprentissage** et la **capacité d'auto-organisation**. Le **Deep Synergy Learning** apporte une philosophie plus **organique**, inspirée des systèmes complexes, pour que l'intelligence artificielle puisse gérer la **variabilité**, la **multimodalité**, et s'auto-adapter en continu.

La section suivante (1.2.6) clarifiera la **terminologie** propre au DSL — notamment les notions de **clusters**, **entités**, **pondérations synergiques**, etc. — puis nous verrons (1.2.7) des **exemples illustratifs** tirés de la nature ou d'applications concrètes, afin de matérialiser les principes évoqués dans ce chapitre.

1.2.6. Terminologies Employées dans le DSL

Au fil des sections précédentes, plusieurs notions-clés sont apparues pour décrire les principes du **Deep Synergy Learning (DSL)**. Il est important de les clarifier et de les organiser en un vocabulaire cohérent, car ces termes forment la **boîte à outils conceptuelle** indispensable pour aborder les mécanismes internes et les applications pratiques du DSL. Dans cette section, nous passons en revue les principaux termes et leur signification, en soulignant les liens entre eux.

Entité d'Information (ou “Information Entity”) : Dans le DSL, une **entité d'information** (souvent notée \mathcal{E}_i) représente l'unité fondamentale du système. Contrairement à un simple vecteur de données, une entité est un *objet d'apprentissage* pouvant inclure :

- Une **représentation** (par ex. un vecteur, un tenseur, ou même une distribution).
- Des **paramètres internes** (θ_i) et un **état** ($s_i(t)$).
- Un **historique** (ou “mémoire”) de ses interactions passées.

C'est à travers ces entités que s'établissent les **synergies** et que se construit la dynamique de l'apprentissage. En pratique, toute source d'information (une image, un signal audio, un embedding textuel, etc.) peut être encapsulée sous forme d'entité.

Synergie (ou “Synergy”) : La **synergie** entre deux (ou plusieurs) entités est la mesure de la **valeur ajoutée** qu'elles obtiennent en coopérant, par rapport à ce qu'elles pourraient réaliser indépendamment (voir 1.2.2). Elle se note souvent $S(\mathcal{E}_i, \mathcal{E}_j)$ pour les paires, et peut être généralisée à des ensembles $\{\mathcal{E}_{k_1}, \dots, \mathcal{E}_{k_m}\}$.

Formes de mesure.

- **Informationnelle** : Basée sur l'entropie, l'information mutuelle, ou d'autres métriques de la théorie de l'information.
- **Basée sur la performance** : Différence de score (classification, regression, etc.) quand on associe \mathcal{E}_i et \mathcal{E}_j .
- **Hybride** : Combinaison d'un critère d'information et d'un critère de performance.

La synergie est la “**force motrice**” du DSL. Elle guide la création, la rupture ou le renforcement des connexions entre entités (voir ci-dessous “pondérations synergiques”).

Pondérations Synergiques (ou “Synergistic Weights”) : Les **Pondérations Synergiques** notées $\omega_{i,j}(t)$, ce sont les **coefficients** qui caractérisent la relation dynamique entre deux entités \mathcal{E}_i et \mathcal{E}_j à l'instant t .

Souvent modélisée par une équation du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η est un taux d'apprentissage, τ un terme de régularisation, et $S(\mathcal{E}_i, \mathcal{E}_j)$ la synergie entre les entités.

Les pondérations synergiques constituent la **matrice d'adjacence** d'un *graphe évolutif*:

$$W(t) = [\omega_{i,j}(t)]_{i,j}.$$

Elles déterminent quelles entités sont fortement liées (hautes synergies) et lesquelles le sont moins voire pas du tout (synergie quasi nulle).

Cluster (ou “Micro-Réseau”) : Un **cluster** est un **sous-ensemble** d'entités qui présentent entre elles une synergie élevée, formant ainsi une structure cohérente et **auto-organisée**.

Les entités $\{\mathcal{E}_1, \dots, \mathcal{E}_k\}$ tendent à se regrouper si leurs **pondérations synergiques** mutuelles sont supérieures à un certain seuil θ , ou si elles maximisent un critère global (p. ex. somme des synergies internes au cluster).

Les **clusters synergiques** formés au sein d'un **DSL** jouent un **rôle** essentiel à deux niveaux. D'abord, ils **favorisent la coopération locale**, en permettant aux entités d'un même cluster d'échanger de manière intensive ; chaque entité contribue ainsi ses données ou compétences spécifiques, renforçant la synergie collective.

Ensuite, ils **facilitent l'adaptation** du réseau : ces clusters peuvent en effet fusionner pour gérer de nouveaux contextes si leur compatibilité s'avère élevée, ou se scinder lorsqu'un manque de synergie interne se manifeste.

Grâce à ce double mécanisme — coopération accrue et flexibilité structurelle —, le système demeure résilient et à même d'évoluer face aux changements de tâches ou d'environnements.

Synergistic Connection Network (SCN) : Le **SCN** représente l'**infrastructure** du DSL. C'est un **réseau** dont les *nœuds* sont les entités $\{\mathcal{E}_i\}$ et dont les *arêtes* sont les pondérations $\{\omega_{i,j}\}$.

Contrairement à un réseau de neurones statique, le SCN est **dynamique** : au fil du temps (ou au fil des itérations d'apprentissage), de nouvelles connexions apparaissent, d'autres se suppriment ou s'affaiblissent, et des clusters émergent.

L'**objectif** central du SCN consiste à **exploiter** les **synergies** entre entités de manière à *auto-organiser* le flot d'information et, ce faisant, à optimiser la **performance** globale du système, qu'il s'agisse d'une tâche supervisée ou non supervisée. L'idée est de permettre aux liens synergiques les plus pertinents de se renforcer, afin que le réseau dirige spontanément les informations vers les chemins les plus efficaces. Ainsi, l'architecture se réage en fonction

des besoins (ou des données) pour offrir un apprentissage et un traitement des informations plus rapide et plus robuste, sans nécessiter de contrôle externe permanent.

Auto-Organisation : **Auto-organisation** désigne la **capacité** d'un réseau à *se structurer* et *se reconfigurer* de façon autonome, sans intervention ou contrôle direct de l'extérieur (cf. section 1.2.4). Ce phénomène repose sur une **évaluation continue** de la synergie entre l'ensemble (ou une partie) des entités : à chaque itération, les pondérations $\omega_{i,j}$ sont **mises à jour** selon une règle d'adaptation, et des **clusters** peuvent se **former** ou se **dissoudre** en fonction des tendances observées.

L'**objectif** de ce mécanisme est triple. D'abord, il s'agit d'acquérir une **robustesse** accrue face aux perturbations, car le réseau peut se réorganiser spontanément lorsque des défaillances ou des changements surviennent. Ensuite, cette approche permet de gérer naturellement la **multimodalité** : au lieu de cloisonner les entités (visuelles, auditives, etc.), on les laisse s'associer ou se séparer au gré de leurs synergies. Enfin, l'auto-organisation ouvre la voie à un **apprentissage continu**, dans lequel de nouvelles représentations émergentes se forment au fil du temps, sans imposer la rigidité d'un schéma hiérarchique figé.

État (ou “State”) d'une Entité : Chaque entité \mathcal{E}_i dispose d'un **état interne** $\mathbf{s}_i(t)$, souvent représenté par un vecteur de dimension d , qui synthétise son “histoire” ou son “contexte” à l'instant t . Cet état évolue selon une **fonction d'actualisation** F , de la forme

$$\mathbf{s}_i(t+1) = F(\mathbf{s}_i(t), \{\omega_{i,j}(t)\}_j, \dots),$$

inspirée, par exemple, de modèles dynamiques ou de mécanismes de type RNN ou “Hebb étendu”. Le **rôle** de $\mathbf{s}_i(t)$ est déterminant pour la réactivité de l'entité : une entité ayant déjà établi de fortes coopérations avec une autre \mathcal{E}_j est généralement plus encline à **se synchroniser** de nouveau avec elle, la mémoire de ses interactions passées renforçant la probabilité d'une synergie future.

Mécanismes de Fusion et de Dissociation : Deux (ou plusieurs) entités $\{\mathcal{E}_i, \mathcal{E}_j, \dots\}$ peuvent **fusionner** s'il s'avère qu'elles sont presque systématiquement dans un même cluster et qu'elles partagent une forte synergie dans la durée. Cette fusion se modélise par la création d'une **nouvelle entité** $\mathcal{E}_{\text{fusion}}$, qui combine leurs états, leurs mémoires et leurs représentations.

Lorsqu'une entité \mathcal{E}_k se trouve dans un cluster peu cohérent (synergie moyenne ou négative), elle peut se **retirer** du cluster ou rompre une fusion antérieure.

Ces mécanismes confèrent au DSL une **plasticité structurale** comparable à celle de certains systèmes biologiques (cerveau, colonies d'insectes, etc.), favorisant l'adaptation face à de nouveaux contextes ou de nouvelles tâches.

Énergie ou Fonction Globale \mathcal{J} : On peut parfois définir une **fonction** $\mathcal{J}(G)$ — parfois appelée “énergie libre” ou “coût global” — qui regroupe, d'une part, les **synergies** positives entre les entités et, d'autre part, un **terme** de pénalisation destiné à éviter une **surabondance** de connexions. Par exemple :

$$\mathcal{J}(G) = - \sum_{i,j} \omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j) + \alpha \|\boldsymbol{\omega}\|^2,$$

où $\boldsymbol{\omega}$ désigne le vecteur de toutes les pondérations synergiques, et α un coefficient de régulation. **Minimiser** $\mathcal{J}(G)$ revient alors à **maximiser** la somme de synergies utiles tout en **limitant** la prolifération de liens non pertinents. Il s'agit ainsi d'une démarche **globale** pour piloter l'auto-organisation du réseau, puisqu'elle encourage les connexions réellement productives tout en imposant un frein à celles qui n'apporteraient aucun gain substantiel.

Apprentissage Continu (ou “Lifelong Learning”) : Dans le **DSL**, l'apprentissage ne se limite pas à une **phase offline** unique ; le réseau peut, au contraire, **évoluer** continuellement face à un flux de données **online**, en réajustant de façon permanente les pondérations $\omega_{i,j}(t)$ ainsi que la configuration des clusters.

Cet **apprentissage continu** présente plusieurs **avantages** : d'une part, il offre une **tolérance** accrue aux perturbations (bruit, changements dans la distribution des données, apparition de nouvelles classes ou contextes), ce qui lui permet de s'adapter plus aisément à des environnements non stationnaires. D'autre part, il contribue à la **réduction** du phénomène de “forgotten knowledge”, puisque les clusters formés pour des tâches antérieures peuvent être préservés et ainsi servir de base à des transferts de connaissances ultérieurs.

Terminologies récurrentes (Synthèse)

Pour faciliter la lecture et l'implémentation, voici un **récapitulatif** des principales terminologies :

1. \mathcal{E}_i : *Entité d'information* numéro i .
2. $S(\mathcal{E}_i, \mathcal{E}_j)$: *Synergie* entre entités i et j .
3. $\omega_{i,j}(t)$: *Pondération synergique* (ou lien) entre entités i et j à l'instant t .
4. $W(t) = [\omega_{i,j}(t)]$: *Matrice* (ou graphe) des pondérations synergiques.
5. **Cluster** : Sous-groupe d'entités fortement liées (hautes $\omega_{i,j}$).
6. **SCN** : *Synergistic Connection Network*, la structure dynamique qui évolue selon les lois d'adaptation.
7. **Auto-organisation** : Processus par lequel la structure $W(t)$ se réarrange spontanément.
8. **Fonction \mathcal{J}** : Mesure globale de la qualité ou de l'état du réseau (peut inclure la somme des synergies, des pénalités, etc.).

Conclusion

Ces termes — **entité**, **synergie**, **pondération synergique**, **cluster**, **SCN**, **auto-organisation**, etc. — forment le *lexique de base* du DSL. Chaque concept y est interdépendant : les **entités** interagissent via des **pondérations synergiques** qui façonnent le **SCN**, lequel se **réorganise** en **clusters** au gré d'un mécanisme d'**auto-organisation** orienté par la **synergie** et, éventuellement, par une **fonction globale \mathcal{J}** .

Dans la section suivante (1.2.7), nous illustrerons ces principes par des **exemples concrets**, qu'ils proviennent de la nature (inspirations biologiques) ou d'applications pratiques (cas d'études multimodales, émergence de schémas cognitifs, etc.). Ce sera l'occasion de vérifier comment l'utilisation rigoureuse de cette terminologie peut clarifier la **logique** et la **mise en œuvre** du Deep Synergy Learning.

1.2.7. Exemples Illustratifs de la Synergie dans la Nature

Les principes de **synergie informationnelle** et d'**auto-organisation** que promeut le Deep Synergy Learning (DSL) trouvent de nombreux échos dans les systèmes naturels. Qu'il s'agisse de colonies d'insectes, de réseaux neuronaux biologiques, d'écosystèmes ou de synchronisations collectives, on observe des processus où l'**ensemble** dépasse la **somme de ses parties**, grâce à des mécanismes coopératifs distribués. Les sous-sections suivantes illustrent comment ces phénomènes naturels inspirent l'approche synergique du DSL.

Colonies d'Insectes et Intelligence Collective

Les colonies de fourmis, d'abeilles ou de termites constituent des exemples emblématiques d'**intelligence collective**, car chaque individu, aux capacités limitées, contribue à la réalisation de tâches pourtant très élaborées : construction de nids sophistiqués, optimisation de la recherche de nourriture, etc. Ces interactions reposent sur des **signaux locaux** (phéromones, contacts antennaires, etc.), sans qu'aucune entité centrale ne dirige l'ensemble. L'émergence d'une organisation globale, comme le traçage de pistes ou la réparation du nid, résulte donc d'une **coopération** distribuée entre entités locales.

Dans le **Deep Synergy Learning (DSL)**, les **entités d'information** jouent un rôle comparable à celui de ces insectes : elles établissent ou rompent des liens en fonction de la **pertinence** (ou synergie) qu'elles y perçoivent. Ces **connexions synergiques** évoluent sans cesse, à l'image des fourmis qui renforcent ou abandonnent certains chemins selon leur utilité.

La formation de "clusters" d'entités synergiques dans le DSL évoque les **micro-sociétés** existant au sein d'une colonie d'insectes, où chaque groupe se spécialise dans une tâche particulière. Cette **auto-organisation** spontanée illustre la force d'un système distribué : sans planification rigide, l'ensemble se coordonne pour atteindre un objectif global.

Synergies dans le Cerveau et les Réseaux Neuronaux Biologiques

Dans le **cerveau** humain ou animal, la **plasticité synaptique** illustre la puissance d'un réseau extrêmement **connecté**, dans lequel les synapses s'ajustent en fonction des interactions locales. Lorsque deux neurones s'associent régulièrement pour traiter un même stimulus, leur **synapse** se renforce (potentialisation à long terme) ; ce phénomène rappelle la **mise à jour** des liaisons synergiques dans le DSL, où les liens forts se consolident à mesure que les entités coopèrent efficacement.

Les neurosciences démontrent également la formation d'**assemblées neuronales** associées à un concept ou à un stimulus précis. Ces assemblées se créent ou se dissolvent selon le **contexte** ou la **tâche** du moment. De la même manière, le DSL autorise la **création** et la **dissolution** de **clusters** d'entités d'information, la synergie entre ces entités évoluant dans le temps pour s'adapter aux besoins et aux données.

Enfin, ce parallèle s'étend à la **mémoire** et à l'**apprentissage** : dans le cerveau, les synapses permettent à un neurone de "se souvenir" des connexions consolidées lors d'apprentissages antérieurs. Dans le DSL, chaque entité conserve un **état interne** et un **historique** (section 1.2.1), ce qui lui confère une **mémoire contextuelle** et améliore la **cohérence** de l'apprentissage sur le long terme.

Écosystèmes et Coopérations Symbiotiques

Les **écosystèmes** offrent de nombreux exemples de **coopération** interspécifique : insectes pollinisant les plantes, lichens nés de la symbiose entre algues et champignons, ou encore la mycorhize qui associe champignons et racines de plantes. Dans chacun de ces cas, les organismes trouvent un **bénéfice mutuel**, qu'il s'agisse d'un accès accru aux ressources, d'une protection renforcée ou d'une capacité d'adaptation élargie. C'est la **notion de "gain commun"** qui se déploie ici, illustrant la **synergie** : la coexistence de deux entités (ou espèces) génère une **valeur ajoutée** que l'on ne retrouverait pas si elles agissaient isolément.

Dans le **Deep Synergy Learning (DSL)**, ce principe s'incarne à travers la **mesure de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (voir section 1.2.2) et l'**ajustement** adapté des pondérations $\omega_{i,j}(t)$. De la même façon que deux espèces coopérantes se renforcent l'une l'autre, deux entités informationnelles voient leurs liens se consolider lorsqu'elles interagissent efficacement. Les **écosystèmes** diversifiés, riches en symbioses, font preuve d'une **résilience** considérable face aux menaces comme la sécheresse ou la préation, grâce aux ressources complémentaires que chaque espèce apporte. Un **réseau synergique** (DSL) réunissant des entités variées (visuelles, textuelles, auditives, etc.) gagne, lui aussi, en **robustesse** et en **flexibilité** : en modulant continuellement ses interactions, il peut mieux réagir aux imprévus ou aux évolutions de l'environnement.

Synchronisation Collective : Bancs de Poissons et Nuées d'Étourneaux

Les bancs de poissons et les vols d'oiseaux, tels que les nuées d'étourneaux, illustrent un phénomène de synchronisation remarquable : de larges groupes se meuvent de façon presque chorégraphiée, sans chef unique. Chaque individu ajuste sa trajectoire en fonction de celle de ses voisins, engendrant ainsi un **effet émergent** de cohésion et d'harmonisation.

Ce type d'organisation se comprend généralement à travers quelques **règles simples** (alignement de la vitesse, distance de sécurité, attraction) qui, une fois agrégées, aboutissent à des comportements collectifs complexes. Dans le cadre du Deep Synergy Learning (DSL), la **synergie** et la **mise à jour** des liens jouent un rôle équivalent : lorsque la coopération entre deux entités se révèle bénéfique, celles-ci se synchronisent, et le **réseau global** s'en trouve optimisé.

Les bancs ou les nuées font également preuve d'une grande **plasticité**, se reconfigurant rapidement face à un prédateur ou un obstacle. De façon parallèle, un **réseau synergique** peut, à tout instant, **adapter** sa structure dès lors que le contexte ou les données évoluent, sans nécessiter de "réentraînement" global et figé.

Conclusion

Les exemples précédents – colonies d'insectes, cerveau, écosystèmes ou synchronisations collectives – démontrent que la synergie émerge lorsque des entités locales et relativement simples **coopèrent** selon des **règles d'interaction** et **d'adaptation**. Sans supervision centrale et sans plan préconçu, il se forme souvent des **structures** ou des **comportements** remarquablement organisés et robustes, capables de s'ajuster aux contraintes du milieu.

Ces observations, empruntées à la nature, guideront la formulation plus formelle des **algorithmes** et la mise en place de **protocoles d'évaluation** pour le DSL. Dans les chapitres suivants, nous verrons comment concrétiser ces analogies sous la forme de modèles mathématiques, de règles de mise à jour et d'applications pratiques, visant à faire du Deep Synergy Learning un **paradigme opérationnel** pour une **IA forte** (ou du moins plus autonome et plus générale).

Problèmes

Problème 1 : Caractérisation Formelle des Entités d'Information

On considère la section 1.2.1, où une **entité d'information** \mathcal{E}_k est définie comme un objet pouvant inclure une représentation \mathbf{x}_k , un état interne \mathbf{s}_k et un ensemble de paramètres θ_k .

1. Représentation mathématique

Proposez une écriture formelle pour l'entité \mathcal{E}_k , par exemple :

$$\mathcal{E}_k = (\mathbf{x}_k, \mathbf{s}_k, \theta_k),$$

et montrez comment on peut généraliser \mathbf{x}_k à un **espace de Hilbert** (au lieu de \mathbb{R}^d). Quelles propriétés (norme, produit scalaire) sont requises pour permettre des mesures de similarité ?

2. Évolution de l'état interne

Supposons qu'on introduise une fonction de transition :

$$\mathbf{s}_k(t+1) = F(\mathbf{s}_k(t), \theta_k, \mathbf{x}_k, \dots).$$

Décrivez quelles hypothèses de **continuité** ou de **lipschitzianité** (si on veut prouver une existence et unicité de $\mathbf{s}_k(t)$) sont nécessaires.

3. Distances et similarités entité–entité

Pour deux entités \mathcal{E}_i et \mathcal{E}_j , on définit une distance :

$$d(\mathcal{E}_i, \mathcal{E}_j) = \alpha \|\mathbf{x}_i - \mathbf{x}_j\| + \beta \|\mathbf{s}_i - \mathbf{s}_j\| + \gamma D(\theta_i, \theta_j),$$

où D est une distance paramétrique (p. ex. la somme des distances des paramètres).

- Montrez que si $\|\cdot\|$ et $D(\cdot, \cdot)$ sont des distances, alors d en est aussi une, sous conditions ($\alpha, \beta, \gamma > 0$).
- Discutez de l'éventuelle non-linéarité de $D(\theta_i, \theta_j)$ (p. ex. si θ est lui-même un paramètre structural).
- **Entités comme agents actifs**

On suppose qu'une entité peut "agir" en modifiant $\mathbf{x}_k, \mathbf{s}_k, \theta_k$. Proposez un cadre formel où l'"action" $\mathbf{a}_k(t)$ d'une entité influe sur son état futur. Quels problèmes d'analyse mathématique se posent pour étudier la **cohérence globale** de multiples entités interagissantes ?

Problème 2 : Mesures de Synergie Informationnelle

La section 1.2.2 introduit la notion de **synergie** entre deux entités \mathcal{E}_i et \mathcal{E}_j . On peut la définir via l'information mutuelle, la corrélation inversée, ou un gain de performance.

1. Axomes de la Synergie

Proposez quatre propriétés (analogues aux axiomes de l'information mutuelle) qu'une fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ devrait vérifier pour être considérée comme une "vraie" synergie (ex. non-négativité, symétrie, etc.). Discutez de la compatibilité avec une définition de type "gain de performance".

2. Lien avec l'entropie

Si \mathbf{X}_i et \mathbf{X}_j sont les variables aléatoires associées à \mathcal{E}_i , \mathcal{E}_j , on définit :

$$S(\mathbf{X}_i, \mathbf{X}_j) = I(\mathbf{X}_i \wedge \mathbf{X}_j) - R(\mathbf{X}_i, \mathbf{X}_j),$$

où I est l'information mutuelle et R mesure la "redondance". Justifiez pourquoi on peut avoir $S > 0$ même si $I(\mathbf{X}_i, \mathbf{X}_j)$ est faible, dans le cas où la combinaison $\mathbf{X}_i + \mathbf{X}_j$ apporte un bénéfice supérieur à la somme des bénéfices individuels.

3. Maximisation de la synergie

Envisagez un problème d'**optimisation** où l'on cherche à trouver des transformations f_i et f_j (appliquées respectivement aux représentations de \mathcal{E}_i et \mathcal{E}_j) pour **maximiser** $S(f_i(\mathcal{E}_i), f_j(\mathcal{E}_j))$.

- Formulez le problème sous forme d'un **argmax**
- Discutez de l'existence potentielle de "solutions triviales" (par ex. tout mapper vers une même constante).
- **Généralisation à la synergie n-aire**
Exposez une définition possible de $S(\mathcal{E}_1, \dots, \mathcal{E}_n)$. Quelles difficultés conceptuelles (et calculatoires) surgissent si on veut éviter la simple somme des synergies binaires, afin de capter les effets strictement "collectifs" ?
- **Équivalence synergie–corrélation non linéaire**
Montrez un exemple (même purement théorique) où deux entités \mathcal{E}_i , \mathcal{E}_j ont une corrélation linéaire nulle, mais une synergie positive. Quel type de fonction non linéaire relie \mathbf{x}_i et \mathbf{x}_j pour produire ce phénomène ?

Problème 3 : Interaction, Synergie et Corrélation

La section 1.2.3 aborde la **différence** entre ces trois notions : interaction, synergie, corrélation.

1. Relation formelle interaction vs. Synergie

Montrez, via un exemple ou un énoncé mathématique, qu'une **forte interaction** (modèle non linéaire où la sortie dépend de $\mathbf{x}_i \times \mathbf{x}_j$) n'implique pas forcément une **synergie positive** au sens "gain commun". On peut utiliser un modèle polynomial pour illustrer cela.

2. Corrélation et redondance

Dans le cas d'un vecteur $\mathbf{X} = (X_1, X_2)$ gaussien, rappelez la formule de la corrélation linéaire ρ . Montrez que si ρ est proche de 1 (corrélation élevée), alors le gain de performance en combinant X_1 et X_2 (pour prédire une variable Y) est faible, relevant d'une **redondance** plutôt que d'une **synergie**.

3. Synergie mesurée par un gain de performance

On définit :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \Phi(\{\mathcal{E}_i, \mathcal{E}_j\}) - [\Phi(\{\mathcal{E}_i\}) + \Phi(\{\mathcal{E}_j\})],$$

où Φ est un score de prédiction (classification, etc.). Élaborez les conditions nécessaires pour que cette définition soit non triviale (par ex. si la somme $\Phi(\{\mathcal{E}_i\}) + \Phi(\{\mathcal{E}_j\})$ n'est pas déjà maximale).

4. Lien avec l'information mutuelle conditionnelle

Proposez une **formulation** reliant l'information mutuelle conditionnelle $I(\mathbf{X}_i, \mathbf{X}_j | \mathbf{Y})$ à l'idée de synergie, où \mathbf{Y} serait la variable cible (label). Dans quelles hypothèses peut-on voir cette conditionnelle comme un indicateur de synergie ?

Problème 4 : Hiérarchie vs. Auto-Organisation

La section 1.2.4 compare l'**approche hiérarchique** (couches fixes) à une **approche auto-organisée** (où les liens se créent/détruisent selon la synergie).

1. Modèle hiérarchique “statique”

Considérez un réseau classique (feed-forward) de L couches. Définissez la fonction de propagation :

$$\mathbf{h}^{(1)} = f^{(1)}(\mathbf{x}), \quad \mathbf{h}^{(2)} = f^{(2)}(\mathbf{h}^{(1)}), \dots, \mathbf{h}^{(L)} = f^{(L)}(\mathbf{h}^{(L-1)}).$$

Quelle est la dimension de l'espace paramétrique ? Pourquoi dit-on que l'architecture n'est pas modifiée par l'apprentissage (seuls les poids le sont) ?

2. Graphe évolutif

Dans une approche auto-organisée, on modélise le réseau par un graphe $G(t)$ à n entités, avec des pondérations $\omega_{i,j}(t)$. Établissez des équations type :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Montrez comment ce système peut dépasser la simple structure de couche “1 vers 2 vers 3...”.

3. Critère de “profondeur”

Dans un réseau hiérarchique, la notion de “profondeur” est définie par le nombre de couches. Comment définir une notion équivalente de “profondeur” ou de “distance” dans un graphe auto-organisé ? (Ex. longueur moyenne des chemins entre entités, diamètre, etc.) Quels problèmes de complexité cela pose-t-il si le graphe évolue en temps réel ?

4. Cas limite : le graphe complet

Que se passerait-il si, dans une approche auto-organisée, toutes les pondérations $\omega_{i,j}(t)$ devenaient fortes ? Discuter d'un point de vue mathématique pourquoi une telle situation est souvent **instable** ou pénalisée (soit par un mécanisme d'énergie libre, soit parce que l'on observerait alors du bruit redondant).

Problème 5 : Réseaux Synergiques et Exemples Naturels

La section 1.2.7 présente des **exemples** de synergie dans la nature (cerveau, colonies d'insectes, écosystèmes...).

1. Equation de type Hebb pour la plasticité synaptique

En neurosciences, la mise à jour “hebbienne” peut être écrite (simplifiée) :

$$w_{ij}(t+1) = w_{ij}(t)\eta [a_i(t) a_j(t)],$$

où $a_i(t)$ est l'activité du neurone i . Montrez en quoi c'est une **approximation** d'une règle de synergie $\omega_{i,j}(t+1) \approx \omega_{i,j}(t) + [S_{i,j}] - \tau \omega_{i,j}(t)$. Quelles hypothèses simplificatrices implique l'absence d'un terme $-\tau w_{ij}$?

2. Modélisation mathématique des colonies d'insectes

Dans la recherche de nourriture, le trajet d'une fourmi peut se modéliser par un **champ de phéromone** ϕ_{ij} sur l'arête (i, j) d'un graphe représentant le terrain. Établissez la similitude avec le DSL :

- $\phi_{ij} \leftrightarrow \omega_{i,j},$
- loi d'évaporation $\phi_{ij}(t+1) = (1 - \delta) \phi_{ij}(t)$ faisant écho au terme $-\tau \omega_{i,j}(t).$
Discuter des contraintes mathématiques pour prouver la convergence vers un chemin optimal.

- **Auto-régulation dans un écosystème**

Imaginez n espèces, chacune ayant une population $x_i(t)$. Les interactions peuvent être modélisées par un système de Lotka–Volterra couplé :

$$\frac{dx_i}{dt} = x_i \left(\alpha_i + \sum_{j \neq i} \beta_{i,j} x_j \right),$$

où $\beta_{i,j}$ peut être positif (synergie) ou négatif (compétition). Montrez en quoi cela illustre la **coévolution** d'entités, et comparez cette dynamique à la formation/dissolution de liens $\omega_{i,j}$ dans le DSL.

- **Synchronisation de bancs de poissons**

Considérez un **modèle de Vicsek** (ou équations similaires) où chaque entité (poisson, oiseau) met à jour sa vitesse en se rapprochant de celle de ses voisins. Comment le phénomène d'“alignement local” peut-il être vu comme un **cas particulier** de synergie, $S(\mathcal{E}_i, \mathcal{E}_j)$? Décrivez la traduction mathématique via un terme $\omega_{i,j} (\mathbf{v}_j - \mathbf{v}_i)$.

- **Conclusion sur la transposition aux “réseaux synergiques”**

Pour chacun des exemples (cerveau, fourmis, écosystème, synchronisation), reformulez succinctement comment on passe d'une description “naturelle” (neuronale, biologique, etc.) à une description “inspirée DSL” (entités d'information, liens synergiques, lois de mise à jour). Quels **défis** (analyse de stabilité, attracteurs multiples, etc.) subsistent pour établir des **résultats formels** sur la convergence ou la robustesse des réseaux synergiques ainsi modélisés ?

1.3. Importance de l'Auto-Organisation

Dans le cadre du **Deep Synergy Learning (DSL)**, l'un des aspects les plus novateurs réside dans sa capacité d'**auto-organisation**. Alors que les approches d'IA traditionnelles s'appuient souvent sur des architectures hiérarchiques fixes, le DSL encourage un processus dynamique où les **entités d'information** établissent et révisent en continu leurs relations, en fonction de la **synergie** qu'elles détectent entre elles (voir sections précédentes). Cette section (1.3) aborde l'importance de l'auto-organisation sous plusieurs angles, depuis les sources d'inspiration biologiques (1.3.1) jusqu'à l'impact sur la résilience des modèles (1.3.6) et les perspectives globales (1.3.7).

1.3.1. Inspirations Biologiques et Cognitives

L'auto-organisation, au sens où l'entend le DSL, n'est pas un concept nouveau. En réalité, de nombreux **systèmes naturels** (organismes vivants, écosystèmes, cerveau, etc.) démontrent d'extraordinaire facultés d'organisation spontanée et adaptative. Le **Deep Synergy Learning** cherche à exploiter ces principes, en transposant aux systèmes informatiques des mécanismes qui ont fait leurs preuves dans le vivant.



Le cerveau comme paradigme de plasticité

Le cerveau humain compte des centaines de milliards de neurones, reliés par des synapses dont les forces (poids synaptiques) évoluent selon l'activité, un phénomène désigné par "plasticité synaptique". On peut symboliser ce principe via une équation de type Hebbien généralisée, par exemple :

$$w_{ij}(t+1) = w_{ij}(t) + \eta [A_{ij}(t) - \lambda w_{ij}(t)],$$

où w_{ij} est la "force" de la synapse reliant le neurone i au neurone j , η un taux d'apprentissage, $A_{ij}(t)$ l'activité conjointe des neurones i et j (ou une mesure de leur association), et λ un terme de régulation. L'important est de voir comment la synapse se renforce si l'activité commune est significative, rappelant la **mise à jour des pondérations synergiques** dans le DSL (cf. section 1.2.6).

Dans le cerveau, des regroupements (ou *assemblées*) de neurones se forment lorsque leurs synapses se renforcent mutuellement. Le concept de cluster auto-organisé dans le DSL (où les entités se regroupent pour exploiter leur synergie) ressemble à cette idée :

$$\mathcal{C}^* = \underset{\mathcal{C} \subset \{\mathcal{E}_1, \dots, \mathcal{E}_n\}}{\operatorname{argmax}} \sum_{(i,j) \in \mathcal{C} \times \mathcal{C}} \omega_{i,j},$$

c'est-à-dire la recherche d'un sous-ensemble \mathcal{C} maximisant la somme des pondérations entre ses membres, analogiquement à des neurones qui s'assemblent pour traiter un même stimulus.



La cognition résulte du fait que certains assemblages persistent dans le temps et renforcent leur "*autoroute synaptique*" (par exemple, mémorisation d'un stimulus répété). Dans le DSL, l'émergence de **micro-réseaux** stables autour d'entités synergiques reflète ce même phénomène d'**apprentissage distribué**, favorisant la récurrence de schémas pertinents.

1.3.1.2. Colonies d'insectes et intelligence collective

Les colonies de fourmis s'illustrent souvent par leur capacité à trouver des chemins optimaux pour la recherche de nourriture. Ce comportement repose sur des **interactions locales** (dépôt de phéromones, etc.), et non sur un plan global. L'analogie avec le DSL se fait sentir lorsque l'on considère que chaque fourmi incarne une "entité" prenant des décisions locales qui, agrégées, produisent une optimisation globale.

On peut modéliser la concentration de phéromones le long d'un chemin par une fonction $\phi(t)$ obéissant à un mécanisme de renforcement/dissipation, par exemple :

$$\frac{d\phi}{dt} = \alpha I(\text{fourmis}) - \beta \phi,$$

où α modélise l'ajout de phéromones (si des fourmis passent) et β la vitesse d'évaporation. Le DSL retrouve un concept similaire quand il gère l'**intensité** d'une liaison synergique via des règles de renforcement ou de décroissance (cf. équations de mise à jour des pondérations, section 1.2.4).



À l'instar de chemins de phéromones qui “naissent” (s'ils sont utiles) et “disparaissent” (s'ils ne sont plus empruntés), les connexions synergiques $\omega_{ij}(t)$ du DSL se renforcent ou s'étendent, entraînant la **création** ou la **rupture** de clusters d'entités d'information. Cette dynamique **auto-organisée** ne requiert pas de commande centrale.

1.3.1.3. Systèmes dynamiques et attracteurs auto-organisés

De nombreux systèmes auto-organisés (réacteurs chimiques de type Belousov-Zhabotinsky, réactions oscillantes, etc.) peuvent s'exprimer par des **équations différentielles** à plusieurs variables, où les interactions locales créent des **boucles de rétroaction** positives ou négatives. Les motifs qui émergent (ondes chimiques, spirales...) sont des exemples d'**attracteurs dynamiques**.

Dans le DSL, on peut considérer que l'**état global** du réseau est donné par la matrice $\Omega(t) = [\omega_{i,j}(t)]$. L'évolution de $\Omega(t)$ suit des règles de type :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Cette équation peut se voir comme un **système dynamique** discret, susceptible de converger vers un attracteur (une configuration stable) ou d'entrer dans des régimes oscillatoires, reflétant l'**auto-organisation**.

Dans un cadre mathématique plus poussé, on peut étudier la **stabilité** de l'état Ω^* en résolvant :

$$\omega_{i,j}(t+1) - \omega_{i,j}(t) = 0,$$

pour tous (i, j) . On obtient alors une condition d'équilibre, ce qui, dans un contexte du DSL, correspond à un réseau où chaque liaison a atteint une valeur stable en regard des synergies (sous réserve que l'on n'introduise pas de nouvelles entités ou de nouvelles données). Une **analyse de la Jacobienne** locale pourrait prédire si cet état Ω^* est un **point d'attraction**, un **cycle limite** ou un **chaos**.

(Voir Appendix 1)

1.3.1.4. Vers un transfert de ces principes au DSL

Les systèmes biologiques et cognitifs décrits ci-dessus mettent en avant quatre qualités fondamentales :

- **Plasticité** (adaptation continue)
- **Robustesse** (résistance au bruit ou à la perturbation)
- **Emergence** (apparition de structures non planifiées)
- **Auto-régulation** (équilibre entre renforcement et inhibition)

Le DSL s'inspire de ces qualités pour proposer un **apprentissage distribué**, dans lequel les entités d'information se comportent comme des agents qui renforcent leurs liens lorsqu'ils détectent un **gain** (cf. notion de synergie) et les relâchent sinon. Matériellement, on transpose ces mécanismes via :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta \left(S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t) \right),$$

et des **clusters** se forment spontanément si le gain est significatif.

Contrairement aux réseaux neuronaux profonds classiques, où seule la valeur des poids évolue dans une architecture figée, l'**auto-organisation** du DSL laisse la structure du réseau se **recomposer** en fonction des synergies détectées. Les entités elles-mêmes peuvent évoluer, en modifiant leur **représentation** ou leur **état** interne, et donc modifier indirectement la dynamique globale du réseau.

1.3.1.5. Conclusion partielle : fondements biologiques de l'auto-organisation

Cette section a illustré, par des références à la **plasticité synaptique**, à l'**intelligence collective** chez les insectes et à divers **systèmes dynamiques**, l'importance des mécanismes d'auto-organisation en biologie et en cognition. Le **Deep Synergy Learning** cherche à exploiter ces mêmes principes pour créer des modèles d'intelligence artificielle moins dépendants d'une structure hiérarchique imposée, plus adaptatifs, et potentiellement plus proches d'une **IA générale**. Les sections suivantes (1.3.2 à 1.3.7) mettront en évidence des concepts plus généraux d'**émergence**, de **feedback**, et détailleront comment l'auto-organisation du DSL se compare aux méthodes d'apprentissage classiques, avant de conclure sur la pertinence de cette approche pour des architectures globales et multimodales.

1.3.2. Concepts Clés : Émergence, Auto-Régulation, Feedback

La dynamique **auto-organisée** que promeut le Deep Synergy Learning (DSL) repose sur plusieurs notions fondamentales, à la fois conceptuelles et mathématiques, qui permettent de comprendre comment un système d'entités peut s'ajuster et se coordonner sans planification centrale. Trois d'entre elles sont particulièrement essentielles : l'**émergence**, l'**auto-régulation** et le **feedback** (rétroaction). Les sous-sections ci-après détaillent la signification de chacun de ces concepts et leur importance dans le cadre du DSL.

Émergence

Le terme **émergence** décrit la situation où un **phénomène global** ou une **structure** à l'échelle macroscopique naît de l'interaction de **composants** à l'échelle microscopique, sans qu'il n'y ait de plan explicite ni de chef d'orchestre. Par exemple, la forme d'une nuée d'oiseaux ou l'organisation complexe d'une ruche peuvent être vus comme des propriétés émergentes d'un grand nombre d'interactions locales.

Dans le Deep Synergy Learning, l'**émergence** se manifeste par la formation de **clusters** (ou micro-réseaux) d'entités d'information, que l'on peut qualifier de "sous-systèmes cohérents" au sein du grand réseau synergique. Ces clusters ne sont pas prédefinis : ils **apparaissent** (et parfois **disparaissent**) en fonction de la synergie détectée entre entités.

Un cluster $\mathcal{C} \subset \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ peut s'illustrer par une somme de pondérations internes $\sum_{(i,j) \in \mathcal{C} \times \mathcal{C}} \omega_{i,j}$ qui se trouve **maximalisée** à un certain moment t .

De plus, ces clusters émergents peuvent générer de nouvelles **représentations** ou entités "conjointes", lesquelles elles-mêmes nourrissent l'**évolution** du réseau.

Sur le plan formel, on peut lier l'émergence au fait que la **dynamique** des pondérations synergiques $\omega_{i,j}(t)$ conduit le système vers des **attracteurs** (états stables ou semi-stables). L'apparition d'un cluster correspond alors à un **bassin d'attraction** particulier, où les liens internes se renforcent suffisamment pour maintenir l'entité groupée, un phénomène typique des **systèmes dynamiques complexes**.

Auto-Régulation

Dans un système auto-organisé, chaque composant (ou chaque lien) s'**ajuste** en fonction de **règles locales** : si la coopération est jugée bénéfique, le lien augmente ; si elle est néfaste ou inutile, le lien diminue. Ce **mécanisme d'ajustement** relève de l'**auto-régulation**, car il ne dépend pas d'une autorité externe qui dirait “ces entités doivent se connecter, celles-ci doivent se séparer”.

Comme décrit en section 1.2.4, une règle fréquemment utilisée dans le DSL pour modéliser l'évolution des pondérations synergiques $\omega_{i,j}(t)$ est :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

- η représente un taux d'apprentissage (contrôle la rapidité de l'ajustement).
- $S(\mathcal{E}_i, \mathcal{E}_j)$ est la **synergie** calculée entre les entités i et j .
- τ agit comme un terme de décroissance ou de “frottement”, évitant que les pondérations ne croissent indéfiniment.

Le paramètre τ et le choix de la fonction $S(\cdot)$ permettent d'**éviter les dérives** (ex. explosion des connexions) et de stabiliser le système dans un régime d'**énergie** ou de **coût** relativement bas (voir fonction \mathcal{J} discutée en 1.2.6). Ce mécanisme d'auto-régulation fait qu'on n'a pas besoin de redéfinir manuellement la topologie du réseau : elle s'ajuste d'elle-même aux contraintes et aux données entrantes.

Feedback (Rétroaction)

Le **feedback** désigne le processus par lequel la **sorite** d'un système ou l'**état** d'une partie du système **retourne** sur lui-même, influençant à son tour la configuration ou le comportement du système. Dans le DSL, le feedback se matérialise surtout à travers l'**apprentissage** des synergies : la façon dont deux entités ont coopéré à l'instant t affecte la pondération $\omega_{i,j}(t+1)$, qui influencera à son tour leur coopération future.

Le **feedback positif** est un mécanisme par lequel une forte synergie entre deux entités, notée $S(\mathcal{E}_i, \mathcal{E}_j)$, conduit à un renforcement mutuel de leur connexion. En d'autres termes, si la synergie initiale entre deux entités est élevée, leur interaction devient progressivement plus forte, renforçant encore davantage leur synergie. Ce processus peut continuer jusqu'à atteindre un certain plafond, au-delà duquel la connexion ne peut plus croître, stabilisant ainsi leur relation synergique.

À l'inverse, le **feedback négatif** intervient lorsque la synergie entre deux entités est faible ou nulle, voire négative. Dans ce cas, la pondération de leur lien décroît progressivement, réduisant la probabilité qu'elles interagissent à l'avenir. Cependant, ce mécanisme n'est pas définitif : un changement de contexte ou d'état pourrait, à terme, améliorer leur compatibilité et rétablir leur capacité à collaborer ou à former une connexion.

Outre les liens binaires (entre paires d'entités), il peut exister des **boucles de rétroaction plus complexes** à l'échelle de plusieurs clusters. Par exemple, un cluster \mathcal{C}_1 peut influencer la formation ou la dissolution d'un autre cluster \mathcal{C}_2 en modifiant le contexte, les ressources partagées, etc. Sur le plan mathématique, cela équivaut à considérer des **fonctions de synergie n-aire**, où les “coûts” ou “gains” de chaque liaison dépendent aussi des états des autres liens et entités.

$$S(\mathcal{E}_i, \mathcal{E}_j | \{\mathcal{E}_k\}_{k \neq i,j}), \quad \omega_{i,j}(t+1) = f(\omega_{i,j}(t), \{S(\mathcal{E}_i, \mathcal{E}_j | \dots)\}, \dots).$$

Synthèse

Les trois concepts d'**émergence**, d'**auto-régulation** et de **feedback** interagissent étroitement dans le Deep Synergy Learning :

- L'**émergence** se produit parce que les liens (pondérations synergiques) se développent localement sans qu'aucun schéma global ne soit imposé.

- L'**auto-régulation** assure que chaque lien est entretenu ou affaibli en fonction de la **pertinence** mesurée (synergie), maintenant ainsi une forme de **plasticité** dans l'ensemble du réseau.
- Le **feedback** perpétuel, tant au niveau des paires d'entités que des ensembles plus larges, crée un **cycle** où l'état du réseau à l'instant t influence les coopérations à l'instant $t + 1$, et ainsi de suite.

Ce mode de fonctionnement “bouclé” et distribué est très différent de la logique hiérarchique classique (voir 1.2.4). Il rappelle davantage les **systèmes biologiques** et autres **systèmes complexes**, où la stabilité et les structures apparaissent comme le fruit d'interactions locales répétées, produisant parfois des schémas surprenants d'ordre (ou de désordre) global.

Comprendre les notions d'**émergence**, d'**auto-régulation** et de **feedback** est crucial pour saisir pourquoi le **DSL** peut offrir des capacités d'adaptation et de créativité supérieures à celles des architectures figées. Loin d'être un simple attribut de “design” théorique, ces concepts constituent la **dynamique interne** qui permet au réseau synergique de se **reconfigurer** en permanence, de **fusionner** ou de **séparer** des entités en clusters, et d'**apprendre** sans devoir figer sa topologie.

Dans la section suivante (1.3.3), nous mettrons en perspective ces mécanismes d'auto-organisation avec les **méthodes d'apprentissage classiques**, afin d'explorer les gains potentiels (et les défis) qu'apporte l'introduction de ces idées dans le champ de l'intelligence artificielle.

1.3.3. Comparaison avec les Méthodes d'Apprentissage Classiques

Les principes d'**auto-organisation** et de **synergie** du Deep Synergy Learning (DSL) contrastent fortement avec les schémas employés par les méthodes d'apprentissage les plus courantes, qu'il s'agisse d'algorithmes supervisés, non supervisés ou même de certaines approches en renforcement. Dans la présente section, nous mettons en lumière les principales différences entre ces paradigmes “classiques” (tels que le **Deep Learning** hiérarchique, la **clustering** statique ou la **rétropropagation de gradient** end-to-end) et le **DSL**, qui s'appuie avant tout sur une **construction dynamique** du réseau via la synergie informationnelle.

Dans un **cadre supervisé** classique, on dispose d'un ensemble de données $\{(\mathbf{x}_i, y_i)\}_{i=1,\dots,N}$, où \mathbf{x}_i constitue l'entrée (vecteur de caractéristiques, image, etc.) et y_i le label associé (catégorie, valeur numérique, etc.). Un **modèle** (par exemple, un réseau de neurones) est alors **entraîné** à minimiser une **fonction de coût** :

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^N L(f_\theta(\mathbf{x}_i), y_i),$$

où f_θ désigne le modèle paramétré par θ (poids et biais), et $L(\cdot)$ une fonction de perte (entropie croisée, MSE, etc.).

La plupart du temps, l'**architecture** du réseau est **figée** : on choisit un nombre de couches, une topologie (CNN, MLP, etc.) et on la conserve durant tout l'entraînement. La **rétropropagation** permet d'ajuster les poids internes θ , sans pour autant réorganiser la **structure** (aucune suppression ni création de neurones, aucune reconfiguration des connexions).

Dans un réseau **DSL**, on ne se limite pas à la mise à jour des paramètres : on autorise la **topologie** à évoluer (création, renforcement, fusion, ou rupture de liens) en se fondant sur la **mesure de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Dans un modèle supervisé classique, la structure reste **inchangée** tout au long de l'apprentissage.

Dans l'approche **supervisée** traditionnelle, l'effort se concentre sur la **réduction de l'erreur** par rapport aux labels y . À l'inverse, le DSL porte son attention sur la **relation** entre les différentes données elles-mêmes : la **formation** de clusters ou l'**auto-organisation** peut survenir même sans labels, dès que le réseau détecte des synergies fructueuses.

Un **modèle supervisé** doit souvent être **réentraîné** (ou au moins fine-tuné) dès lors qu'apparaissent de nouveaux types de données ou qu'une **distribution** se modifie. Dans le DSL, la structure s'**auto-réorganise** en intégrant ou en écartant

certaines entités, en fonction de la synergie relevée, ce qui apporte davantage de **souplesse** et de **plasticité** face aux environnements changeants.

Dans l'apprentissage **non supervisé**, on cherche généralement à regrouper des points \mathbf{x}_i en clusters. Des méthodes comme **k-means** démarrent avec un nombre de clusters (k) et une mesure de distance (par exemple euclidienne ou cosinus), puis procèdent à une série d'itérations visant à minimiser

$$\sum_{i=1}^N \min_{1 \leq c \leq k} \| \mathbf{x}_i - \boldsymbol{\mu}_c \|^2,$$

où $\boldsymbol{\mu}_c$ sont les centres de clusters. Dans **k-means**, le paramètre k doit être défini **a priori**, et même si d'autres algorithmes (tels que DBSCAN ou le clustering agglomératif) peuvent estimer le nombre de groupes, ils reposent néanmoins sur une **logique statique**, c'est-à-dire qu'on exécute l'algorithme une fois et on obtient une unique partition.

De plus, dans ces approches classiques, les points — ou vecteurs — ne bénéficient pas d'une **dynamique** interne : ils ne modifient ni leur représentation ni les liens qui pourraient évoluer au fil du temps en fonction d'une collaboration éventuelle. Le DSL se distingue précisément sur ce point : les **clusters** peuvent y **apparaître** ou se **dissoudre** de manière **progressive** selon les pondérations synergiques $\omega_{i,j}(t)$, sans qu'un paramètre k fixe ne borne le regroupement. La structure du réseau se **façonne** ainsi en continu, grâce à la dynamique d'apprentissage.

En outre, chaque entité \mathcal{E}_i du réseau DSL peut être **active**, c'est-à-dire posséder un **état interne** et évoluer au cours du temps. Dans un clustering **statique**, les points demeurent figés dans leurs coordonnées, sans aucun mécanisme de feedback local ou de synergie émergente. Enfin, lorsqu'apparaissent de nouvelles données ou qu'un changement de distribution survient, le DSL est capable de **réévaluer** les synergies et de **reconfigurer** automatiquement les clusters, tandis qu'un clustering non supervisé traditionnel exigerait la relance complète de l'algorithme, ignorant dans la foulée toute information antérieure.

Dans le **Reinforcement Learning**, un **agent** interagit avec un environnement, perçoit des **récompenses** (positives ou négatives) selon ses actions, et cherche à **maximiser** un **cumul de récompenses**. Les algorithmes de RL (Q-learning, SARSA, méthodes basées sur une politique) visent souvent à estimer la fonction de valeur $Q(\mathbf{s}, a)$ ou la politique optimale $\pi(a|\mathbf{s})$. De nombreux scénarios de RL (par ex. Deep Q-Network) s'appuient sur un **réseau profond** paramétré, dont les poids sont ajustés via un **objectif** portant sur la récompense cumulée.

Dans la plupart des implémentations RL traditionnelles, la **structure** du réseau reste fixée, on ne modifie pas, au cours de l'apprentissage, le nombre de couches, ni la forme du modèle.

Alors que le RL privilégie l'action menant à la **maximisation** de la récompense, le DSL promeut la **coopération** (ou la connexion) qui maximise la **synergie**. Il est possible d'envisager un "DSL-RL" où la synergie devient une composante de la **fonction de récompense** : plus la **collaboration** entre entités est jugée fructueuse, plus les liens correspondants se consolident.

Dans le RL classique, on n'ajoute ni ne supprime spontanément des neurones ou des liaisons durant l'apprentissage. À l'inverse, le DSL autorise une **recomposition** de l'architecture si cela contribue à améliorer la performance (ou la cohérence). Les pondérations peuvent émerger, se renforcer, ou disparaître au fil du temps.

On peut voir les entités du DSL comme autant d'"agents partiels" qui s'accordent via la **synergie**, au lieu de recourir à un **agent unique** muni d'un policy network fixe. Cette perspective diffère sensiblement du RL centralisé : l'organisation naît de l'**interaction** locale entre entités, plutôt que de se conformer à une unique stratégie imposée à l'ensemble du réseau.

Synthèse

Les approches classiques, qu'elles relèvent du **supervisé**, du **non supervisé** ou du **renforcement**, s'appuient généralement sur une **architecture** (ou un **ensemble d'hypothèses**) prédéfinie : il peut s'agir d'un réseau feed-forward, d'un certain nombre de clusters à trouver ou d'une paramétrisation donnée. À l'inverse, le **DSL** ne fixe pas la structure

à l'avance ; il permet à celle-ci de **se former** et de **se modifier** de manière autonome, simplement en s'appuyant sur les mesures de **synergie** et sur les **règles** d'auto-organisation.

Dans les méthodes classiques, on priviliege soit le **rapprochement** entre données et labels (apprentissage supervisé), soit la **répartition** de points dans des groupes (apprentissage non supervisé). Le DSL, au contraire, met l'accent sur la **relation** entre les entités, qu'il s'agisse de leur **coexistence** ou de leur **synergie**, et c'est ce mécanisme relationnel qui détermine l'organisation d'ensemble.

Ce renversement de perspective — considérer la donnée non pas comme un **input passif**, mais comme un **acteur** recherchant activement des **connexions** — constitue la marque distinctive du DSL. Les algorithmes traditionnels sont, de surcroît, souvent conçus pour un entraînement **en batch** : on recueille les données, on entraîne le modèle, puis on fige la configuration. Des versions en ligne existent, mais elles n'ont pas pour vocation de retoucher la **structure** du modèle ; elles n'agissent que sur les poids.

Le DSL, quant à lui, se prête naturellement à un **flux continu** de données : à mesure que de nouvelles informations apparaissent, le réseau peut se **reconstruire** pour répondre aux conditions changeantes ou accueillir de **nouvelles entités**, poursuivant ainsi une **adaptation** permanente plutôt que de s'en tenir à un unique schéma déterminé à l'avance.

Conclusion

La comparaison avec les méthodes classiques souligne le caractère **révolutionnaire** de l'auto-organisation prônée par le DSL : la capacité de faire “**évoluer la carte du réseau**” plutôt que de se restreindre à des architectures fixes, la prise en compte d'une **synergie** mesurée entre entités, la facilité d'**apprentissage continu** et la résistance aux changements de distribution.

Ces différences ne signifient pas que le DSL vienne nécessairement **remplacer** l'ensemble des paradigmes existants. On peut au contraire envisager des **approches hybrides** qui combinent la puissance des algorithmes classiques (supervisés ou RL) avec la **plasticité** et la **coévolution** propres au DSL. Les sections qui suivront (1.3.4 à 1.3.7) approfondiront l'idée d'une **auto-organisation multimodale**, ainsi que l'**impact** concret de cette approche sur la robustesse et la résilience des modèles, pour finalement ouvrir la voie à une IA plus générale et plus souple.

1.3.4. Le Rôle des Flux d'Information Multimodaux

Parmi les avantages majeurs de l'**auto-organisation** telle que proposée dans le **Deep Synergy Learning (DSL)** figure la capacité à gérer efficacement des **flux d'information** variés (image, texte, audio, données sensorielles, etc.). En effet, l'idée de **synergie informationnelle** (développée aux sections 1.2.2 et 1.2.3) s'applique de manière particulièrement féconde dès lors que plusieurs modalités sont en jeu, car chaque flux peut **renforcer** ou **compléter** les autres. Cette sous-section (1.3.4) illustre la manière dont l'auto-organisation des entités d'information s'adapte à la **multi-modalité**, et pourquoi cela peut conférer au DSL une **puissance d'intégration** hors de portée des architectures classiques, souvent cloisonnées.

Dans un système multimodal, on distingue plusieurs sources de données :

- **Modalité visuelle** : images, vidéos (entité $\mathcal{E}_{\text{visuelle}}$)
- **Modalité auditive** : signaux sonores, musique, parole (entité $\mathcal{E}_{\text{auditive}}$)
- **Modalité textuelle** : séquences de mots, documents, balises sémantiques (entité $\mathcal{E}_{\text{textuelle}}$)
- **Modalités sensorielles complémentaires** : capteurs physiques (température, distance, pression), signaux biométriques, etc.

Dans le **Deep Synergy Learning**, chaque flux peut être représenté par une ou plusieurs **entités d'information** (voir section 1.2.1). Ainsi, un flux visuel pourra donner naissance à plusieurs entités – par exemple, une entité pour la “carte de caractéristiques” (feature map) d'une image, une autre pour son histogramme de couleurs, etc. Ces entités ne sont

pas cloisonnées dans une couche unique : elles peuvent potentiellement interagir avec toutes les autres entités (visuelles, textuelles, auditives...), en fonction de leur **synergie**.

La **synergie** est particulièrement critique en multi-modalité :

$$S(\mathcal{E}_{\text{visuelle}}, \mathcal{E}_{\text{auditive}}) > 0 \Rightarrow \text{l'information visuelle et l'information auditive se complètent efficacement.}$$

Par exemple, dans une scène vidéo où l'on entend quelqu'un parler, la composante auditive (voix) et la composante visuelle (mouvements de lèvres, expressions faciales) coopèrent pour renforcer la compréhension globale.

Lorsqu'une modalité est dégradée (par exemple, l'audio est parasité par du bruit de fond), une autre modalité (l'image) peut prendre le relais, augmentant la **tolérance aux perturbations**. Dans un **réseau synergique**, cela se traduit par une réduction dynamique du poids $\omega_{\text{audio},\text{visuel}}(t)$ si l'audio devient momentanément peu fiable, et par le renforcement d'autres liens synergiques.

Les entités associées à différentes modalités peuvent créer des **clusters** communs si leur synergie mutuelle est élevée. Mathématiquement, on peut repérer un cluster \mathcal{C}_m formé d'entités $\{\mathcal{E}_{\text{visuelle}}^a, \mathcal{E}_{\text{auditive}}^b, \mathcal{E}_{\text{textuelle}}^c\}$ tel que

$$\sum_{(i,j) \in \mathcal{C}_m \times \mathcal{C}_m} \omega_{i,j} > \theta_{\text{seuil}},$$

où θ_{seuil} est un paramètre symbolisant la cohésion requise pour se constituer en cluster. Ce regroupement favorise l'**intégration sémantique** de différents types de signaux.

Dans un réseau hiérarchique classique, la fusion multimodale est généralement réalisée dans une “couche” dédiée (p. ex. concaténation de features). Dans le DSL, la fusion naît **spontanément** :

$$\omega_{\text{visuelle},\text{textuelle}}(t+1) = \omega_{\text{visuelle},\text{textuelle}}(t) \eta [S(\mathcal{E}_{\text{visuelle}}, \mathcal{E}_{\text{textuelle}}) - \tau \omega_{\text{visuelle},\text{textuelle}}(t)].$$

Plus la coopération entre les deux modalités est jugée bénéfique (ex. augmentation de la performance ou de l'information mutuelle), plus ce lien se consolide.

Le fait qu'un cluster multimodal soit **éolutif** est crucial : en fonction du **contexte**, la modalité audio peut prendre plus ou moins d'importance (ex. : un concert de musique vs. un environnement silencieux). Les liens synergiques intermodaux varient alors dans le temps, faisant émerger ou disparaître des sous-ensembles cohérents.

On peut également enrichir la mesure de synergie S par un **contexte** $\mathbf{c}(t)$, de sorte qu'il y ait des synergies conditionnelles :

$$S(\mathcal{E}_i, \mathcal{E}_j | \mathbf{c}(t)).$$

Dans une scène nocturne, par exemple, la vision est moins fiable, tandis que l'audio reste pertinent. Le contexte “nuit” pourrait diminuer certaines liaisons visuelles et renforcer des liaisons audio ou thermiques, mettant l'accent sur la modalité la plus fiable.

Exemples

Imaginez un système de surveillance qui reçoit en continu des flux vidéo, audio, et des capteurs de mouvement. Dans une approche classique, il faudrait concevoir à l'avance un schéma de fusion ; dans le DSL, les **entités** correspondant à chaque flux s'**auto-organisent** pour détecter des patrons communs (ex. détection d'un intrus combinant une silhouette anormale et des sons inhabituels).

Le **renforcement** des liens entre un flux “caméra infrarouge” et un flux “microphones” peut se produire si ces deux modalités se complètent pour la détection nocturne.

Dans le cadre d'interfaces multimodales (voix, gestes, expressions faciales), chaque modalité peut être captée par une entité DSL. Si l'utilisateur parle tout en faisant des gestes, le cluster "voix + gestes" obtient un fort score de synergie, permettant de mieux interpréter la signification globale (commande ou intention).

L'auto-organisation permet au système de **s'adapter** aux préférences de l'utilisateur : si celui-ci a une diction peu claire mais des gestes très expressifs, les pondérations vers la modalité gestuelle se renforcent spontanément.

Le DSL peut traiter des couples "image + légende textuelle" : une entité "extrait de texte" peut découvrir qu'elle augmente sa synergie avec une entité "détecteur d'objets" dans l'image, notamment si certains mots clés coïncident avec des formes reconnues.

Les entités "texte" et "image" peuvent alors former un **cluster** spécifique de type "concept visuel + label textuel", facilitant les tâches d'annotation automatique ou de recherche d'images par mots-clés (et vice versa).

Forces et défis de l'approche synergique multimodale

L'**absence** de couche de fusion prédéfinie procure une grande **flexibilité** : au lieu d'imposer un schéma de combinaison entre les flux (visuel, audio, texte, etc.), le réseau (DSL) **décide** de lui-même quelles modalités s'avèrent les plus pertinentes à associer. Cette adaptabilité favorise également la **robustesse** : si un flux devient bruité ou se perd (par exemple, un microphone défectueux), le système peut se reconfigurer et compter davantage sur d'autres canaux. Enfin, l'**auto-organisation** ouvre la voie à la **découverte** de combinaisons inédites : deux modalités supposées "peu corrélées" peuvent révéler une forte synergie dans un certain contexte, ce qu'un pipeline classique, plus rigide, aurait pu manquer.

En contrepartie, la **complexité computationnelle** peut monter rapidement à mesure que le nombre de modalités et d'entités augmente : évaluer le **gain de synergie** entre multiples canaux peut devenir coûteux, nécessitant l'emploi d'**heuristiques** ou d'**approximations**. De plus, l'enchaînement rapide de contextes (passer d'une scène de rue à une scène d'intérieur, par exemple) peut induire une **volatilité** élevée des connexions synergiques ; il faut alors recourir à des **coefficients de régulation** (τ) ou à des dispositifs de mémorisation pour **stabiliser** la configuration. Enfin, l'**interopérabilité** entre flux implique de disposer de **représentations** comparables ou d'un espace commun adapté (par exemple, une même métrique pour estimer la similarité entre un vecteur d'image et un vecteur de texte).

Conclusion

Le **Deep Synergy Learning** apparaît particulièrement adapté à l'intégration de **flux multimodaux**. Là où les méthodes classiques imposent généralement une architecture de fusion fixe (par exemple, concaténer l'output d'un CNN d'image et d'un RNN de texte), le DSL laisse les **entités** (issues de différentes modalités) **explorer** leurs synergies possibles. Les liens synergiques se **cristallisent** ou se **dissolvent** en fonction de leur valeur ajoutée, permettant une **adaptation dynamique** aux contextes et aux sources de bruit.

Cette approche auto-organisée favorise l'**émergence** (voir section 1.3.2) de clusters multimodaux plus ou moins stables, qui peuvent évoluer au fil du temps. Sur le plan applicatif, il en résulte une **robustesse** et une **flexibilité** uniques, aussi bien pour la reconnaissance d'événements complexes, la détection d'anomalies, ou encore la mise en place d'interactions homme-machine riches. Les sections suivantes (1.3.5, 1.3.6, 1.3.7) examineront l'**évolution dynamique** de ces réseaux, l'**impact** sur la **résilience** des modèles, et la perspective d'une **approche plus globale** de l'apprentissage.

1.3.5. Évolution Dynamique et Adaptation en Continu

Le **Deep Synergy Learning (DSL)** se distingue des approches classiques de l'IA par sa capacité à laisser les entités d'information et leurs connexions **évoluer** en permanence, au gré de la synergie détectée entre les flux de données. Cette caractéristique d'**évolution dynamique** et d'**adaptation en continu** s'inspire directement de divers phénomènes naturels (cerveau, écosystèmes, colonies d'insectes...) et confère au DSL une **plasticité** inhabituelle dans le domaine

de l'apprentissage automatique. Dans cette section, nous analysons pourquoi cette évolution constante est essentielle, quels sont ses principes fondamentaux et comment elle se formalise mathématiquement.

Dans le **DSL**, l'apprentissage ne se cantonne pas à une phase d'entraînement figée ; il se présente comme un **processus d'auto-organisation** continu, actif tant que le réseau demeure en service. Chaque entité peut ainsi **affiner** ou **réécrire** ses connexions et son rôle lorsqu'affluent de nouvelles données, ou même de nouvelles entités.

Contrairement à un réseau neuronal classique, la **structure** du réseau n'est pas fixée : les pondérations synergiques $\omega_{i,j}(t)$ peuvent **croître** ou **décroître**, au point de passer sous un **seuil** de rupture ω_{\min} qui entraîne la disparition effective de la liaison entre \mathcal{E}_i et \mathcal{E}_j . Ce mécanisme libère des ressources et prévient l'encombrement du réseau. À l'inverse, si une synergie autrefois négligeable devient significative, une **nouvelle liaison** peut se former pour consolider la coopération entre entités (voir la section 1.2.6 concernant les notations).

L'environnement ou la distribution des données peut également fluctuer : augmentation du bruit, apparition de nouvelles classes, transformation de contexte, etc. Dans un schéma statique, il faudrait réentraîner ou procéder à un "fine-tuning" coûteux. Dans le DSL, au contraire, l'**adaptation** est **naturellement** assurée par la mise à jour permanente des **liens synergiques** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Si les informations nouvellement introduites modifient la synergie entre entités, la **topologie** du réseau se **recompose** spontanément, sans nécessité de réapprentissage global.

On peut représenter le réseau synergique à l'instant t par le graphe $G(t)$, constitué de nœuds $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ et d'arêtes pondérées $\omega_{i,j}(t)$. L'évolution dans le temps se décrit par :

$$G(t+1) = \mathcal{F}(G(t), \mathcal{D}(t), \Theta),$$

où $\mathcal{D}(t)$ désigne les **données reçues** à l'instant t (ou sur la fenêtre $[t, t + \Delta]$), et Θ englobe les paramètres de mise à jour (taux η , régularisation τ , etc.).

Chaque entité \mathcal{E}_i peut maintenir un **état** $\mathbf{s}_i(t)$ permettant d'intégrer une forme de **mémoire** (cf. 1.2.1). Cet état influe sur la synergie :

$$S(\mathcal{E}_i, \mathcal{E}_j) = f(\mathbf{s}_i(t), \mathbf{s}_j(t), \mathcal{D}(t), \dots).$$

On a donc un **système dynamique couplé**, dans lequel la mise à jour des connexions $\omega_{i,j}(t)$ et celle des états $\mathbf{s}_i(t)$ interagissent :

$$\begin{cases} \omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)], \\ \mathbf{s}_i(t+1) = \mathbf{s}_i(t) + \gamma g(\mathbf{s}_i(t), \{\omega_{i,k}(t)\}, \dots). \end{cases}$$

On obtient alors un **couplage** entités-réseau qui peut donner naissance à des attracteurs, des cycles limites ou même des phénomènes de bifurcation.

Au-delà de la simple création/suppression de liens, il est possible que deux entités \mathcal{E}_a et \mathcal{E}_b **fusionnent** si leur collaboration est très constante, formant une **nouvelle entité** \mathcal{E}_{ab} . De même, une entité fortement hétérogène peut se **scinder** en plusieurs entités spécialisées si cela améliore la synergie globale. Ces mécanismes se décrivent par des règles de type :

$$\text{Fusion}(\mathcal{E}_a, \mathcal{E}_b) \Leftrightarrow \sum_{t'=t_0}^{t_1} \omega_{a,b}(t') > \delta_{\text{fusion}},$$

avec δ_{fusion} un seuil de stabilité sur une durée $[t_0, t_1]$. De tels phénomènes rappellent les “assemblées neuronales” durables ou la “spéciation” dans un contexte évolutionniste.

De nombreux problèmes réels (prévision financière, capteurs industriels, suivi de l’activité cérébrale, etc.) impliquent des distributions de données qui **changent** dans le temps. Le DSL, grâce à son **évolution dynamique**, s’ajuste progressivement plutôt que de s’en tenir à un modèle figé.

Dans un réseau traditionnel, apprendre une nouvelle tâche (ou un nouveau domaine) peut faire oublier les acquis précédents, faute de mécanisme de préservation structurelle. Dans le DSL, rien n’oblige à redémarrer l’optimisation : les clusters utiles peuvent **persistir**, tandis que de nouvelles entités se lient et modulent la synergie pour assimiler un contexte différent.

Au fur et à mesure de l’évolution, si certaines liaisons n’apportent plus de gain, elles s’effacent ou diminuent. Ce phénomène **régule** la complexité du réseau en évitant une explosion exponentielle du nombre de liens. On peut donc voir le DSL comme un réseau à **topologie parcimonieuse** autorégulée.

Exemples

Un **système DSL** qui reçoit en continu des mesures (température, pression, vibrations) peut servir à détecter des anomalies ou à anticiper des pannes. Lorsque les conditions d’usage se modifient progressivement (usure, évolution de la chaîne de production), le réseau **réorganise** ses liens synergiques : certains capteurs découvrent une complémentarité avec d’autres, pendant que certains liens historiques deviennent moins utiles et s’affaiblissent. Cette **auto-organisation** permet de maintenir une alerte fiable, sans qu’il soit nécessaire de procéder à un nouvel entraînement global à intervalles réguliers.

Un **chatbot multimodal** traitant à la fois la voix, le texte et l’image du visage (via webcam) illustre également l’adaptabilité d’un réseau évolutif. À mesure que l’utilisateur se familiarise avec le système (introduction de nouveaux termes, évolution de ses habitudes gestuelles), les entités “voix” et “vision” ajustent leurs connexions et peuvent, le cas échéant, **former un cluster** spécialisé pour reconnaître l’utilisateur concerné. Les anciennes tâches (reconnaître d’autres utilisateurs) ne sont pas oubliées, car elles persistent dans des clusters distincts ; la connaissance acquise est ainsi **préservée** sur le long terme.

Limites et défis de l’évolution continue

Si le réseau cherche à suivre de trop près les **fluctuations** de la synergie, il peut devenir **instable**, oscillant sans cesse ou détruisant prématurément des liens pertinents. Les paramètres η , τ , δ_{fusion} , etc., exigent donc un **calibrage** soigné pour maintenir un **équilibre** entre plasticité et stabilité (c’est le “dilemme stabilité–plasticité”).

Par ailleurs, dans un **modèle statique**, on fige les poids après l’apprentissage et on évalue la performance sur un ensemble de test. Dans le cadre d’un **réseau évolutif**, la performance est **susceptible de varier** au fil du temps, rendant nécessaire l’adoption de **métriques d’apprentissage continu** (p. ex. une mesure de l’erreur sur une fenêtre glissante).

Enfin, la reconfiguration permanente des liens implique un **coût de calcul**. Si le réseau compte n entités et que la synergie est évaluée entre toutes les paires, on fait face à $O(n^2)$ opérations. Des stratégies de **parsimonie** (seuil de rupture, mise à jour partielle, échantillonnage) sont souvent indispensables pour demeurer **scalable** dans des systèmes de grande ampleur.

Conclusion

L’**évolution dynamique** et l’**adaptation en continu** font du **Deep Synergy Learning** une forme de réseau “vivant”, capable de remodeler son organisation interne au fur et à mesure que l’environnement (ou les données) se transforment. Ce **changement de paradigme** – de l’entraînement ponctuel et figé à une auto-organisation permanente – ouvre la

voie à des **applications** de l'IA plus flexibles et plus robustes, réduisant la dépendance à la supervision humaine et accroissant la **longévité** des modèles dans des scénarios réels et évolutifs.

Les sections suivantes (1.3.6 et 1.3.7) creuseront deux aspects cruciaux de cette perspective : la **robustesse** et la **résilience** que procure l'auto-organisation (1.3.6) et les **pistes** pour une approche plus générale de l'apprentissage (1.3.7), potentiellement orientée vers une IA plus proche de la cognition ou de l'**intelligence générale**.

1.3.6. Impacts sur la Robustesse et la Résilience des Modèles

L'une des conséquences les plus marquantes de l'**auto-organisation** et de la **synergie informationnelle** dans le **Deep Synergy Learning (DSL)** réside dans l'**amélioration** de la **robustesse** et de la **résilience** des systèmes d'apprentissage. Contrairement aux approches classiques, souvent vulnérables aux perturbations, aux changements de distribution ou aux défaiillances partielles, le DSL tire parti de sa structure **évolutive** et **coopérative** pour mieux absorber ces aléas. Dans la présente section, nous examinons les notions de robustesse et de résilience, puis montrons comment elles s'expriment dans un réseau synergique.

1.3.6.1. Définitions et enjeux

La **robustesse** d'un modèle désigne sa capacité à **maintenir** un niveau de performance élevé malgré la présence de **perturbations** ou d'**incertitudes**. Dans le contexte de l'IA, ces perturbations peuvent inclure :

- Du **bruit** dans les données (capteurs défectueux, images floues, signaux audio corrompus, etc.).
- Des **attaques adversariales** (petites perturbations “adversarial noise” rendant le modèle confus).
- Des **incohérences** ou **lacunes** dans les échantillons (valeurs manquantes, distribution très variée).

Un système robuste parvient à faire face à ces dégradations sans “collapsus” brutal de la performance.

La **résilience** va plus loin que la robustesse en impliquant non seulement la résistance aux perturbations, mais aussi la **capacité à récupérer** ou à **se réorganiser** après un choc. En IA, cela signifie que si la distribution des données change, ou si le système subit une panne partielle (certains capteurs tombent en panne, certains flux de données disparaissent), la résilience se manifeste par la **reconfiguration** interne permettant de continuer la tâche ou de s'en rapprocher au mieux.

1.3.6.2. Mécanismes de robustesse dans un réseau synergique

Dans le DSL, plusieurs entités d'information peuvent recouvrir partiellement la même “zone de compétence” ou la même modalité, tout en se distinguant suffisamment pour apporter une **valeur ajoutée**. Du point de vue de la robustesse, cela signifie qu'un flux corrompu (ou une entité dysfonctionnelle) n'est pas catastrophique, car d'autres entités peuvent **prendre le relais**.

$$\omega_{i,j}(t) \rightarrow \omega_{k,j}(t+1), \text{ si } \mathcal{E}_i \text{ défaillante, alors } \mathcal{E}_k \text{ peut compenser.}$$

Lorsque la synergie entre une entité défectueuse \mathcal{E}_{def} et les autres entités chute (bruit, erreur répétée, etc.), les pondérations $\omega_{\text{def},j}(t)$ s'affaiblissent progressivement. Le réseau “apprend” ainsi à **alléger** la connexion avec la source fautive et à **rediriger** l'information vers des entités plus fiables. Ce mécanisme évite un fort impact d'une entité isolée dysfonctionnelle.

Les entités réagissent localement au **bruit** ou aux **incohérences** en resserrant leurs liens synergiques avec celles qui restent cohérentes. On observe alors la création de **clusters** (voir 1.2.6) qui s'auto-assemblent autour de flux fiables ou complémentaires, renforçant la robustesse globale.

$$\mathcal{C}^* = \arg \max_{\mathcal{C}} \sum_{(i,j) \in \mathcal{C} \times \mathcal{C}} \omega_{i,j}, \quad \text{en excluant } \mathcal{E}_{\text{def.}}$$

1.3.6.3. Résilience via l'adaptation en continu

Le DSL met en œuvre un apprentissage **continu** (section 1.3.5), ce qui signifie que le réseau s'ajuste aux nouvelles données ou aux nouveaux contextes sans nécessiter une refonte globale. Ainsi, si une modalité devient soudainement imprécise (caméra saturée de lumière, micro exposé à un fort bruit ambiant), le réseau réorganise ses **pondérations synergiques** pour ne plus dépendre de cette source.

$$\omega_{\text{camera}, \mathcal{E}_j}(t+1) = \omega_{\text{camera}, \mathcal{E}_j}(t) - \eta [\tau \omega_{\text{camera}, \mathcal{E}_j}(t)] \quad \text{si } S(\mathcal{E}_{\text{camera}}, \mathcal{E}_j) \approx 0 \text{ (bruit élevé).}$$

La résilience suppose la **récupération** ou le maintien d'une performance acceptable même après un choc. Le fait de **supprimer** ou d'**affaiblir** les liens inutiles (ou trompeurs) et de **renforcer** les liens pertinents génère une reconfiguration topologique :

$$G(t+1) = \mathcal{U}[G(t), \{\omega_{i,j}\}, \text{nouveaux contextes}].$$

Ainsi, le réseau peut se redessiner sans qu'on doive recourir à une procédure hors-ligne complexe.

Les entités (et clusters) qui étaient **efficaces** pour d'anciennes tâches peuvent persister en parallèle, assurant une forme de **mémoire** ou de **transfert** vers des scénarios futurs. Cette cohabitation d'anciens et de nouveaux liens améliore la résilience, car le réseau ne jette pas systématiquement ses anciennes connaissances : il les conserve tant qu'elles restent synergiques, évitant le phénomène de "catastrophic forgetting".

1.3.6.4. Exemples d'Applications Robustes et Résilientes

Le **Deep Synergy Learning (DSL)** se révèle particulièrement adapté aux situations où la robustesse et la résilience constituent des critères essentiels. Deux scénarios illustrent la manière dont la **synergie informationnelle** et la **dynamique auto-organisée** du DSL permettent d'absorber des pannes, de réagir à des anomalies et de préserver la continuité opérationnelle. Dans chaque cas, la structuration coopérative du réseau offre une forme de **tolérance aux défaillances**, car l'affaiblissement d'un module ou d'un capteur ne remet pas en cause l'ensemble de la configuration. Les exemples ci-dessous mettent en évidence ces mécanismes de résilience et soulignent les **bienfaits** d'un réseau qui se **reconfigure** en temps réel.

A. Robotique et Systèmes Autonomes

Considérons un **robot** doté de multiples capteurs, tels que des caméras, un LIDAR, des gyroscopes, ainsi que d'autres senseurs spécialisés. Dans ce contexte, un modèle DSL peut absorber la défaillance partielle ou totale d'un capteur, sans qu'il soit nécessaire de procéder à un réapprentissage global. Cette propriété découle du caractère **auto-adaptatif** du Synergistic Connection Network (SCN) sous-jacent, où chaque capteur \mathcal{C}_i est traité comme une **entité** de l'ensemble, tandis que les liens $\omega_{i,j}$ mesurent la **coopération** ou la **complémentarité** de deux capteurs i et j .

Dans le cas d'une **défaillance**, supposons que le capteur \mathcal{C}_k soit victime d'un bruit extrême ou d'une panne soudaine. Le **score de synergie** $S(\mathcal{C}_k, \mathcal{C}_m)$ avec les autres capteurs \mathcal{C}_m diminuerait alors, ce qui provoquerait une mise à jour négative des liens $\omega_{k,m}$ par l'équation

$$\omega_{k,m}(t+1) = \omega_{k,m}(t) + \eta [S(\mathcal{C}_k, \mathcal{C}_m) - \tau \omega_{k,m}(t)],$$

où η désigne le taux d'apprentissage local et τ le coefficient de régulation. À mesure que la **synergie** s'affaiblit, la pondération $\omega_{k,m}$ décroît, de sorte que ce capteur devient quasi inactif au sein du SCN. Les autres capteurs, préservant des liens ω élevés entre eux s'ils demeurent fiables et cohérents, continuent de coopérer pour fournir les informations nécessaires à la navigation ou à la réalisation des tâches robotiques. L'ajustement s'effectue en continu, sans qu'un entraînement complet soit relancé, montrant ainsi la **résilience** inhérente au DSL.

B. Surveillance Critique

Dans les domaines de la **surveillance** ou du **monitoring** (installations sensibles, applications médicales, détection d'intrusions), il est fréquent de recourir à un grand nombre de capteurs hétérogènes. Un système basé sur le DSL peut y déployer ses mécanismes d'**auto-organisation** de manière à détecter plus vite d'éventuelles anomalies et à tolérer les dysfonctionnements partiels d'un ou plusieurs capteurs. Le déploiement s'opère suivant la logique où chaque capteur \mathcal{D}_i constitue un **nœud** du SCN, et où le **score de synergie** $S(\mathcal{D}_i, \mathcal{D}_j)$ quantifie la correspondance ou la complémentarité entre deux capteurs.

Lorsque l'un de ces nœuds présente une anomalie ou se déconnecte, la **pondération** $\omega_{i,j}$ chute au fil des itérations, car la synergie mesurée devient faible. Mathématiquement, le processus de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{D}_i, \mathcal{D}_j) - \tau \omega_{i,j}(t)]$$

réduit progressivement les liens inutilisables ou compromis. La structure globale demeure néanmoins cohérente, car les capteurs restant fiables maintiennent des valeurs de ω importantes, ce qui préserve la capacité du réseau à **fonctionner**.

Le phénomène inverse se produit lorsqu'une alarme est déclenchée de façon synchronisée entre plusieurs capteurs. Dès lors que ceux-ci fournissent un signal convergent sur la présence d'un événement inhabituel, leur **synergie** augmente, et ils forment un **cluster** spécialisé dans l'analyse d'un risque potentiel. Cette coopérative rapide et flexible permet au SCN de hiérarchiser l'information critique. L'avantage repose sur l'**absence de réapprentissage intégral** et sur la capacité du système à s'**auto-réorganiser** pour mettre en avant les flux les plus pertinents.

Conclusion

Que ce soit dans un scénario de **robotique** multi-capteurs ou dans un dispositif de **surveillance** critique, le **Deep Synergy Learning** démontre ainsi son **aptitude** à gérer la panne ou la perte de fiabilité d'un sous-module sans perturber l'ensemble de l'architecture. La dynamique de **mise à jour** des synergies et des liens ω assure une continuité de service et un renforcement rapide des signaux valides, ce qui illustre la solidité opérationnelle et l'efficacité de cette **approche auto-organisée** pour la gestion des situations critiques ou complexes.

1.3.7. Perspectives pour une Approche plus Globale de l'Apprentissage

Les sections précédentes (1.3.1 à 1.3.6) ont mis en évidence l'importance de l'**auto-organisation** dans le Deep Synergy Learning (DSL), ses fondements biologiques et cognitifs (1.3.1), les concepts d'émergence et de feedback (1.3.2), la comparaison avec les méthodes classiques (1.3.3), le rôle crucial de la multi-modalité (1.3.4), la dynamique adaptative (1.3.5) et l'impact sur la robustesse/résilience (1.3.6). Nous en arrivons maintenant à la question centrale : **dans quelle mesure** cette architecture auto-organisée peut-elle contribuer à une **approche plus globale** de l'apprentissage, allant potentiellement vers une IA plus générale, moins cloisonnée et plus proche de la cognition humaine ?

1.3.7.1. Vers une intégration de multiples paradigmes

L'un des attraits majeurs du DSL réside dans sa **flexibilité** structurelle, permettant l'intégration simultanée de tâches variées (reconnaissance d'images, traitement du langage, analyse de signaux). Plutôt que de multiplier les sous-modules indépendants, le DSL **mutualise** la capacité d'adaptation et de co-évolution. Cela ouvre la voie à des systèmes capables d'aborder des **problèmes multi-domaines** de façon unifiée.

Beaucoup de chercheurs s'interrogent sur la possibilité de combiner une **représentation symbolique** (logique, règles, ontologies) et des **réseaux neuronaux** (approche sub-symbolique). Le DSL offre un espace où des **entités symboliques** (représentant des concepts, des règles) pourraient coexister avec des **entités sub-symboliques** (features non supervisées, clusters contextuels), et forger des **liens synergiques** si ces représentations s'avèrent mutuellement bénéfiques.

$$\omega_{\text{symbolique}, \text{sub-symbolique}}(t+1) = \omega_{\text{symbolique}, \text{sub-symbolique}}(t) \eta [S(\mathcal{E}_{\text{symbolique}}, \mathcal{E}_{\text{sub-symbolique}})].$$

Une autre piste consiste à inclure des **boucles de récompense** (façon apprentissage par renforcement) à l'intérieur du DSL, de manière à ce que la **synergie** prenne en compte non seulement l'interaction entre entités, mais aussi un **signal de performance** plus global. Cela pourrait mener à des architectures où l'**exploration** et la **sélection** des connexions synergiques s'effectuent au service d'une stratégie d'agent, par exemple dans un environnement en évolution (robotique, jeux, etc.).

1.3.7.2. Apprentissage contextuel et raisonnement adaptatif

Dans de nombreux problèmes (analyse de scène, dialogue, diagnostic médical), l'**environnement** ou la **situation** évolue. Le DSL permet de faire émerger des **clusters contextuels**, regroupant les entités les plus pertinentes pour un contexte donné. Ces clusters peuvent ensuite se **désagréger** ou se **recombiner** quand le contexte change, offrant un niveau de **contextualisation dynamique** qu'on retrouve rarement dans les modèles hiérarchiques figés.

Plutôt que de raisonner par règle (“si A alors B”), le réseau peut **découvrir** que certaines combinaisons d'entités $\{\mathcal{E}_i, \mathcal{E}_j, \dots\}$ produisent un **effet global** pertinent. On pourrait formaliser cette découverte comme un **raisonnement émergent**, où l'apparition d'un cluster synergique équivaut à la création d'un “concept” ou d'une “hypothèse” confirmée par l'amélioration de la performance ou de la cohérence.

$$\text{Concept}_{\alpha} \leftrightarrow \{\mathcal{E}_i, \mathcal{E}_j, \dots\} \quad \text{avec} \quad \sum_{(i,j) \in \alpha} \omega_{i,j} > \theta.$$

Les entités dans le DSL peuvent porter un **état interne** (1.2.1), ce qui permet de **mémoriser** certains événements, associations ou transitions. Couplé à l'idée de clusters, on obtient une forme de mémoire **distribuée** et **adaptative**, capable de se ré-agencer au fil du temps, au lieu d'être confinée dans des architectures rigides (ex. LSTM).

1.3.7.3. Ouverture vers une IA plus proche de la cognition biologique

Nous avons déjà souligné les parallèles entre la synapse biologique et les connexions synergiques (sections 1.3.1, 1.3.2). Un système auto-organisé qui évolue au fil du temps, fusionne ou dissocie des entités, renforce ou supprime des liens, s'**approche** d'une **dynamique neuronale** élémentaire. Il est donc envisageable que des **architectures DSL** se montrent plus aptes à **simuler** ou **comprendre** certains phénomènes cognitifs.

Bien que spéculative, la question d'une **conscience artificielle** ou d'une **auto-consistance cognitive** pourrait trouver dans le DSL un champ d'exploration. Certains travaux suggèrent qu'un degré d'**intégration d'information** (comme l'approche de la “Phi measure” de Giulio Tononi) est requis pour émerger une forme de conscience. Si le DSL parvient à développer des **clusters** hautement connectés et persistants (au sens “informationnellement intégrés”), il pourrait servir de terrain expérimental pour avancer sur ces hypothèses.

L'IA forte (IAG) ou Intelligence Artificielle Générale se définit par la capacité d'un système à maîtriser un large éventail de tâches, à apprendre de manière autonome et à faire face à des environnements variés. La **plasticité** et la **co-évolution** du DSL, sa manière de laisser chaque entité s'adapter, fusionner, se spécialiser, suggèrent un potentiel pour **surmonter** les obstacles auxquels se heurtent les méthodes rigides (catastrophic forgetting, absence de transfert, etc.). Cela ne garantit pas l'émergence d'une IAG, mais en offre une piste conceptuelle plus proche du fonctionnement adaptatif du vivant.

1.3.7.4. Principaux défis et voies de recherche

Le **Deep Synergy Learning (DSL)** introduit un cadre inédit, reposant sur une **interaction dynamique** et une **auto-organisation** continue des entités d'information. Malgré l'attrait de cette proposition, il subsiste plusieurs défis mathématiques, algorithmiques et pratiques, dont la résolution s'avère essentielle pour la généralisation à grande échelle. Sur le plan **computational**, l'évaluation des synergies entre un nombre important d'entités peut impliquer une complexité en $O(n^2)$, où n désigne le nombre d'entités. Cette explosion combinatoire requiert des stratégies de

parsimonie et de **sparsification** afin de restreindre la croissance du nombre de connexions à évaluer. Des techniques d'**échantillonnage** adaptatif peuvent aussi être envisagées, de manière à ne considérer que les paires d'entités présentant une synergie potentiellement élevée. D'un point de vue plus formel, on peut réduire la densité des connexions actives en imposant un mécanisme de seuil sur les poids ω_{ij} , de sorte qu'une liaison (i, j) ne soit conservée que si

$$\omega_{ij} > \omega_{\min},$$

avec ω_{\min} un paramètre choisi pour garantir la structure globale du réseau.

S'agissant de l'**interprétabilité** et de l'**explicabilité**, la capacité du DSL à faire émerger des **clusters** et à redessiner en continu ses connexions favorise une organisation plus fluide, mais rend ardue l'analyse a posteriori des décisions. Pour éclairer la nature d'une prédiction, il devient nécessaire de retracer l'historique des synergies et d'examiner l'état interne des entités concernées. La mise à jour de chaque poids ω_{ij} peut être décrite par une équation de type

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{ij}(t)],$$

où η désigne le taux d'apprentissage et τ le coefficient de décroissance. Cette dynamique rend possible l'apparition ou la suppression de liens dont la contribution n'est plus jugée pertinente, tout en renforçant ceux qui semblent les plus profitables au regard de la synergie mesurée. Cependant, un tel fonctionnement complique la traçabilité : pour extraire des règles ou des justifications, il faudrait stocker ou synthétiser l'ensemble des états successifs, ainsi que les évaluations de $S(\mathcal{E}_i, \mathcal{E}_j)$.

La question de la **convergence** suscite également un vif intérêt. L'étude de la stabilité et des attracteurs potentiels liés aux mises à jour des poids dans le cadre d'un **système dynamique** exige d'analyser l'évolution de l'**énergie** du réseau ou d'une **fonction de Lyapunov** associée. Si l'on considère une énergie

$$E(t) = - \sum_{i \neq j} \omega_{ij}(t) S(\mathcal{E}_i, \mathcal{E}_j),$$

la décroissance de $E(t)$ au fil des itérations peut servir d'indicateur de convergence vers un état stable où les pondérations cessent d'évoluer de manière significative. L'identification de conditions garantissant l'existence d'un minimum global ou local, ainsi que la caractérisation de la vitesse de convergence, demeurent des questions ouvertes pour la communauté scientifique. La présence de boucles de rétroaction complexes, de reconfigurations topologiques et d'interactions non linéaires peut engendrer des comportements oscillatoires, voire chaotiques, si aucune **régularisation** adéquate n'est mise en place.

Dans les **domaines critiques** (santé, finance, transport, sécurité), le DSL pourrait apporter une meilleure **robustesse** grâce à la **fusion multimodale** et à la **répartition adaptative** des informations, permettant de gérer des données hétérogènes ou partiellement manquantes. Toutefois, de telles applications exigent des garanties fortes sur la fiabilité et la sûreté de fonctionnement, impliquant des **processus de certification** exigeants. Il devient essentiel d'y intégrer des mécanismes de **sécurité** contre d'éventuelles attaques adversariales, où de subtiles perturbations de l'entrée peuvent tromper même des systèmes neuronaux performants. La dynamique adaptive et coopérative du DSL pourrait, d'un côté, renforcer la **résilience** en détectant des incohérences via les synergies internes, mais de l'autre côté, elle complique l'établissement de **bornes** de sûreté ou de **preuves** formelles de robustesse.

Ainsi, malgré un potentiel considérable, le DSL fait face à des enjeux théoriques et pratiques déterminants pour son adoption à grande échelle. La **réduction de la complexité algorithmique**, l'**amélioration de l'interprétabilité**, l'**étude minutieuse de la convergence** et l'**adaptation aux environnements critiques** constituent autant de pistes de recherche centrales pour consolider ce paradigme. Les avancées futures consisteront notamment à concevoir des **stratégies d'approximation** limitées en ressources de calcul, à développer des **méthodes d'exploration** et de **visualisation** capables d'exposer la structure émergente du réseau, et à établir des **cadres mathématiques** plus complets validant la stabilité et la fiabilité du processus d'**auto-organisation**. L'ensemble de ces travaux pourra contribuer à faire du Deep Synergy Learning un jalon clé dans l'évolution de l'IA vers une plus grande **plasticité**, **résilience** et **intelligence**.

1.3.7.5. Conclusion : vers un nouveau paradigme d'apprentissage

En somme, l'auto-organisation et la synergie, éléments centraux du **Deep Synergy Learning**, ouvrent la voie à une **approche plus globale** de l'apprentissage, susceptible d'englober plusieurs paradigmes existants (supervisé, non supervisé, renforcement, symbolique, etc.) tout en proposant des mécanismes d'évolution continue, de raisonnement contextuel et de mémorisation distribuée.

Cette vision n'est pas encore un aboutissement : de nombreux **défis** scientifiques, méthodologiques et éthiques (scalabilité, interprétabilité, robustesse, etc.) demeurent à relever. Néanmoins, le DSL fournit d'ores et déjà un **cadre unificateur** pour penser l'**IA adaptative**, apte à accueillir de multiples flux d'information et à co-évoluer de façon organique, en s'inspirant des systèmes vivants.

Les chapitres suivants approfondiront encore la **formulation mathématique** du DSL, ses **algorithmes** et ses **applications pratiques**, en vue de concrétiser ce nouveau paradigme et d'explorer son potentiel pour des systèmes intelligents, au-delà des limites imposées par les architectures hiérarchiques classiques.

Problème 1 : Analyse de la Dynamique des Pondérations Synergiques

On considère un réseau de n entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Les pondérations synergiques $\omega_{i,j}(t)$ évoluent selon la règle suivante :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ et $\tau > 0$ sont des constantes, et $S(\mathcal{E}_i, \mathcal{E}_j)$ représente une mesure (constante) de synergie entre les entités \mathcal{E}_i et \mathcal{E}_j .

1. Équilibre local

Montrez qu'à temps long, pour chaque paire (i, j) , la pondération $\omega_{i,j}(t)$ peut atteindre un point d'équilibre $\omega_{i,j}^*$. Exprimez $\omega_{i,j}^*$ en fonction de $S(\mathcal{E}_i, \mathcal{E}_j)$, η et τ .

2. Stabilité linéaire

Établissez la condition (en termes de η et τ) sous laquelle l'équilibre $\omega_{i,j}^*$ est **stable** dans le cadre d'une analyse linéaire (linéarisation autour de $\omega_{i,j}^*$).

3. Convergence globale

Proposez une justification ou un argument (non nécessairement exhaustif) permettant d'expliquer la **convergence** globale du système $\{\omega_{i,j}(t)\}$ vers un ensemble d'états d'équilibre, en supposant que tous les couples (i, j) obéissent à la même loi de mise à jour.

4. Impact de la régularisation

Supposons qu'on introduise un terme quadratique supplémentaire dans la mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{i,j} - \tau \omega_{i,j}(t)] - \alpha \omega_{i,j}(t)^3,$$

avec $\alpha > 0$. Discutez l'impact potentiel de ce terme "cubic" sur la stabilité et la répartition finale des $\omega_{i,j}(t)$. Quelles questions mathématiques (bifurcations, multiplicité des équilibres) cela soulève-t-il ?

5. Extension n-aire

Généralisez la dynamique ci-dessus à une **synergie n-aire**, $S(\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_n})$, où la mise à jour de $\omega_{i_1, \dots, i_n}(t)$ dépend d'un terme commun $\tau \omega_{i_1, \dots, i_n}(t)$. Proposez une formulation des équations et discutez le principal défi analytique pour montrer la convergence dans ce cas.

Problème 2 : Émergence de Clusters et Attracteurs de Graphe

Soit un graphe $G(t)$ dont les sommets sont les entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ et les arêtes sont pondérées par $\omega_{i,j}(t)$. On se propose d'étudier l'**émergence** de sous-graphes fortement connectés (clusters) en tant qu'attracteurs de la dynamique.

6. Fonction de coût globale

Introduisez une fonction "énergie" ou "coût" globale :

$$\mathcal{J}(G(t)) = - \sum_{(i,j)} \omega_{i,j}(t) S_{i,j} + \beta \sum_{(i,j)} [\omega_{i,j}(t)]^2,$$

où $S_{i,j}$ est la synergie (positive ou négative) et $\beta > 0$ un paramètre de régularisation. Expliquez pourquoi minimiser \mathcal{J} favorise la **création de clusters** de liens positifs (forte synergie) et la **suppression** de liens inutiles.

7. État stable et clusterisation

Montrez qu'un **état stable** (un minimum local de \mathcal{J}) peut correspondre à la formation d'un sous-ensemble \mathcal{C} tel que $\omega_{i,j}$ est élevé si $i,j \in \mathcal{C}$, et faible sinon. Expliquez, sur un plan mathématique, comment on peut interpréter \mathcal{C} comme un **cluster**.

8. Analyse de la multiplicité

Justifiez l'existence possible de **plusieurs** configurations d'équilibre local, chacune correspondant à un partitionnement différent du graphe. Quels outils d'optimisation ou de théorie des graphes peut-on invoquer pour analyser la multiplicité de ces minima ?

9. Transitions entre attracteurs

Décrivez un scénario mathématique permettant à un **sous-ensemble de sommets** \mathcal{C} de se dissocier en deux clusters \mathcal{C}_1 et \mathcal{C}_2 . Quelle condition sur la somme des synergies (interne vs. externe) pourrait induire la scission ? Dans quel cas obtient-on une fusion ?

Problème 3 : Analyse de la Stabilité–Plasticité dans l'Auto-Organisation

On considère la double contrainte “stabilité–plasticité” : le réseau doit être suffisamment **plastique** pour intégrer de nouveaux signaux, mais suffisamment **stable** pour ne pas détruire systématiquement les acquis précédents.

10. Formulation d'un critère mathématique

Proposez une formalisation où la “stabilité” \mathcal{S} et la “plasticité” \mathcal{P} sont deux fonctions mesurant respectivement la conservation des liens existants et la facilité de créer/supprimer des liens. Définissez un **objectif** $\Gamma(\mathcal{S}, \mathcal{P})$ à maximiser et discutez l’arbitrage mathématique qui en découle.

11. Stabilité locale vs. stabilité globale

Montrez comment on peut distinguer **stabilité locale** (analogie à la dérivée de $\omega_{i,j}$ près d'un équilibre) et **stabilité globale** (résilience à de grandes perturbations). Proposez un schéma d'analyse linéaire pour la stabilité locale des pondérations, et un schéma plus qualitatif pour la stabilité globale.

12. Équation couplée avec un taux de plasticité

Introduisez un **taux de plasticité** $\alpha(t)$ qui peut varier dans le temps :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\alpha(t)\eta [S_{i,j} - \tau\omega_{i,j}(t)].$$

Identifiez les conditions sur $\alpha(t)$ pour que le réseau n'entre pas dans des oscillations permanentes.

13. Compromis entre exploitation et exploration

Interprétez le rôle de $\alpha(t)$ comme un **compromis** entre “exploitation” (stabilité) et “exploration” (plasticité). Montrez, par un raisonnement mathématique ou par une étude de fonctions de Lyapunov, qu'un **décroissement progressif** de $\alpha(t)$ peut favoriser la convergence vers un cluster stable à long terme.

Problème 4 : Modélisation de la Multi-modalité et de la Synergie Conditionnelle

On souhaite étudier un système où chaque entité \mathcal{E}_i appartient à l'une de plusieurs **modalités** (par exemple : visuel, auditif, textuel). La synergie entre deux entités dépend aussi d'un **contexte c**.

14. Synergie conditionnelle

Formalisez la mesure :

$$S(\mathcal{E}_i, \mathcal{E}_j | \mathbf{c}),$$

où $\mathbf{c} \in \mathcal{C}$ est une variable contextuelle. Quels axiomes mathématiques (monotonie, symétrie partielle, etc.) peut-on imposer à $S(\cdot | \mathbf{c})$ pour qu'elle reste cohérente ?

15. Mise à jour des pondérations selon le contexte

Proposez une loi d'évolution contextuelle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta [S(\mathcal{E}_i, \mathcal{E}_j | \mathbf{c}(t)) - \tau \omega_{i,j}(t)].$$

Montrez comment cette équation peut, en pratique, conduire à la formation de **clusters spécifiques** à certains contextes \mathbf{c} .

16. Transfert de contexte

Supposons que le contexte \mathbf{c}_1 soit "nuit" et \mathbf{c}_2 soit "jour". Étudiez la possibilité d'un **transfert** de liens synergiques depuis \mathbf{c}_1 vers \mathbf{c}_2 si certaines entités se révèlent robustes aux deux contextes. Quel est le rôle d'une éventuelle **intersection** d'ensembles de clusters $\mathcal{C}_1 \cap \mathcal{C}_2 \neq \emptyset$?

17. Extension à la multimodalité n-aire

Discutez comment généraliser la synergie binaire à la synergie n-aire conditionnelle, $S(\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_n} | \mathbf{c})$. Quels problèmes de **complexité** surgissent si l'on veut mettre à jour l'ensemble $\omega_{i_1, \dots, i_n}(t)$ en fonction d'un contexte $\mathbf{c}(t)$?

Problème 5 : Étude Formelle de l'Émergence et de la Cognition Distribuée

Ce dernier problème se concentre sur la **dimension cognitive** et les possibilités d'aller vers une **approche globale** de l'apprentissage, inspirée par la section 1.3.7.

18. Assemblées auto-organisées et "concepts"

Modélisez mathématiquement la notion de "concept émergent" comme un **cluster** \mathcal{C} dont la somme des pondérations internes dépasse un seuil θ . Proposez une définition fonctionnelle :

$$\text{Concept}_\alpha := \left\{ \mathcal{E}_i \mid \sum_{j \in \alpha} \omega_{i,j} > \theta \right\}.$$

Comment prouver que ce concept est stable sous l'évolution des pondérations ?

19. Hiérarchie de concepts

Envisagez la construction d'une **hiérarchie** de concepts \mathcal{H} , où un "métaconcept" α^+ apparaît si deux concepts α_1 et α_2 ont une forte synergie inter-clusters. Formulez les conditions nécessaires :

$$\sum_{(i \in \alpha_1, j \in \alpha_2)} \omega_{i,j} > \theta_2,$$

et discutez la formation d'une structure arborescente ou en treillis.

20. Mémoire distribuée et clusters persistants

Définissez une **fondction de persistance** mesurant la durée pendant laquelle un cluster \mathcal{C} reste cohérent (c.-à-d. affiche des pondérations internes supérieures à un certain seuil). Prouvez ou argumentez qu'une entité ayant participé longtemps à un cluster \mathcal{C} garde une "mémoire partielle" de cette association, même si $\omega_{i,j}$ se réduit par la suite.

21. Discussion sur la convergence vers une IA forte

Proposez un **modèle** ou un **schéma** montrant comment ces mécanismes (formation de concepts, hiérarchie, mémoire persistante) peuvent contribuer à une forme plus globale de "cognition distribuée". Formulez au moins deux questions mathématiques ouvertes (liées aux attracteurs, à la complexité, ou à la dynamique asymptotique) qui se posent si l'on veut prouver la "capacité générale" d'un tel réseau.

22. Lien avec la théorie de l'information intégrée

(Optionnel) Évoquez le lien potentiel entre la somme de synergies $\sum_{i,j} \omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j)$ et une mesure d'**information intégrée**. Quels prolongements théoriques imaginez-vous pour formaliser la "quantité d'intégration" qui pourrait soutenir une cognition plus "globale" ?

Ces **cinq problèmes** offrent un panorama de questionnements mathématiques autour de la **section 1.3** (Importance de l'Auto-Organisation) : la **dynamique** des pondérations (Problème 1), l'**émergence** et la **clusterisation** (Problème 2), l'équilibre subtil **stabilité-plasticité** (Problème 3), la gestion de la **multi-modalité** et du **contexte** (Problème 4), et enfin une ouverture vers la **cognition distribuée** (Problème 5). Aucun calcul numérique n'est exigé ; il s'agit exclusivement de **questions formelles**, destinées à approfondir la compréhension mathématique de l'auto-organisation dans le cadre du **Deep Synergy Learning**.

Appendix 1

Étude mathématique du système dynamique et résolution de l'équation d'équilibre

L'équation donnée :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

représente un système dynamique discret, où les pondérations $\omega_{i,j}(t)$ évoluent sous l'influence des synergies $S(\mathcal{E}_i, \mathcal{E}_j)$, du taux d'apprentissage $\eta > 0$, et du coefficient de décroissance $\tau > 0$.

1. Condition d'équilibre

À l'équilibre, les pondérations cessent de changer, ce qui implique que :

$$\omega_{i,j}(t+1) - \omega_{i,j}(t) = 0.$$

Substituons cette condition dans l'équation dynamique :

$$0 = \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

En simplifiant, on obtient :

$$\omega_{i,j}^* = \frac{S(\mathcal{E}_i, \mathcal{E}_j)}{\tau},$$

où $\omega_{i,j}^*$ représente la pondération à l'équilibre.

2. Stabilité de l'équilibre

Pour étudier la **stabilité** de cet équilibre, nous examinons comment une perturbation autour de $\omega_{i,j}^*$ évolue.

a) Écart autour de l'équilibre

Posons :

$$\delta\omega_{i,j}(t) = \omega_{i,j}(t) - \omega_{i,j}^*.$$

En remplaçant $\omega_{i,j}(t)$ dans l'équation dynamique :

$$\delta\omega_{i,j}(t+1) = \delta\omega_{i,j}(t) - \eta\tau\delta\omega_{i,j}(t).$$

Cela se simplifie en :

$$\delta\omega_{i,j}(t+1) = (1 - \eta\tau)\delta\omega_{i,j}(t).$$

b) Condition de stabilité

Pour que $\delta\omega_{i,j}(t) \rightarrow 0$ (stabilité), il faut que :

$$|1 - \eta\tau| < 1.$$

Cela donne :

$$0 < \eta\tau < 2.$$

Ainsi, le système est **stable** si le produit $\eta\tau$ est dans l'intervalle $]0,2[$.

3. Cas des oscillations

Si $\eta\tau$ dépasse 2, le facteur $1 - \eta\tau$ devient négatif, ce qui entraîne une **oscillation** de $\delta\omega_{i,j}(t)$ autour de l'équilibre. Cela correspond à un comportement oscillatoire amorti ou amplifié selon les paramètres.

Oscillations amorties

Si $\eta\tau > 2$ mais reste modéré, les oscillations se réduisent progressivement et finissent par converger.

Oscillations non amorties

Si $\eta\tau \gg 2$, les oscillations peuvent persister indéfiniment ou même s'amplifier, menant à un régime instable.

4. Analyse globale avec la Jacobienne

La Jacobienne du système au point d'équilibre $\Omega^* = [\omega_{i,j}^*]$ s'écrit :

$$J_{i,j} = \frac{\partial}{\partial \omega_{k,l}} [\omega_{i,j}(t+1) - \omega_{i,j}(t)].$$

En calculant explicitement :

$$\frac{\partial}{\partial \omega_{k,l}} [\omega_{i,j}(t+1) - \omega_{i,j}(t)] = \begin{cases} -\eta\tau, & \text{si } (i,j) = (k,l), \\ 0, & \text{sinon.} \end{cases}$$

La Jacobienne est donc diagonale avec les termes $-\eta\tau$ sur la diagonale. La stabilité globale dépend des valeurs propres, qui sont $1 - \eta\tau$. Si toutes ces valeurs propres vérifient $|1 - \eta\tau| < 1$, alors le système est stable.

5. Simulation numérique (résolution discrète)

Paramètres de simulation :

- Nombre de nœuds : $N = 10$,
- Synergies $S(i,j)$: semi-aléatoires ($\sim \mathcal{U}(0,1)$),
- Taux d'apprentissage $\eta = 0.1$,
- Coefficient de décroissance $\tau = 1.0$.

Algorithme :

23. Initialiser $\omega_{i,j}(0) \sim \mathcal{U}(0,0.01)$,
24. Mettre à jour $\omega_{i,j}(t)$ à chaque itération selon l'équation dynamique,
25. Calculer $\delta\omega_{i,j}(t)$ pour vérifier la convergence.

Résultats attendus :

26. Convergence rapide vers $\omega_{i,j}^* \approx S(i,j)/\tau$ si $\eta\tau < 2$,
27. Oscillations visibles si $\eta\tau \geq 2$.

6. Applications et interprétation

- **Réseaux biologiques** : Modélisation des interactions dynamiques entre neurones ou molécules.
- **Systèmes physiques** : Étude des attracteurs dans les systèmes oscillants ou auto-organisés.
- **Réseaux artificiels** : Régulation des pondérations dans un réseau neuronal ou un DSL (Deep Synergy Learning).

Conclusion

L'étude de l'équation $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(E_i, E_j) - \tau\omega_{i,j}(t)]$ met en lumière des dynamiques d'auto-organisation riches et complexes. Elle montre comment les synergies et les paramètres de régulation influencent la

stabilité et les comportements oscillatoires, offrant des outils puissants pour comprendre et modéliser des systèmes complexes.

1.4. Architecture Générale du DSL

Dans les chapitres précédents, nous avons défini les **fondements conceptuels** du Deep Synergy Learning (DSL), en soulignant notamment l'importance de l'**auto-organisation** et des **synergies** entre entités d'information (sections 1.2 et 1.3). Nous savons désormais que chaque entité n'est pas qu'un simple vecteur de données : elle peut évoluer, interagir, et créer de nouvelles représentations lorsqu'elle coopère avec d'autres entités. Ces processus de coopération ne sont pas imposés par un schéma hiérarchique rigide, mais émergent librement à mesure que les entités détectent un **gain** dans leur association.

Afin de concrétiser ces principes dans un **cadre uniifié**, il est nécessaire de décrire l'**architecture générale** du DSL. Autrement dit, comment ces entités s'organisent-elles ? Quelles structures se dégagent quand la synergie est élevée entre certaines paires (ou ensembles) d'entités ? Et comment modéliser leur évolution au fil du temps ? La section 1.4 répond à ces questions en introduisant :

- Les **principes de base** des entités et des liens (1.4.1)
- La notion de **Synergistic Connection Network (SCN)** (1.4.2)
- La formation de **clusters** et de **macro-clusters** (1.4.3)
- Les différentes **fonctions de synergie** (1.4.4)
- Le fonctionnement **adaptatif** des pondérations (1.4.5)
- Les interactions **directes** et **indirectes** (1.4.6)
- Les cas particuliers de **synergie binaire** et **n-aire** (1.4.7)

C'est cette architecture globale du **DSL** qui soutient la plasticité et la capacité d'adaptation du système : plutôt que de transmettre passivement les données dans des couches, chaque entité peut, en temps réel, renforcer ou affaiblir ses liens avec d'autres entités, créer des regroupements spontanés (clusters), et faire émerger de nouvelles représentations plus riches que la somme de leurs composantes individuelles.

1.4.1. Principe de Base : Entités et Liens Synergiques

Le **Deep Synergy Learning** envisage chaque **entité d'information** comme un **nœud actif** de l'architecture globale. Plutôt que de propager passivement un signal, comme le ferait un neurone dans un réseau hiérarchique, cette entité adopte un rôle décisif en disposant de **paramètres internes** θ_k , souvent constitués d'une **représentation** \mathbf{x}_k (vectorielle ou tensorielle), d'un **état interne** \mathbf{s}_k qui se modifie au gré des interactions, et de tout **hyperparamètre** requis pour la modalité concernée (par exemple un encodeur audio ou un embedding textuel). Chaque entité s'accompagne également de **mécanismes de décision** qui lui permettent d'**observer** la synergie obtenue lorsqu'elle se connecte à d'autres entités et, selon cette observation, de renforcer, d'affaiblir ou de rompre certains liens. Elle peut même, dans les cas où la coopération se révèle durablement élevée, envisager une **fusion** avec une autre entité ou, au contraire, une **spécialisation** en segmentant ses propres paramètres pour répondre à des tâches spécifiques. Cette description, encore préliminaire, fait déjà ressortir l'idée que les **nœuds** du DSL sont bien plus dynamiques que dans un réseau de neurones classique, puisque chacun "recherche" les coopérations les plus fructueuses pour améliorer sa représentation tout en contribuant à la robustesse du réseau dans son ensemble.

Chaque entité \mathcal{E}_i peut se connecter à une autre entité \mathcal{E}_j par le biais d'une **pondération synergique** $\omega_{i,j}(t)$, laquelle traduit la **valeur ajoutée** perçue dans cette association. Quand la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ est forte, on observe un accroissement de $\omega_{i,j}(t)$, tandis que dans le cas contraire, ce lien tend à se résorber puis à disparaître. L'évolution de $\omega_{i,j}$ se modélise alors par une équation simple, par exemple

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η désigne le taux d'apprentissage et τ un paramètre de régularisation. Cette formulation, exposée plus en profondeur dans les sections ultérieures, peut aussi s'enrichir de seuils pour limiter la croissance des poids ou supprimer les connexions trop faibles, assurant de la sorte une **régulation** de la densité du graphe. Une fois que l'on modélise ainsi les **liens synergiques**, on se dote d'une **matrice d'adjacence** évolutive qui dicte quels nœuds interagissent, quand ils le font et avec quelle intensité.

La grande différence avec une organisation hiérarchique est que l'information n'est plus obligée de transiter selon des voies prédéfinies. Toute entité peut potentiellement échanger avec n'importe quelle autre, pour peu que la **pondération synergique** atteigne un niveau jugé satisfaisant. Dans le cadre d'une tâche multimodale, il est donc envisageable que des entités "visuelles" se lient directement à des entités "textuelles" si elles y trouvent un bénéfice mutuel. Il se forme alors des **clusters** éphémères ou plus durables, réunissant deux, trois ou davantage d'entités, et la persistance de ces regroupements dépend de la permanence de leur synergie. De tels micro-réseaux se constituent et se défont librement, donnant lieu à une **auto-organisation** dont l'architecture n'a pas été spécifiée a priori mais découle de la dynamique interne du système.

Cette perspective se décrit volontiers comme un **graphe évolutif** $G(t)$. Les **nœuds** correspondent aux entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$, et les **arêtes** représentent les liaisons $\omega_{i,j}(t)$. À chaque itération, une règle d'actualisation \mathcal{U} recalcule $\omega_{i,j}(t+1)$ en tenant compte de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ et de la **régularisation** (qui inclut, au besoin, la suppression des liens faibles). Cela rend possibles des analyses croisées avec la **théorie des graphes** ou la **dynamique non linéaire**, et l'on peut même faire appel à des méthodologies d'**optimisation combinatoire** lorsqu'on cherche, par exemple, à minimiser un coût global ou à identifier une configuration de connexions correspondant à un optimum local.

Cette approche basée sur des **nœuds actifs** et des **arêtes dynamiques** confère au **DSL** une **flexibilité** qui se révèle supérieure à celle de nombreuses architectures traditionnelles. La topologie n'est plus figée dans des couches de traitement ; elle se réorganise au gré des opportunités de synergie détectées, ce qui est d'un intérêt crucial pour les situations multimodales ou pour l'adaptation à des données en évolution. Les entités elles-mêmes peuvent fusionner, se subdiviser ou changer de paramétrage, et toute la structure s'oriente vers une organisation où la synergie la plus forte est mise en avant. Cette logique favorise l'apparition de **propriétés émergentes** comme la consolidation de **macro-clusters** ou l'élaboration de schémas coopératifs inédits, sans qu'aucune hiérarchie ne soit imposée dès le départ.

En somme, la clef de voûte du **DSL** réside dans l'idée que chaque entité constitue un **nœud autonome**, et chaque connexion, un **lien synergique** dont l'évolution est régie par le degré d'enrichissement mutuel. Les observations de clusters, de reconnections ou de regroupements imprévus y sont donc la norme, et ce caractère **distribué** et **adaptatif** positionne le **DSL** comme un prolongement original des paradigmes existants, taillé pour gérer la variété et la complexité croissantes des données actuelles.

1.4.2. Présentation du Synergistic Connection Network (SCN)

Le **Synergistic Connection Network (SCN)** constitue la **pièce maîtresse** du Deep Synergy Learning (DSL) en matérialisant à la fois l'**espace** dans lequel les entités évoluent et le **mécanisme** même qui autorise la naissance, la transformation ou la disparition de leurs liens. Alors que les réseaux de neurones classiques reposent sur une **topologie fixée** avant l'entraînement, le SCN propose, au contraire, une **structure adaptive** dont les entités et leurs connexions se reconfigurent de manière autonome au fil du temps.

La représentation du SCN comme **graphe** place les **entités** $\{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n\}$ en tant que **nœuds**, connectés par des **pondérations synergiques** $\omega_{i,j}$ capables d'évoluer : chaque entité correspond à un bloc fonctionnel (par exemple visuel, textuel ou auditif) et peut, en fonction de la **synergie** détectée, renforcer ou, au contraire, réduire sa liaison avec une autre entité. Cette plasticité confère au SCN un caractère **vivant** : la structure n'est jamais figée, mais se **recompose** continuellement, en conservant les connexions jugées utiles et en éliminant les autres.

A. Les principes fondateurs du SCN

Coopération locale, cohérence globale

Au cœur du SCN, chaque liaison $\omega_{i,j}$ se modifie selon une **décision locale** : deux entités évaluent la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ — qu'elle s'appuie sur une similarité, une co-information ou un gain de performance — puis ajustent $\omega_{i,j}$. Pourtant, cette dynamique locale affecte la **structure globale** : en synchronisant leurs mises à jour, des milliers ou des millions de connexions peuvent, en quelques itérations, donner naissance à des **macro-structures** ou des **clusters** (voir section 1.4.3).

Adaptation continue

Plutôt qu'un apprentissage ponctuel, le SCN procède par **itérations** successives. Les pondérations $\omega_{i,j}$ se recalculent via une équation du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η indique le rythme de l'apprentissage et τ limite la persistance d'un lien. Cette **dynamique** empêche toute stagnation et confronte en permanence l'évolution des connexions aux signaux de synergie qui apparaissent ou disparaissent.

Mécanismes de régulation

Pour qu'un trop grand nombre de liaisons ne se maintienne artificiellement, on peut **couper** celles dont la pondération demeure en deçà d'un certain **seuil** ω_{\min} . À l'inverse, fixer un **plafond** ω_{\max} empêche l'explosion de liens exagérément forts. Ces stratégies contribuent à la **parsimonie** du réseau : en n'y retenant que les connexions bénéfiques, on favorise la formation de sous-ensembles stables et la mise en évidence de motifs structurels plus riches.

B. Du réseau neuronal traditionnel au SCN

Un **réseau de neurones classique** se caractérise par une **architecture pré définie** : chaque couche contient un nombre fixe de neurones, et les connexions linéaires entre couches demeurent invariables. Bien que l'on ajuste les poids internes via la rétropropagation du gradient, la **topologie** — nombre de couches, disposition des neurones, schéma des connexions — reste, elle, **immuable**. À l'inverse, le SCN instaure un paradigme radicalement différent :

28. **La topologie évolue** spontanément. Les connexions peuvent apparaître ou disparaître en fonction de la **synergie** détectée.
29. Les entités elles-mêmes sont **actives** : elles disposent d'un état interne ou de mécanismes de représentation susceptibles d'évoluer pour améliorer leur **coopération**.
30. Le réseau peut **se réinventer** au gré de l'arrivée de nouvelles données ou de changements contextuels : on voit apparaître puis disparaître des **clusters** (micro-réseaux spécialisés), sans qu'un algorithme extérieur ne doive intervenir pour redéfinir la structure.

Avec cette **plasticité**, le SCN parvient à incorporer et à privilégier les synergies révélées par les données, adaptant la répartition des connexions bien plus librement qu'un réseau hiérarchique pré déterminé.

C. Domaines d'application du SCN

31. **Multimodalité fluide**. En associant chaque modalité (vision, texte, audio, capteurs) à des entités spécialisées, le SCN crée et supprime des liaisons au gré de la **valeur ajoutée** qu'elles apportent. Les entités "visuelles" peuvent ainsi s'allier directement à des entités "textuelles" si elles trouvent un **gain mutuel**.
32. **Apprentissage continu et évolutif**. Dans un environnement changeant, le SCN **réajuste** sa configuration en continu, renforçant les liens utiles, délaissant les autres. Cette plasticité itérative apporte une **résilience** accrue, évitant tout réapprentissage intégral au moindre écart de distribution.

33. **Découverte de patrons “n-aires”.** Au-delà de l’interaction par paires, le SCN encourage la formation de **macro-clusters** quand plusieurs entités $\{\mathcal{E}_1, \dots, \mathcal{E}_m\}$ coopèrent de manière à engendrer une plus-value collective supérieure à la somme de leurs synergies binaires.

Conclusion

Le **Synergistic Connection Network (SCN)** illustre la **vision** fondamentale du DSL : un réseau **auto-organisé, évolutif**, où chaque entité se connecte, se réoriente ou se scinde suivant la **synergie** qu’elle détecte autour d’elle. À l’inverse d’un canevas hiérarchique traditionnel, cette organisation se déploie sans prescriptions initiales : elle se **bâtit** et se **rebâtit** au gré des liens les plus productifs, aboutissant à l’apparition de **clusters** et de **macro-structures** cohérentes, façonnées par l’apprentissage lui-même. Dans ce paradigme, l’interaction réciproque et la mise en commun de l’information prennent le pas sur la simple transmission d’un signal, créant une flexibilité particulièrement adaptée à la complexité et à la dynamique des données actuelles.

Dans la section suivante (1.4.3), nous verrons comment ces liens synergiques favorisent la naissance de **clusters auto-organisés**, et en quoi ceux-ci peuvent se regrouper en **macro-clusters** pour donner naissance à des entités d’information plus puissantes, ou plus abstraites, dans la démarche d’apprentissage distribué du Deep Synergy Learning.

1.4.3. Notion de Cluster et de Macro-Cluster

Dans le **Synergistic Connection Network (SCN)**, décrit à la sous-section précédente (1.4.2), chaque entité d’information \mathcal{E}_i est reliée aux autres par des **pondérations synergiques** $\omega_{i,j}(t)$ qui évoluent dans le temps. Ce mécanisme autorise l’**émergence spontanée** de sous-structures au sein du réseau : des groupes d’entités qui présentent entre elles une synergie élevée et tendent à **collaborer** plus fréquemment. On appelle couramment ces regroupements des **clusters**.

Au-delà de ces regroupements de base, il est possible que **plusieurs** clusters se regroupent encore à un niveau supérieur pour former des **macro-clusters**. Cette section (1.4.3) approfondit la *définition mathématique* de ces concepts, leurs *règles d’apparition* et leurs *conséquences* sur la dynamique d’apprentissage dans le **Deep Synergy Learning (DSL)**.

Dans la perspective du **Synergistic Connection Network (SCN)**, il est souvent utile de modéliser la structure courante du réseau par un **graphe** $G(t)$ dont les **sommets** $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ correspondent aux entités d’information, et dont chaque **arête** est dotée d’une **pondération synergique** $\omega_{i,j}(t)$. À l’instant t , ces pondérations se réunissent dans une **matrice** :

$$W(t) = [\omega_{i,j}(t)]_{1 \leq i,j \leq n}.$$

Cette représentation matricielle capture simultanément l’ensemble des liens du réseau, chaque $\omega_{i,j}(t)$ traduisant l’intensité de la coopération entre l’entité \mathcal{E}_i et l’entité \mathcal{E}_j . Dans un tel graphe, un **cluster** $\mathcal{C} \subset \{1, \dots, n\}$ (c’est-à-dire un sous-ensemble des indices d’entités) se caractérise par des **connexions internes** plus **denses** ou plus **fortes** que ses connexions avec l’extérieur. Plusieurs manières existent pour formaliser cette notion de cluster :

Un **premier** point de vue consiste à **maximiser la somme** des pondérations internes. Plus précisément, on cherche un sous-ensemble \mathcal{C} maximisant

$$\sum_{i \in \mathcal{C}, j \in \mathcal{C}} \omega_{i,j}(t),$$

ce qui revient à rechercher la zone du graphe possédant la **densité** la plus élevée en liaisons fortes. Cette démarche peut être utile, par exemple, lorsque l’on souhaite mettre en évidence un noyau collaboratif de grande intensité.

Un **deuxième** critère, plus nuancé, repose sur le **rapport** entre la force interne et la force externe. On introduit alors une fonction de ratio

$$R(\mathcal{C}) = \frac{\sum_{i,j \in \mathcal{C}} \omega_{i,j}(t)}{\sum_{i \in \mathcal{C}, j \notin \mathcal{C}} \omega_{i,j}(t) + \epsilon},$$

où $\epsilon > 0$ agit comme un **terme de régularisation** évitant les divisions par zéro. Le cluster \mathcal{C} optimal maximise alors $R(\mathcal{C})$, ce qui revient à privilégier des groupes dont la **cohésion interne** se révèle importante face aux connexions dirigées vers l'extérieur.

Enfin, un **troisième** point de vue fait appel à une **fonction d'énergie** \mathcal{J} , ou fonction de coût, qui **récompense** la densité intra-cluster tout en **pénalisant** les connexions externes (cf. section 1.4.2 pour une présentation plus générale du concept de coût dans le DSL). Les **clusters** apparaissent alors comme des **minima locaux** de \mathcal{J} . Cette perspective offre un cadre théorique aisément connectable à la physique statistique ou aux techniques d'optimisation combinatoire, permettant d'étudier la **stabilité** ou le **caractère** global/local des solutions.

Par-delà la variété de ces approches, le principe fondamental demeure : un **cluster** est un **groupe d'entités** qui, au sein de la matrice $W(t)$, entretient des liaisons relativement plus fortes entre ses membres qu'avec le reste du réseau. On retrouve ainsi l'idée intuitive d'une **collaboration** accrue à l'intérieur du sous-ensemble, associée à un **isolement** ou à une **différenciation** face aux entités extérieures. Dans la suite du texte, ces définitions de clusters s'avèrent essentielles pour analyser la manière dont le **SCN** laisse émerger automatiquement, au fil de la dynamique d'apprentissage, des **strates** coopératives plus ou moins vastes.

Dans le cadre du **Deep Synergy Learning**, les pondérations $\omega_{i,j}(t)$ évoluent fréquemment selon la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ et $\tau > 0$ sont des constantes, tandis que $S(\mathcal{E}_i, \mathcal{E}_j)$ désigne la **synergie** entre les entités i et j . Dès lors qu'une pondération $\omega_{i,j}(t)$ dépasse un seuil ω_{\min} , on considère qu'il existe un **lien effectif** entre les deux entités. Plusieurs phénomènes concourent alors à la **formation d'un cluster** :

- **Renforcement interne.** Lorsqu'un certain **groupe** $\mathcal{C} \subset \{1, \dots, n\}$ présente des synergies élevées entre ses membres, les pondérations $\omega_{i,j}(t)$ associées ont tendance à **augmenter** pour tous $i, j \in \mathcal{C}$.
- **Isolement progressif.** À mesure que les liens internes se consolident, les connexions externes (i.e. celles dont la synergie avec les entités extérieures demeure plus faible) s'**atténuent**, rendant le sous-ensemble \mathcal{C} de plus en plus autonome et homogène.

Ce processus engendre un **cluster** cohérent, formé sans qu'aucun agencement préalable ne soit imposé.

Exemple de scénario d'émergence

34. **Temps initial (t=0).** Les pondérations $\omega_{i,j}(0)$ sont faibles ou distribuées de manière uniforme, si bien qu'aucune structure évidente ne se détache.
35. **Interactions locales.** Certaines entités détectent une synergie ($S(\mathcal{E}_i, \mathcal{E}_j) > 0$) suffisamment avantageuse pour se renforcer localement. Les $\omega_{i,j}(t)$ correspondantes s'amplifient.
36. **Formation d'une "graine".** Progressivement, un **mini-cluster** naît sous la forme de 2 ou 3 entités dont les liens internes, devenus plus élevés, assurent un noyau de haute synergie.
37. **Attraction.** Si une entité \mathcal{E}_k à la périphérie présente aussi une bonne synergie avec cette graine, ses liaisons $\omega_{k,i}$ ($i \in \mathcal{C}$) se renforcent, ce qui intègre \mathcal{E}_k au cluster en expansion.
38. **Consolidation.** Une fois que les **liens internes** atteignent un niveau stable et que les connexions vers l'extérieur restent faibles, le groupe \mathcal{C} se **stabilise** comme cluster. D'un point de vue formel, cela correspond à un **minimum local** d'une fonction d'énergie \mathcal{J} , où \mathcal{C} se comporte comme un **attracteur**.

Au-delà des **clusters** de base, il n'est pas rare que plusieurs sous-groupes finis se **rassemblent** pour former un **macro-cluster**, c'est-à-dire une entité supérieure englobant plusieurs clusters déjà constitués.

Pour cela, on considère un **méta-groupe** \mathcal{M} rassemblant des **clusters** $\mathcal{C}_1, \mathcal{C}_2, \dots$. Les pondérations entre deux clusters \mathcal{C}_1 et \mathcal{C}_2 se définissent par

$$\Omega(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{|\mathcal{C}_1| \cdot |\mathcal{C}_2|} \sum_{\substack{i \in \mathcal{C}_1 \\ j \in \mathcal{C}_2}} \omega_{i,j}(t).$$

Si $\Omega(\mathcal{C}_1, \mathcal{C}_2)$ dépasse un certain niveau, les deux groupes **fusionnent**, produisant ainsi un **macro-cluster** élargi.

Processus de fusion

- 39. **Clusters initiaux.** Supposons que \mathcal{C}_1 et \mathcal{C}_2 soient déjà établis.
- 40. **Renforcement inter-clusters.** Dès lors que plusieurs liaisons $\omega_{i,j}$ ($i \in \mathcal{C}_1, j \in \mathcal{C}_2$) se consolident, la synergie globale entre \mathcal{C}_1 et \mathcal{C}_2 augmente.
- 41. **Fusion structurelle.** Sitôt ce renforcement assez prononcé pour franchir un **seuil**, les deux clusters cessent de constituer des sous-groupes autonomes : ils forment un **macro-cluster** \mathcal{M} .
- 42. **Vue hiérarchique.** Cette démarche rappelle le **clustering hiérarchique agglomératif**, à la différence majeure que, dans le DSL, les pondérations $\omega_{i,j}$ ne sont pas statiques mais se **règlent** en permanence, autorisant une consolidation plus dynamique.

Pour formaliser ces phénomènes de clusterisation, on introduit couramment une **fonction d'énergie** \mathcal{J} :

$$\mathcal{J}(\{\omega_{i,j}\}) = - \sum_{(i,j)} \omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j) + \alpha \sum_{(i,j)} (\omega_{i,j})^2 + \dots$$

- Le premier terme **récompense** les liaisons de haute synergie : chercher à **minimiser** \mathcal{J} revient donc à **maximiser** la somme $\sum \omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j)$.
- Le second terme (avec $\alpha > 0$) introduit une forme de **régularisation** ou de “penalty” pour empêcher une croissance excessive et simultanée de tous les liens.

Dans ce cadre, un **cluster** renvoie à un **minimum local** de \mathcal{J} , où l'on observe un niveau élevé de pondérations **en interne**. Un **macro-cluster** surgit quand on identifie un autre minimum local, rassemblant plusieurs groupes préexistants.

L'émergence de **clusters** segmente le Synergistic Connection Network en **régions** spécialisées (p. ex. un cluster associant vision et texte, un autre combinant audio et capteurs, etc.). On obtient ainsi une forme de **division du travail** au sein d'un réseau de grande taille, ce qui favorise la **lisibilité** et la **résilience** globale.

Par ailleurs, la formation de **macro-clusters** peut être rapprochée de l'idée de **modules cognitifs**, dans lesquels plusieurs sous-ensembles d'entités se lient pour s'attaquer à des tâches plus complexes. Le réseau acquiert de la sorte une **modularité adaptative**, qu'il s'agisse de manipuler différents canaux de données ou d'élaborer des représentations de plus haut niveau.

D'un point de vue **algorithmique**, détecter et gérer explicitement les clusters permet de segmenter la **mise à jour** des pondérations : plutôt que d'analyser un vaste graphe dans sa globalité, on peut restreindre certains calculs à des modules faiblement connectés entre eux. Cette **auto-organisation** en groupes soudés se démarque d'un simple “clustering statique”, car les pondérations $\omega_{i,j}(t)$ évoluent continuellement, autorisant des scissions, des fusions et même la disparition de clusters entiers au gré des fluctuations de la synergie.

Exemples d'application

- **Analyse multimodale en temps réel.** Dans une application de surveillance, on peut regrouper en un même cluster les entités traitant l'image et la parole (cluster “visage + discours”), et dans un autre, celles dédiées à

la détection de mouvements anormaux ou de sons inhabituels. Les agrégats évoluent de manière dynamique selon la synergie perçue entre différents flux (lumière, son, déplacements, etc.).

- **Recommandation et filtrage collaboratif.** En représentant les utilisateurs et les contenus par des entités, les connexions $\omega_{u,c}$ sont élevées quand un utilisateur u apprécie un contenu c . La dynamique du réseau provoque l'émergence de **clusters** d'utilisateurs ayant des goûts proches et de contenus similaires ; un **macro-cluster** peut alors regrouper un large groupe d'utilisateurs et l'ensemble des contenus qui leur plaisent collectivement.
- **Traitement biologique ou neuroscientifique.** Si l'on assimile les neurones (ou de petites populations neuronales) à des entités \mathcal{E}_i , on peut observer la formation de **clusters** assimilables à des assemblées neuronales locales, puis la mise en place de **macro-clusters** associant plusieurs aires cérébrales interconnectées. Ceci traduit une **organisation fonctionnelle** plus étendue de la dynamique neuronale.

Conclusion

La **formation de clusters** (puis de **macro-clusters**) incarne un des atouts majeurs du **DSL** : le réseau peut **s'auto-structurer** en sous-ensembles flexibles, chacune de ces composantes se stabilisant ou fusionnant selon la synergie qui émerge dans le temps. Surtout, il ne s'agit pas d'une simple classification unique et immuable : les pondérations $\omega_{i,j}(t)$ continuent de se réévaluer, donnant lieu à des réorganisations spontanées.

Dans la suite (sections 1.4.4 à 1.4.7), nous examinerons les bases mathématiques qui sous-tendent ce phénomène, en décrivant la **fonction de synergie** (distance, similarité ou information mutuelle), la façon dont les pondérations $\omega_{i,j}$ s'adaptent dans le temps, la distinction entre **interactions directes** et **indirectes**, puis la question de la **synergie n-aire**, où plusieurs entités coopèrent simultanément au-delà des simples relations binaires.

1.4.4. Fonctions de Synergie : Distance, Similarité et Co-Information

Au cœur du **Deep Synergy Learning (DSL)**, les *pondérations synergiques* $\omega_{i,j}$ reliant deux entités \mathcal{E}_i et \mathcal{E}_j évoluent en fonction d'une **mesure de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Cette fonction S évalue dans quelle mesure la **coopération** entre deux entités apporte un **gain** supérieur à l'utilisation isolée de leurs informations.

Bien que le **principe** de la synergie demeure le même (capturer l'apport mutuel), il existe **plusieurs manières** de la **définir** et de la **calculer** dans la pratique. Les plus communes reposent sur (1) des **distances** entre représentations, (2) des **similarités** (souvent normalisées), ou (3) des mesures d'**information** (entropie, co-information, etc.). Cette section expose en détail ces approches, avec une formulation mathématique approfondie.

Dans le **Deep Synergy Learning (DSL)**, la pondération $\omega_{i,j}(t)$ reliant deux entités \mathcal{E}_i et \mathcal{E}_j suit fréquemment une équation de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η et τ sont des hyperparamètres positifs, et $S(\mathcal{E}_i, \mathcal{E}_j)$ la **fonction de synergie**. Cette fonction reflète dans quelle mesure les entités \mathcal{E}_i et \mathcal{E}_j estiment pouvoir coopérer avantageusement. Dès lors,

- Si $S(\mathcal{E}_i, \mathcal{E}_j)$ est **positif** et assez grand, la liaison $\omega_{i,j}$ se **renforce**.
- Si $S(\mathcal{E}_i, \mathcal{E}_j)$ est faible ou négatif, la liaison s'**affaiblit**.

Le **choix** de la fonction S influe directement sur la **topologie** finale du réseau : deux entités jugées proches ou complémentaires se retrouveront dans un même **cluster** (voir section 1.4.3), tandis que des entités peu compatibles verront leur lien s'étioler. Différentes approches existent pour définir cette synergie.

1. Synergie fondée sur la distance

Une façon intuitive d'instaurer la synergie est de partir d'une **distance** $d(\mathcal{E}_i, \mathcal{E}_j)$ calculée sur leurs représentations, souvent des vecteurs $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$. Un exemple répandu consiste en la **distance euclidienne** $\|\mathbf{x}_i - \mathbf{x}_j\|$. On convertit ensuite cette distance en un score de synergie **décroissant** :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \frac{1}{1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2},$$

ou bien, en autorisant des valeurs négatives, on peut poser

$$S(\mathcal{E}_i, \mathcal{E}_j) = -\|\mathbf{x}_i - \mathbf{x}_j\|.$$

L'idée est qu'une faible distance se traduit par une **forte synergie**, tandis qu'une distance importante produit un score négatif ou tendant vers zéro. Il est fréquent d'ajouter un **paramètre d'échelle** $\sigma > 0$ pour réguler la sensibilité aux écarts, par exemple

$$S(\mathcal{E}_i, \mathcal{E}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2 / 2\sigma^2).$$

Cette fonction valorise la synergie pour les entités **très proches** et la fait décroître rapidement au-delà d'un rayon σ . La distance choisie dépend ensuite de la nature des données : L1, L2, distance sur des tenseurs d'images ou spectrogrammes, etc.

2. Synergie fondée sur la similarité

Un autre choix repose sur une **similarité** plutôt que sur une distance. Par exemple, la **similarité cosinus** :

$$\text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\langle \mathbf{x}_i, \mathbf{x}_j \rangle}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

qui varie dans $[-1, 1]$. On peut alors définir

$$S(\mathcal{E}_i, \mathcal{E}_j) = \max(0, \text{sim}(\mathbf{x}_i, \mathbf{x}_j)) \quad \text{ou} \quad S(\mathcal{E}_i, \mathcal{E}_j) = 1/2 (\text{sim}(\mathbf{x}_i, \mathbf{x}_j) + 1),$$

histoire de normaliser la valeur dans $[0, 1]$. Cette approche est pratique quand on souhaite regrouper des **vecteurs** se ressemblant en direction ou quand on manipule des **corrélations** (ex. coefficient de Pearson). Elle est parfois moins adaptée si l'on cherche à capter la complémentarité non linéaire entre des entités fortement dissemblables.

3. Synergie fondée sur la co-information ou l'information mutuelle

Lorsque les entités $\mathcal{E}_i, \mathcal{E}_j$ renvoient à des **variables aléatoires** $\mathbf{X}_i, \mathbf{X}_j$, on peut évaluer la synergie via des **quantités entropiques** :

$$I(\mathbf{X}_i; \mathbf{X}_j) = H(\mathbf{X}_i) + H(\mathbf{X}_j) - H(\mathbf{X}_i, \mathbf{X}_j),$$

qui mesure l'**information mutuelle**. Toutefois, cette mesure ne différencie pas la simple **redondance** d'une véritable **coopération**. On peut alors recourir à la **co-information** ou à la **Partial Information Decomposition (PID)**, laquelle décompose l'information partagée en portions "synergique" et "redondante". Cela donne un score :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \max(0, I_{\text{syn}}(\mathbf{X}_i, \mathbf{X}_j | \mathbf{Y})),$$

où I_{syn} désigne la contribution strictement synergique, au sens de la définition employée (co-info, PID, etc.). Cette méthode peut détecter des interactions non linéaires et complexes, mais elle se révèle plus **onéreuse** à calculer, en particulier pour des données de haute dimension, et nécessite de **choisir** la mesure d'information la plus pertinente au contexte.

Conseils pour le choix de la fonction S

Le **DSL** ne prescrit pas une forme unique de synergie, mais un **cadre** flexible où n'importe quelle mesure d'**apport mutuel** entre entités peut convenir. On retient néanmoins certains principes :

43. **Distance.** Simple à implémenter et intuitive, elle va favoriser l'association d'entités similaires (au sens d'un espace de représentation). Elle sous-estime parfois la complémentarité de deux entités très différentes mais hautement coopératives.
44. **Similarité.** Plus adaptée dès lors qu'on compare des directions ou des vecteurs déjà normalisés. Les grandes similarités indiquent une forte parenté, mais la complémentarité non linéaire peut lui échapper.
45. **Information mutuelle / co-info / PID.** Très puissante pour les **coopérations** complexes et non linéaires, mais coûteuse en ressources de calcul et sensible aux méthodes d'estimation statistique.

Il est évidemment possible de **combiner** plusieurs indicateurs (distance, similarité, information mutuelle) ou de pondérer différentes composantes pour confectionner un score $S(\cdot, \cdot)$ plus adapté. Le choix exact dépendra de la nature des données, du domaine d'application et des **objectifs** poursuivis (clustering, prédiction, fusion multimodale, etc.).

Exemples d'utilisation

46. **Analyse d'images.** Considérons des entités \mathcal{E}_i représentant des patchs ou des régions d'images. Une distance L2 sur des vecteurs de caractéristiques (par ex. issus d'un réseau de neurones convolutionnel) peut fournir un score de synergie décroissant : plus les patchs se ressemblent, plus ils coopèrent.
47. **Fusion multimodale.** Supposons deux entités $\mathbf{X}_{\text{audio}}$ et $\mathbf{X}_{\text{visuel}}$. On mesure leur co-information conditionnellement à une variable Y (classe d'événement). Si leur combinaison apporte un gain clair, on augmente la synergie et, par conséquent, on renforce leur lien.
48. **Système hybride symbolique-connexionniste.** Un module symbolique (décrivant des règles ou faits abstraits) et un module vectoriel (des embeddings d'images, par exemple) peuvent relier leurs entités si l'information mutuelle ou la similarité cosinus démontre qu'ils s'améliorent réciproquement.

Conclusion et ouverture

La fonction de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ est la **pierre angulaire** du Deep Synergy Learning. Selon qu'on s'appuie sur une **distance**, une **similarité** ou une **information mutuelle**, on oriente la manière dont les entités **coopèrent** et se **relient** au sein du **Synergistic Connection Network**.

- Les **fonctions de distance** favorisent l'agrégation d'éléments proches dans l'espace des caractéristiques.
- Les **fonctions de similarité** rassemblent des vecteurs corrélés, tout en ignorant parfois la complémentarité.
- Les **mesures entropiques** (co-info, PID, etc.) explorent la coopération profonde, prenant en compte les facettes non linéaires et la contribution uniquement révélée par l'association des entités.

Les sections suivantes (1.4.5 à 1.4.7) approfondiront la **mise à jour temporelle** de $\omega_{i,j}(t)$ fondée sur cette synergie, la distinction entre **interactions directes** et **indirectes**, et la manière de gérer la **synergie n-aire**. L'ensemble de ces points complète la **vision** du DSL comme un réseau évolutif, autonome et apte à faire émerger des **structures** d'apprentissage complexe dans des contextes variés.

1.4.5. Pondérations Adaptatives et Évolution Temporelle

Les sections précédentes (1.4.1 à 1.4.4) ont établi les **fondements** de l'architecture générale du **Deep Synergy Learning (DSL)** : chaque **entité** peut se relier à d'autres par des **pondérations synergiques** $\omega_{i,j}$ qui reflètent la **valeur ajoutée** d'une coopération. Reste à décrire **comment** ces pondérations **changent** dans le temps, c'est-à-dire la **loi**

d'**évolution** qui fait que le réseau se **reconfigure** en permanence. C'est l'objet de cette sous-section (1.4.5), qui explique :

- Les **équations** gouvernant l'adaptation des pondérations,
- Le sens de cette **dynamique** (renforcement / affaiblissement),
- Les implications sur la **convergence** ou la **stabilité** des liens,
- Le rôle des **paramètres** (taux d'apprentissage, régularisation, etc.) dans l'auto-organisation du réseau.

Cette **évolution temporelle** est au cœur du **SCN** (Synergistic Connection Network), car c'est elle qui autorise la création progressive de **clusters** et, par extension, la modélisation d'un **apprentissage** réellement continu.

Dans la plupart des formulations du **Deep Synergy Learning (DSL)**, la pondération $\omega_{i,j}(t)$ reliant deux entités \mathcal{E}_i et \mathcal{E}_j se met à jour suivant un schéma adaptatif de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η désigne un **taux d'apprentissage** (ou **pas de mise à jour**) et τ un **coefficent de régulation** (ou *terme d'oubli*). La fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ correspond à la mesure de **synergie** choisie (distance, similarité, co-information, etc.). Ce cadre de mise à jour s'interprète ainsi :

- Lorsque la synergie S entre \mathcal{E}_i et \mathcal{E}_j est assez élevée, et que $\omega_{i,j}(t)$ reste encore modérée, le terme $\eta [S - \tau \omega]$ demeure **positif**, accroissant la liaison à chaque itération.
- À l'inverse, si la synergie est faible (ou négative), ou si $\omega_{i,j}(t)$ est déjà trop grande, le terme devient **négatif**, ce qui affaiblit la liaison.

On peut mettre en évidence un **point fixe** en imposant $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$. La condition associée,

$$\omega_{i,j}^* = S/\tau,$$

montre que plus la **synergie** est grande, plus la **pondération d'équilibre** $\omega_{i,j}^*$ atteint un niveau élevé.

Variantes et enrichissements

Plusieurs adaptations sont possibles autour de cette règle. On peut retenir un **historique** afin de lisser la dynamique ou pondérer les valeurs antérieures $\omega_{i,j}(t-1)$. La forme de mise à jour elle-même peut être **non linéaire** : on quitte la version linéaire $\omega + \eta [S - \tau \omega]$ pour des équations saturantes, exponentielles ou inspirées des règles "hebbiennes". De plus, la synergie S peut varier avec le temps si les entités $\mathcal{E}_i(t)$ et $\mathcal{E}_j(t)$ mettent à jour leurs représentations internes ; la pondération $\omega_{i,j}(t)$ influence alors la synergie, qui en retour modifie la structure globale.

- pour les signes de qualité relevant de l'audit énergétique, un bilan des audits présentés comme référence ;
- pour les signes de qualité relevant de l'installation d'unités de production d'électricité photovoltaïque, un bilan des audits présentés comme référence ;
- le nombre d'audits effectués et une synthèse générale des résultats et écarts constatés et des sanctions appliquées à l'issue des contrôles ;

Système dynamique discret ou continu

Le modèle

$$\omega_{i,j}(t+1) - \omega_{i,j}(t) = \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

est une **équation aux différences**, interprétable comme un **système dynamique** discret dans un espace de dimension $\binom{n}{2}$ (ou $n(n-1)$) selon que l'on distingue la direction $i \rightarrow j$). On y analyse notamment :

49. Les **points fixes** $\omega_{i,j}^*$ satisfaisant

$$\omega_{i,j}^* = \omega_{i,j} + \eta [S - \tau \omega_{i,j}^*].$$

50. La **stabilité** de ces points. Par exemple, en linéarisant autour de l'équilibre, on obtient

$$\Delta\omega_{i,j}(t+1) = \Delta\omega_{i,j}(t) - \eta \tau \Delta\omega_{i,j}(t),$$

montrant que la condition $\eta \tau < 1$ est usuellement requise pour maintenir une évolution stable des pondérations.

L'équivalent **en temps continu** est donné par l'ODE

$$d\omega_{i,j}/dt = \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Ici encore, la solution tend vers $\omega_{i,j}^* = S/\tau$ s'il ne survient pas d'autres perturbations ni dépendances supplémentaires.

Seuils, cap et parsimonie

Dans le **DSL**, on introduit souvent un **seuil** ω_{\min} en deçà duquel la connexion est jugée inexiste :

$$\omega_{i,j}(t) < \omega_{\min} \Rightarrow \text{aucun lien effectif entre } i \text{ et } j.$$

Même si l'équilibre $\omega_{i,j}^*$ est légèrement positif, la liaison doit dépasser ω_{\min} pour être considérée. À l'autre extrémité, on peut imposer un **cap** $\omega_{i,j}(t) \leq \omega_{\max}$ pour freiner l'augmentation des pondérations et éviter les valeurs extrêmes. Sur un **diagramme de phase** ($\omega-\dot{\omega}$), ces deux seuils garantissent une **parcimonie** structurale : seuls les liens ayant démontré une synergie conséquente survivent, tandis que ceux qui resteraient trop faibles ou monteraient exagérément se trouvent coupés ou plafonnés.

Interdépendance globale

Dans un réseau comptant n entités, la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ peut dépendre d'autres entités (cf. section 1.4.4). On écrit parfois

$$S(\mathcal{E}_i, \mathcal{E}_j \mid \{\mathcal{E}_k\}_{k \neq i,j}),$$

pour souligner le fait qu'une **co-information** conditionnelle peut influer sur la pondération. Les **mises à jour** des $\omega_{i,j}$ sont alors **couplées** : la liaison $i \leftrightarrow j$ se renforce ou s'affaiblit en interaction avec d'autres liaisons $j \leftrightarrow k$, etc. C'est ce couplage qui fait émerger des **clusters** (section 1.4.3) : les entités formant un sous-groupe \mathcal{C} stabilisent mutuellement leurs pondérations internes, aboutissant à une "bulle" de coopération élevée, tandis que les liens extérieurs, moins profitables, s'atténuent.

Feedback et plasticité

Interpréter la dynamique du DSL comme un système à **feedback** offre un éclairage pertinent :

51. **Feedback positif.** Dès lors qu'une liaison $\omega_{i,j}$ grandit (parce que $S > 0$), \mathcal{E}_i et \mathcal{E}_j accroissent encore leur coopération, consolidant le lien.

52. **Feedback négatif.** Si, au contraire, la liaison s’emballe, le terme $-\tau \omega_{i,j}(t)$ ou une autre pénalisation (par ex. un terme cubique) freine cette croissance, garantissant la **stabilisation**.

De façon analogique, on peut s’inspirer de la **plasticité synaptique** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left(a_i(t) a_j(t) \right) - \alpha \omega_{i,j}(t),$$

où $a_i(t)$ et $a_j(t)$ sont les “activités” ou “états” des entités, vus comme un substitut instantané à la synergie $S_{i,j}(t)$. Par ailleurs, l’usage d’un **terme cubique** $-\beta (\omega_{i,j})^3$ agit comme une stabilisation encore plus marquée, évitant la croissance sans borne.

Impacts et conclusions

53. **Flexibilité continue.** Grâce à l’actualisation en continu de $\omega_{i,j}(t)$, le **Synergistic Connection Network** (SCN) s’adapte sans cesse aux nouveaux flux ou aux modifications d’environnement, bien plus qu’un réseau à architecture fixe.
54. **Formation de clusters.** Les entités liées par une synergie forte consolident leurs liaisons, s’assemblent en **clusters**, et peuvent se **fusionner** en macro-clusters (voir section 1.4.3). Cette recomposition est fluide et dépend directement de la dynamique des pondérations.
55. **Régime stable vs. oscillatoire.** Selon les choix de synergie, de η , de τ , etc., le réseau peut tendre vers un état stable, osciller, ou coexister dans plusieurs configurations d’équilibre.
56. **Aspects computationnels.** Mettre à jour toutes les paires (i,j) exige un coût a priori en $O(n^2)$. Des heuristiques (sparse updates, random sampling) peuvent s’avérer nécessaires pour gérer de grands n de façon **scalable**.

Cette **dynamique adaptative** des pondérations constitue le moteur de l'**auto-organisation** dans le DSL, permettant de découvrir et de consolider des **sous-structures** au sein du réseau. Les sections suivantes (1.4.6 et 1.4.7) s’intéresseront à la **distinction entre interactions directes et indirectes** et à la **synergie n-aire**, deux aspects cruciaux pour comprendre la **richesse** et la **réactivité** de ce paradigme d’apprentissage.

1.4.6. Interactions Directes et Indirectes

Jusqu’à présent, la description du **Deep Synergy Learning (DSL)** s’est essentiellement focalisée sur la **relation directe** entre deux entités \mathcal{E}_i et \mathcal{E}_j , mesurée par une **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ et traduite dans une **pondération** $\omega_{i,j}$. Pourtant, dans un réseau où de nombreuses entités coexistent, les **interactions indirectes** jouent un rôle crucial : deux entités qui n’ont pas de liaison directe peuvent tout de même s’influencer si elles sont **connectées** via une **chaîne** (ou un **chemin**) de plusieurs liens synergiques. Cette section (1.4.6) explique :

- 57. Comment les **interactions** peuvent se propager à travers le **Synergistic Connection Network (SCN)**,
- 58. Comment la **synergie** entre deux entités peut être **modulée** par leurs relations avec d’autres,
- 59. Quelles implications mathématiques découlent de ces boucles d’influence plus complexes, notamment pour la **formation** de clusters et l’**émergence** de comportements globaux.

Cette problématique est centrale dans la compréhension de l'**auto-organisation** : même si chaque entité ne considère que les liens qui la concernent directement, l’ensemble du réseau peut manifester des effets d’**influence à distance** ou de **coordination** en chaîne.

On appelle **interaction indirecte** entre \mathcal{E}_i et \mathcal{E}_j toute séquence $\mathcal{E}_i \rightarrow \mathcal{E}_k \rightarrow \dots \rightarrow \mathcal{E}_m \rightarrow \mathcal{E}_j$ dans laquelle chacune des paires $(\mathcal{E}_u, \mathcal{E}_v)$ du chemin possède une pondération $\omega_{u,v}(t)$ significative. Le **Synergistic Connection Network** (SCN) autorise ainsi la propagation d'**influence** ou de **coopération** le long d'un **chemin** reliant deux entités, même si ces dernières ne sont pas directement connectées.

1. Principes généraux : chemins et relai d'information

Lorsque la liaison $\omega_{k,m}$ est élevée, l'entité \mathcal{E}_k peut **transmettre** (ou "relayer") certaines données ou incitations à \mathcal{E}_m . C'est ainsi qu'une synergie faible ou nulle entre \mathcal{E}_i et \mathcal{E}_j peut progressivement **s'intensifier** si elles interagissent via des entités communes $\{\mathcal{E}_k, \mathcal{E}_\ell, \dots\}$ qui procurent un **gain mutuel**. De même, si l'on modélise la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ comme une quantité **contextuelle**, on peut définir

$$S(\mathcal{E}_i, \mathcal{E}_j | \mathcal{E}_k),$$

qui tient compte du fait que la coopération entre \mathcal{E}_i et \mathcal{E}_j peut être **facilitée** ou **rendue moins nécessaire** par la présence d'une entité pivot \mathcal{E}_k .

Exemple : pivot et redondance

- Si \mathcal{E}_k remplit déjà les rôles combinés de \mathcal{E}_i et \mathcal{E}_j , l'intérêt pour \mathcal{E}_i et \mathcal{E}_j de se lier **directement** diminue (redondance).
- Au contraire, si \mathcal{E}_k fournit un complément utile à \mathcal{E}_i et \mathcal{E}_j , ce **contexte** peut augmenter leur synergie bilatérale, incitant l'émergence d'un lien direct $\omega_{i,j}$.

2. Notion de chemins et puissance de la matrice W

Le **graphe** du SCN se décrit par la matrice pondérée $W(t)$, de taille $n \times n$, dont l'entrée $\omega_{i,j}(t)$ indique la connexion directe entre \mathcal{E}_i et \mathcal{E}_j . Lorsqu'on considère un **chemin** de longueur 2 entre \mathcal{E}_i et \mathcal{E}_j via un intermédiaire \mathcal{E}_k , la multiplication de matrices révèle que $(W^2)_{i,j} = \sum_{k=1}^n \omega_{i,k}(t) \omega_{k,j}(t)$. De manière analogue, $(W^p)_{i,j}$ réunit l'effet de **tous** les chemins de longueur p liant i à j .

Si la dynamique des pondérations reste **relativement stable**, l'examen des puissances W^2, W^3, \dots peut dévoiler des **interactions indirectes** (chemins multiples ou plus longs). Des valeurs élevées de $(W^p)_{i,j}$ traduisent une **forte influence** de \mathcal{E}_i sur \mathcal{E}_j à travers plusieurs intermédiaires. En présence de **cycles** (ex. $\mathcal{E}_i \rightarrow \mathcal{E}_k \rightarrow \mathcal{E}_m \rightarrow \mathcal{E}_i$), on peut observer des effets d'**amplification** : si chaque liaison du cycle se renforce, cela accroît la synergie circulant en boucle.

D'un point de vue **linéaire**, la **valeur propre** de plus grande amplitude (rayon spectral $\rho(W)$) reflète la **stabilité** du système :

- Si $\rho(W)$ excède 1, de petites perturbations internes risquent de se propager et de s'amplifier (effet possible d'oscillation).
- Si $\rho(W) < 1$ (après régulation), la dynamique a tendance à converger vers un état stable.

3. Contexte et synergie conditionnelle

Dans un réseau synergique, l'existence de **chemins indirects** modifie considérablement la manière dont deux entités $\mathcal{E}_i, \mathcal{E}_j$ perçoivent leur **coopération**. Même si $\omega_{i,j}$ démarre faible, il peut se consolider via l'influence d'entités tierces \mathcal{E}_k . Formellement, on modélise cette dépendance par une fonction

$$S(\mathcal{E}_i, \mathcal{E}_j | \{\omega_{i,k}, \omega_{k,j}, \dots\}),$$

introduisant un **système dynamique non linéaire**. Les équations des pondérations deviennent

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j | \{\omega_{i,k}, \omega_{k,j}, \dots\}) - \tau \omega_{i,j}(t)].$$

La dépendance entre les $\omega_{i,j}$ fait naître des **boucles de feedback**, où un renforcement dans un segment du réseau rejaillit sur d'autres liaisons, et ainsi de suite. En pratique, l'**analyse** de stabilité ou de convergence peut impliquer des techniques avancées (théorie des bifurcations, méthodes de Lyapunov, etc.).

4. Rôle des interactions indirectes dans la dynamique globale

Les **interactions indirectes** expliquent comment un réseau DSL parvient à :

- **Faire émerger** de nouveaux liens : si \mathcal{E}_i et \mathcal{E}_j constatent qu'elles coopèrent déjà via un pivot \mathcal{E}_k , elles peuvent juger bon de forger une connexion directe $\omega_{i,j} > 0$.
- **Fusionner** des clusters : deux communautés initialement distinctes, reliées par quelques chemins transitifs, peuvent se reconnaître mutuellement profitables et **coalescer** en un macro-cluster.
- **Gérer** la circulation de bruit : une entité bruyante ou imprécise peut nuire indirectement à des voisins qui ne sont pourtant pas liés à elle de façon directe, tandis qu'un intermédiaire de confiance peut rehausser la fiabilité des échanges.

Ce mécanisme d'**influence à distance** se retrouve dans des domaines variés. En **robotique multi-agents**, un agent \mathcal{E}_i peut influencer un autre \mathcal{E}_j via un agent \mathcal{E}_k jouant le rôle de relai ; la structure d'échange se reconfigure selon la performance observée. En **neurosciences**, de nouvelles synapses directes se forment après une longue période où l'activité neuronale transite par plusieurs synapses intermédiaires.

Conclusion et ouverture

Les **interactions indirectes** forment un pivot crucial pour saisir l'**auto-organisation** d'un Synergistic Connection Network (SCN). Deux entités \mathcal{E}_i et \mathcal{E}_j initialement faiblement connectées peuvent se **rapprocher** grâce aux chemins multiples les reliant via des tiers, tout comme des boucles internes peuvent amplifier ou réguler la synergie. Cette dimension de **propagation** et de **contexte** est essentielle pour comprendre la **formation** de macro-structures, la dynamique des **cycles** et la **coévolution** des pondérations dans le DSL.

La section suivante (1.4.7) examinera de plus près la **synergie binaire** et surtout la **synergie n-aire**, c'est-à-dire l'impact de coopérations impliquant simultanément plus de deux entités, un aspect qui renforce encore la **richesse** des comportements émergents dans le Deep Synergy Learning.

1.4.7. Synergie binaire et n-aire : au-delà des relations deux à deux

La **synergie binaire** demeure la forme la plus couramment utilisée pour évaluer le degré de coopération entre deux entités \mathcal{E}_i et \mathcal{E}_j . On définit pour cela une fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ (pouvant être fondée sur une distance, une similarité, de la co-information, etc.) qui mesure dans quelle mesure

$$\mathbf{x}_i (+ \mathbf{s}_i, \theta_i) \Rightarrow \mathbf{x}_j (+ \mathbf{s}_j, \theta_j)$$

s'enrichissent mutuellement ou, au contraire, s'avèrent redondantes ou peu utiles l'une à l'autre.

- **Mise à jour.** À chaque itération, la pondération $\omega_{i,j}(t)$ se voit recalculée en fonction de la **valeur** $S(\mathcal{E}_i, \mathcal{E}_j)$.

- **Clusters.** Les clusters décrits en section 1.4.3 émergent généralement d'une **somme** ou d'une **agrégation** de ces liaisons binaires (souvent assortis d'un seuil, stabilisant la formation de groupes plus étendus).

Toutefois, la synergie binaire suppose que l'apport mutuel entre \mathcal{E}_i et \mathcal{E}_j ne dépende **pas** explicitement des autres entités. Or, la section 1.4.6 a montré que le **contexte** global (présence de \mathcal{E}_k, \dots) peut influer de manière significative : certaines lacunes peuvent être comblées, certaines informations complémentaires se mutualiser, ce qui peut révéler une **coopération** émergente entre plusieurs entités à la fois.

1. Synergie n-aire : idée d'un effet collectif

La **synergie n-aire** engage un **ensemble** $\{\mathcal{E}_{k_1}, \dots, \mathcal{E}_{k_m}\}$ (avec $m \geq 3$ dans la plupart des cas), dont la **combinaison** fournit une valeur ajoutée qu'on ne saurait réduire à la somme des synergies binaires. On écrit

$$S_n(\mathcal{E}_{k_1}, \dots, \mathcal{E}_{k_m})$$

pour désigner la synergie qui s'exprime **simultanément** entre ces m entités. On veut y saisir l'idée que "toutes ensemble, elles valent davantage que la somme de leurs collaborations deux à deux". Pour le cas de trois entités $\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c$, on parle parfois de **complémentarité stricte** :

$$S_n(\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c) > S(\mathcal{E}_a, \mathcal{E}_b) + S(\mathcal{E}_b, \mathcal{E}_c) + S(\mathcal{E}_a, \mathcal{E}_c),$$

signifiant que le **trio** dans son intégralité apporte plus que n'importe quelle addition de paires isolées. Pour $m > 3$, on généralise ce principe : c'est la **coopération collective** qui prime.

En **théorie de l'information**, on retrouve cette distinction dans :

60. Information mutuelle totale.

$$I(\mathbf{X}_1, \dots, \mathbf{X}_m) = \sum_{i=1}^m H(\mathbf{X}_i) - H(\mathbf{X}_1, \dots, \mathbf{X}_m),$$

laquelle ne fait pas le tri entre redondance et synergie réellement n-aire.

61. Partial Information Decomposition (PID).

Ce formalisme sépare la part d'information partagée par tous (redondance) de la part spécifiquement **synergique**, c'est-à-dire l'information qui n'apparaît qu'en combinant l'ensemble complet des variables $\mathbf{X}_1, \dots, \mathbf{X}_m$. S'il émerge une synergie n-aire positive, on découvre un **surcroît** d'information inexistant dans tout sous-groupe plus restreint.

Pour le **DSL**, ces notions indiquent comment un cluster de taille m peut se constituer lorsqu'on détecte une synergie n-aire **positive** (ou dépassant un certain seuil).

2. Intégration de la synergie n-aire dans un réseau

Si les pondérations $\omega_{i,j}$ portent sur des **liaisons binaires**, comment tirer parti d'une synergie n-aire ? Plusieurs pistes :

1. Approche factorisée.

Créer explicitement une pondération ω_{k_1, \dots, k_m} reliant simultanément les entités $\{\mathcal{E}_{k_1}, \dots, \mathcal{E}_{k_m}\}$. Il faut alors manipuler un **hyper-graph** (où une "arête" connecte plus de deux nœuds) ou stocker un **tenseur** de pondérations, ce qui devient coûteux si m croît.

2. Approche "bonus/malus".

Reprendre le cadre binaire $\omega_{i,j}$ en y ajoutant un **terme** lié à la synergie n-aire. Par exemple, pour trois entités $\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c$, on écrit :

$$\omega_{a,b}(t+1) = \omega_{a,b}(t) + \eta [S(\mathcal{E}_a, \mathcal{E}_b) + \gamma S_3(\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c) - \tau \omega_{a,b}(t)].$$

Ici, γ pèse la contribution de la synergie tripartite. Si $\{\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c\}$ forment un trio très coopératif, chaque lien binaire (a–b, b–c, a–c) reçoit un **surcroît** de renforcement, favorisant la cristallisation d'un micro-cluster à trois.

3. Émergence de micro- ou macro-clusters grâce à la synergie n-aire

Lorsqu'un groupe $\{\mathcal{E}_{k_1}, \dots, \mathcal{E}_{k_m}\}$ manifeste une synergie n-aire élevée, on s'attend à :

1. **Renforcer** les liens binaires internes,
2. **Stabiliser** un sous-graphe “complet” ou presque, traduisant la cohésion du groupe,
3. Obtenir un **gain** global supérieur à la somme des seules paires.

Si, plus tard, un autre sous-groupe $\{\mathcal{E}_{r_1}, \dots, \mathcal{E}_{r_q}\}$ se joint à ce collectif parce que la **synergie** entre ces deux ensembles élargis se révèle à son tour avantageuse, on peut assister à la **fusion** de clusters en un **macro-cluster** plus vaste. L'approche n-aire explique comment le DSL n'est pas limité au cas binaire : il peut trouver qu'un ensemble de 4, 5 ou 10 entités coopèrent de manière exceptionnelle, justifiant la formation d'un **module** émergent.

On peut y voir une **hiérarchie** (ou un treillis) :

- Les **feuilles** : synergies binaires $\omega_{i,j}$.
- Les **nœuds supérieurs** : synergies tri-partites, 4-partites, etc.
- Les **sommets** : macro-clusters englobant de grands ensembles.

L'**auto-organisation** d'un SCN peut alors être comprise comme un cheminement dans cet espace d'interactions multiples, valorisant les groupes où la **complémentarité** est nettement profitable.

4. Illustrations et implications

1. **Multimodalité.** Audio et visuel n'ont pas forcément une forte similarité, mais l'ajout d'une troisième modalité (ex. texte) peut révéler une synergie à trois entités. Ainsi, un macro-cluster se forme pour gérer ensemble la parole, l'image du locuteur et les informations projetées.
2. **Systèmes d'agents.** Trois robots (ou plus) peuvent constituer un **équipage** dont la coopération tri- ou multi-partite débloque une solution qu'aucune sous-combinaison plus simple ne parvenait à réaliser. Le DSL rend compte de cette nécessité de formation de “super-équipes”.
3. **Chimie ou biologie.** Deux composés chimiques A et B ne réagissent qu'imparfaitement, jusqu'à ce qu'on introduise un troisième C qui agit en **catalyseur**. On met alors en évidence une **synergie triple** (A–B–C) supérieure à toute combinaison restreinte.

Conclusion

Si la **synergie binaire** constitue le socle même du SCN, la prise en compte de la **synergie n-aire** enrichit de manière considérable la capacité du DSL à **déceler et exploiter** des coopérations plus étendues. Les interactions à trois entités ou davantage peuvent révéler des **complémentarités** ou des **effets émergents** impossibles à discerner en se limitant aux paires. D'un point de vue pratique, le coût combinatoire de la gestion explicite de synergies n-aires peut se révéler élevé, mais des stratégies de type "bonus/malus" sur les liens binaires offrent des voies **approximatives** pour modéliser ces phénomènes.

Ainsi, la **synergie n-aire** confère au DSL un cadre globalement **plus puissant** et **expressif**, permettant de voir s'**auto-organiser** des ensembles coopératifs complexes et de mieux expliquer la naissance de **macro-clusters** fortement intégrés. Cette dimension s'ajoute à la dynamique temporelle (1.4.5) et aux interactions indirectes (1.4.6) pour compléter la **vision** d'un **réseau** éminemment **adaptatif**, ajustant ses liaisons au fil des découvertes sur la synergie de ses entités.

1.5. Pourquoi une Approche Synergique ?

Les sections précédentes ont mis en évidence la manière dont le **Deep Synergy Learning (DSL)**, à travers son **architecture générale** et son **auto-organisation** (voir sections 1.3 et 1.4), se distingue d'une approche classique. Pour mieux comprendre l'intérêt d'une telle démarche, il importe de répondre à une question cruciale : **pourquoi** choisir un paradigme "synergique" plutôt que de se contenter des réseaux de neurones profonds ou d'autres méthodes d'apprentissage ? Cette section 1.5 propose plusieurs raisons majeures :

4. **Avantages par Rapport aux Réseaux Neuronaux Profonds** (1.5.1)
5. **Gestion Naturelle de la Multi-modalité** (1.5.2)
6. **Flexibilité vis-à-vis des Données Incomplètes ou Bruitées** (1.5.3)
7. **Potentiel d'Auto-Évolution et d'Adaptation Continue** (1.5.4)
8. **Réduction de la Dépendance à la Supervision Humaine** (1.5.5)
9. **Création de Représentations Riches et plus Interprétables** (1.5.6)
10. **Intégration de Dimensions Symboliques ou Cognitives** (1.5.7)

Chacun de ces points met en lumière les **bénéfices** d'un modèle qui valorise la **co-opération** dynamique entre entités, plutôt qu'une hiérarchie de couches figées. Nous allons tout d'abord (1.5.1) confronter la logique **synergique** du DSL à la structure rigide des **réseaux neuronaux profonds (Deep Learning)**, pour en dégager les avantages potentiels.

1.5.1. Avantages par Rapport aux Réseaux Neuronaux Profonds

Les **réseaux neuronaux profonds** (CNN, RNN, Transformers, etc.) ont assurément démontré leur **efficacité** dans de nombreux domaines. Cependant, le **DSL** propose une **philosophie** radicalement différente, susceptible d'apporter :

11. **Flexibilité Structurelle**
12. **Plastique et Auto-Organisé**
13. **Réduction de la Spécialisation Rigueure**
14. **Facilité d'Intégration Multi-entités**
15. **Apprentissage Continu**

Chacun de ces volets constitue un avantage important, que nous détaillons ci-après.

1.5.1.1. Flexibilité Structurelle

Dans un **réseau neuronal profond** classique, la **topologie** (nombre et type de couches, neurones, schémas de fusion de données) est entièrement **préétablie** avant l'apprentissage. Par exemple, un **CNN** (Convolutional Neural Network) organise les données selon des couches **convolutionnelles**, suivies éventuellement de **pooling** et de **couches fully-connected**, sans changement d'architecture au cours de l'entraînement. De même, les **Transformers** enchaînent des **blocs self-attention** répétés, toujours selon un canevas décidé en amont. Ces modèles **ne modifient** pas leur structure interne en cours de fonctionnement :

Nombre de couches, arrangement des blocs et synapses demeurent fixes pendant tout l'apprentissage.

Ils **n'autorisent** pas la création ou la **suppression** de neurones ou de liaisons et conservent un agencement identique, même si la **distribution** des données ou le **contexte** subit des modifications significatives.

À l'inverse, dans le **Deep Synergy Learning (DSL)**, on insiste sur la dynamique de ses **entités** et de leurs **liaisons**. Les pondérations $\omega_{i,j}(t)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j évoluent selon la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Il est possible qu'un **lien** émerge (si la synergie grandit et dépasse un certain **seuil**), qu'il **disparaisse** (si l'intérêt coopératif s'avère trop faible), ou qu'il se **recrée** au cours du temps. La **flexibilité** topologique qui en résulte se formalise par des règles de mise à jour de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η désigne le **taux d'apprentissage** et τ un **terme de régulation**. Quand la synergie S s'avère positive et assez élevée, la liaison se **renforce** ; dans le cas contraire, elle **s'affaiblit** et peut même s'annuler. Cette dynamique se révèle d'autant plus souple qu'on peut imposer une **parsimonie** via un seuil ω_{\min} ou un **cap** ω_{\max} , comme le rappelle la section 1.4.5.

La conséquence directe de cette **reconfiguration dynamique** tient dans la capacité du réseau à **s'adapter** en continu à l'arrivée de nouvelles données ou à un **changement** de distribution. Au lieu de maintenir invariablement le même chemin de traitement (comme dans un **CNN** figé ou un **Transformers** aux blocs intangibles), le **DSL** autorise une **reliance** directe entre certaines entités apparues pertinentes, tout en relâchant ou en supprimant d'autres connexions devenues inutiles. On peut ainsi écrire, de façon synthétique :

$$\text{Flexibilité topologique} \Rightarrow \text{Capacité d'adaptation et résilience face à l'évolution du contexte.}$$

Dès lors, si la distribution des données subit un changement progressif (par exemple, l'apparition d'une nouvelle modalité sensorielle), le **DSL** peut incorporer de nouvelles **entités** \mathcal{E}_{nov} et renforcer les liaisons $\omega_{\text{nov},j}$ jugées porteuses d'une forte **synergie**. Simultanément, il peut affaiblir ou rompre des liens moins pertinents. C'est précisément cette **malléabilité** qui caractérise le **SCN** et le différencie de la rigidité structurelle des architectures profondes traditionnelles.

Ce **principe** de flexibilité structurelle jette les bases d'une **évolution** continue de la topologie, conforme à l'esprit d'**auto-organisation** présenté dans les sections 1.4.3 et 1.4.5. En ce sens, le **DSL** épouse davantage la **complexité** et la **variabilité** des environnements réels, tout en garantissant que seules les **connexions** profitables (au sens de la synergie) se **consolident** dans le temps.

1.5.1.2. Plasticité et Auto-Organisation

Dans les **réseaux de neurones** usuels, la phase d'apprentissage consiste principalement à **ajuster** les **poids** dans une **architecture** figée à l'avance. En revanche, le **Deep Synergy Learning (DSL)** adopte un parti pris plus **plastic** : il autorise la **topologie du Synergistic Connection Network (SCN)** à **évoluer** au cours du temps. Les liaisons $\omega_{i,j}(t)$ qui peinent à démontrer une **synergie** positive ou dont la valeur reste trop faible peuvent **disparaître**, tandis que d'autres connexions, révélant un intérêt mutuel marqué, se **renforcent** jusqu'à forger de nouveaux chemins ou **clusters**.

Pour exprimer cette mise à jour, on recourt le plus souvent à une règle sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η représente le **taux d'apprentissage**, τ un terme de **régulation**, et $S(\mathcal{E}_i, \mathcal{E}_j)$ la **synergie** (distance, similarité, information mutuelle...). Ainsi, chaque itération propose une **réévaluation** de la pertinence du lien $\omega_{i,j}$. Si la synergie demeure assez élevée, la liaison persiste ou se consolide ; sinon, elle tend vers zéro et peut être considérée comme **rompue** selon un seuil ω_{\min} (cf. section 1.4.5 pour la gestion des seuils).

Cette **démarche** de régulation évoque la **plasticité synaptique** que l'on observe dans les **systèmes biologiques**. À l'image des neurones qui renforcent leurs connexions en cas de co-activation et en abandonnent d'autres, les entités du DSL remanient leur **graphe** de manière à privilégier les **coopérations** les plus fructueuses. Les **clusters** (sections 1.4.3 et 1.4.4) apparaissent alors **spontanément**, sans exiger de paramétrage explicite du nombre de groupes : ils émergent lorsque les pondérations internes d'un sous-ensemble d'entités dépassent certains niveaux de synergie, consolidant un **sous-réseau** fortement lié.

Cette aptitude à **auto-organiser** la structure s'avère avantageuse pour l'**adaptation** en continu. On peut en effet passer d'une tâche à l'autre ou intégrer une **nouvelle modalité** de données (image, audio, signal textuel) sans reconstruire l'architecture dans son intégralité, puisqu'il suffit d'ajouter des entités et de laisser les liaisons se former ou s'éteindre de façon autonome selon la **synergie** détectée. Les **clusters** demeurant utiles persistent, tandis que de **nouveaux** se créent si des **interactions** inédites se révèlent rentables. Le **réseau** conserve ainsi un **degré de flexibilité** qui lui permet d'**ajuster** sa **complexité** selon le **contexte**, proposant un fonctionnement plus souple qu'un modèle entièrement figé.

1.5.1.3. Réduction de la Spécialisation Rigide

Dans un **réseau profond** standard comme un **CNN** dédié à la reconnaissance visuelle, la structure se présente souvent de manière **verticalisée**, où les premières couches extraient des **bords**, puis des formes plus élaborées, jusqu'à la classification finale. Cette disposition rend la **réutilisation** des *features* complexes si l'on souhaite aborder d'autres tâches ou modalités, à moins d'effectuer un **fine-tuning** parfois lourd. Les couches initiales sont fixées dans une fonction très spécifique, et la **coopération** entre différents blocs (par exemple la fusion de vision et d'audio) se voit généralement cantonnée à un **niveau** imposé dans l'architecture ou à un **module** explicitement conçu pour la multimodalité.

Dans le cadre du **Deep Synergy Learning (DSL)**, la vision se veut plus **flexible**. Les **entités** composant le **Synergistic Connection Network (SCN)** incarnent chacune une **source d'information** ou un **bloc fonctionnel** susceptible d'être réutilisé dans d'autres contextes. La **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ détectée entre deux **entités** \mathcal{E}_i et \mathcal{E}_j ne se limite pas à un étage "plus haut" ou "plus bas" : toute entité peut, au fil des itérations, créer un **lien** avec toute autre, dès lors que la coopération leur procure un **gain** mesuré par la fonction S .

Un **exemple** concret peut être imaginé lorsqu'un **bloc** $\mathcal{E}_{\text{bords}}$, spécialisé dans la détection de **bords** visuels, observe soudain qu'il existe un **gain** d'associer ces **features** à un **bloc** $\mathcal{E}_{\text{audio}}$ focalisé sur les **fréquences** sonores, notamment quand des corrélations entre mouvements de lèvres et signaux vocaux se manifestent. Au sein d'un **CNN** traditionnel, l'**intégration** de telles informations se produirait typiquement à un **niveau tardif** ou via un **module spécialisé**, ce qui limite la portée de la coopération. Dans le **DSL**, au contraire, la liaison $\omega_{\text{bords}, \text{audio}}$ peut croître de manière **spontanée** si la **synergie** calculée (par exemple $\exp(-\|\mathbf{x}_{\text{bords}} - \mathbf{x}_{\text{audio}}\|^2/\sigma^2)$) s'avère **élevée** et dépasse un certain **seuil** ω_{\min} (section 1.4.5). Les deux blocs initient alors un **micro-cluster**, échangent leurs informations, et ainsi la **fusion** s'opère de manière autonome. Cette capacité à **tisser** des liens coopératifs entre entités diverge de la spécialisation rigide instaurée par des **pipelines** de couches fixes, et elle illustre la **souplesse** du **DSL** pour exploiter des **synergies** inattendues.

1.5.1.4. Facilité d'Intégration Multi-Entités

Dans la plupart des **réseaux neuronaux profonds** classiques, il est usuel de concevoir un **design** architectural bien particulier pour **fusionner** plusieurs entrées issues de différentes modalités, comme une **image**, un **texte** et un **signal audio**. On trouve ainsi des approches où un **CNN** traite l'image, un **RNN** (ou un **Transformers**) interprète le texte, puis une **couche de fusion** se situe au sommet pour combiner les vecteurs latents. Cette disposition requiert, dans son principe, une intervention humaine afin de spécifier à quel niveau et selon quelles règles la **fusion** se produit.

Dans le **Deep Synergy Learning (DSL)**, la perspective est nettement plus **flexible**. Chaque **modalité** s'incarne dans une ou plusieurs **entités** \mathcal{E}_{mod} dédiées, que l'on peut définir comme un bloc perceptif ou un ensemble de **features** cohérent. Les **liens** $\omega_{i,j}$ se **construisent** de manière autonome, selon la **synergie** mesurée entre les entités \mathcal{E}_i et \mathcal{E}_j . On peut exprimer cette dynamique à l'aide de la règle d'adaptation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où η désigne le **taux d'apprentissage**, τ un **terme de régulation**, et $S(\mathcal{E}_i, \mathcal{E}_j)$ la **synergie** (section 1.4.4). Cette règle fait en sorte que toute **coopération** mutuellement avantageuse s'**intensifie** si elle contribue à un gain, alors que les associations moins pertinentes se **dissipent**. Dans un **contexte** multimodal, l'image, le texte et l'audio sont autant

d'entités $\{\mathcal{E}_{\text{image}}, \mathcal{E}_{\text{texte}}, \mathcal{E}_{\text{audio}}\}$ susceptibles de découvrir des **corrélations** bilatérales ou trilatérales, sans qu'une **couche de fusion** imposée par un concepteur soit nécessaire.

Lorsque la **synergie** entre ces modalités se révèle conséquente, les pondérations $\omega_{\text{image}, \text{texte}}$, $\omega_{\text{image}, \text{audio}}$ ou $\omega_{\text{texte}, \text{audio}}$ croissent, formant de manière **spontanée** un **cluster** multimodal où les entités coopèrent directement. Cette **auto-organisation** se renforce si l'association apporte un bénéfice mesurable (par exemple, une performance améliorée sur une tâche de classification ou une plus grande robustesse). Inversement, si une modalité \mathcal{E}_{mod} n'apporte pas d'information utile ou se révèle trop bruitée, la **synergie** reste faible et la liaison ne s'établit pas, ou se rompt rapidement.

Une autre conséquence réside dans la **capacité** du réseau à **adapter** en continu les entités multimodales, autorisant la **formation** et la **dissolution** d'un **cluster** au gré du contexte. Cette propriété est en net contraste avec l'idée traditionnelle d'une couche de fusion figée où toutes les modalités se rejoignent selon un schéma unique. Le **DSL** propose donc une **répartition** dynamique de l'intégration multimodale, dans laquelle les **entités** se reconfigurent en fonction de l'**opportunité** d'apprentissage.

Au final, la **facilité d'intégration multi-entités** découle directement de la manière dont les **liaisons synergiques** sont définies et mises à jour. Chaque modalité, qu'elle concerne le **visuel**, le **textuel**, le **sonore** ou un **autre flux**, est **libre** de **découvrir** et de **former** des **connexions** directes avec n'importe quel **bloc** complémentaire, dès lors qu'une **valeur ajoutée** en résulte. Le **réseau** opère ainsi sous une forme d'**auto-organisation** (sections 1.4.3 et 1.4.5), permettant d'exploiter la diversité des **modalités** sans nécessiter le design d'un unique **module** de fusion, et garantissant une **adaptation** continue à la **variabilité** de l'environnement.

1.5.1.5. Apprentissage Continu

Dans de nombreux **réseaux neuronaux profonds**, l'entraînement se déroule dans un cadre essentiellement statique. On se contente généralement d'effectuer une rétropropagation sur un **ensemble de données** figé, et il s'avère difficile de réaliser un **apprentissage continu** sans retomber dans le phénomène de **catastrophic forgetting** au moment d'acquérir de nouvelles tâches ou de faire face à un changement de distribution. De plus, la structure même du réseau (couche, neurones) demeure immuable pendant tout l'apprentissage, ce qui limite considérablement la possibilité d'**évoluer** au fil du temps.

À l'inverse, le **Deep Synergy Learning (DSL)** s'appuie sur un **mécanisme de pondérations adaptatives** (discuté en section 1.4.5) dans lequel chaque liaison $\omega_{i,j}(t)$ peut se créer, se renforcer ou disparaître selon la **synergie** qu'elle procure. Cette propriété ouvre la voie à un **apprentissage continu** plus fluide, car le **réseau** peut accueillir de nouvelles **entités** $\mathcal{E}_{\text{nouv}}$ chaque fois qu'un nouveau flux de données ou une nouvelle modalité fait son apparition. Les **liaisons** $\omega_{\text{nouv}, j}$ se forment si elles sont jugées bénéfiques, tandis que les anciens **clusters** jugés toujours utiles demeurent stables. Ainsi, l'**oubli** des acquis se trouve **atténué**, puisque les **connections** consolidées par le passé ne sont pas supprimées tant qu'elles conservent une synergie positive. Les entités correspondant aux tâches ou domaines précédents (sections 1.4.3 et 1.4.4) ne disparaissent pas, et les sous-groupes de coopération qui s'étaient formés peuvent persister ou se réactiver ultérieurement.

Une telle **flexibilité** convient particulièrement aux **environnements dynamiques**, où le **réseau** doit sans cesse **apprendre** de nouvelles classes d'objets, de nouvelles langues, ou s'adapter à des évolutions de contexte sans recommencer l'**entraînement** depuis zéro. Le **DSL** se distingue alors de la rigidité caractéristique d'un **réseau profond** standard, qui impose généralement de réinitialiser ou de "fine-tuner" un ensemble de couches fixes lors de l'introduction d'un nouveau domaine. La possibilité de **reconfigurer** la topologie et d'**absorber** ou de **cesser** d'utiliser certaines sources de données fait du **SCN** un cadre mieux adapté à un **apprentissage Lifelong** ou **continu**, tout en préservant la robustesse et la puissance d'un paradigme sub-symbolique.

Conclusion

En comparaison directe avec les **réseaux neuronaux profonds**, l'**approche synergique** du **DSL** propose :

- **Une topologie évolutive**, au lieu d'une hiérarchie figée,
- **Une plasticité** inspirée (partiellement) de la biologie, plutôt qu'un pipeline statique,
- **Une modularité** et **une fusion** multimodale plus naturelles, sans imposer de couches de fusion,
- **Un apprentissage continu** mieux géré, permettant l'ajout ou la suppression d'entités sans tout réapprendre.

Il ne s'agit pas de nier la **puissance** des réseaux profonds : ils restent extrêmement performants lorsqu'on dispose de gros volumes de données annotées et qu'on accepte une architecture prédéfinie. Mais le DSL entend dépasser les limites liées à la **spécialisation**, au **statisme** et à la **dépendance** de l'apprentissage neuronal classique, ouvrant la voie à une **IA adaptative** et plus **ouverte** à la co-opération des informations.

Dans les sous-sections suivantes (1.5.2 à 1.5.7), nous approfondirons d'autres avantages majeurs, comme la **gestion naturelle de la multi-modalité** (1.5.2), la **flexibilité** face aux données partielles (1.5.3), l'**auto-évolution** (1.5.4), la **réduction** de la supervision (1.5.5), la **création** de représentations plus **riches** (1.5.6) et l'**intégration** de dimensions **symboliques** (1.5.7). L'ensemble consolidera l'idée qu'une **approche synergique** offre un panel d'avantages pour concevoir des systèmes apprenants plus **évolutifs**, **modulaires** et **généraux** que ne le permettent les architectures entièrement figées du deep learning traditionnel.

1.5.2. Gestion Naturelle de la Multi-modalité

Une des motivations fortes qui a conduit à l'élaboration du **Deep Synergy Learning (DSL)** est la possibilité d'intégrer plusieurs types de données (*images, sons, textes, capteurs variés, etc.*) de manière à ce qu'elles **coopèrent** au lieu de simplement être fusionnées dans une couche dédiée. Dans les réseaux neuronaux profonds classiques, la multi-modalité impose souvent de **concevoir** des modules spécifiques (p. ex. un CNN pour l'image, un RNN ou un Transformer pour le texte, un autre réseau pour l'audio), puis de **fusionner** ces modules dans une partie supérieure de l'architecture. Cette approche, quoique efficace dans de nombreux cas, reste relativement **rigide** : la fusion se fait à un niveau imposé et ne varie guère en fonction du contexte ou de la dynamique interne.

Le **DSL**, au contraire, encourage une **intégration plus organique** : chaque modalité est représentée par une (ou plusieurs) **entité(s)** \mathcal{E}_{mod} , qui cherchent spontanément à établir ou à rompre des liens avec des entités d'autres modalités, suivant la **synergie** constatée. Cette sous-section (1.5.2) détaille :

16. **Pourquoi** la multi-modalité est “naturellement” absorbée par le DSL,
17. Comment les entités **visuelles, auditives, textuelles** (par ex.) peuvent **former** des clusters multimodaux,
18. Quels **avantages** cette gestion apporte (robustesse, découvertes de liens inattendus, etc.).

1.5.2.1. Les Entités comme Vecteurs ou Blocs Multimodaux

Dans le **Deep Synergy Learning (DSL)**, chaque **entité** \mathcal{E}_i peut se spécialiser dans un type de données particulier ou combiner déjà plusieurs **descripteurs**. Il n'y a pas l'obligation, comme dans un réseau multimodal classique, d'assembler deux flux (image, audio, etc.) au niveau d'une couche imposée ou dans des branches séparées. Au lieu de cela, toutes les entités coexistent dans un **même “espace”** ou **graphe** $\{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n\}$. On y retrouve des entités $\mathcal{E}_{\text{image}}$, $\mathcal{E}_{\text{audio}}$, $\mathcal{E}_{\text{texte}}$, ou des blocs déjà **multimodaux** combinant différents **features**. Il n'existe pas de **pyramide** de couches fixes, comme un CNN qui détecterait d'abord les bords, puis les textures, puis les objets. Au contraire, le **DSL** insiste sur le principe d'**auto-organisation**, faisant que toute entité \mathcal{E}_i peut se **lier** à toute autre \mathcal{E}_j dès lors qu'une **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ significative est détectée. La mise à jour de la pondération $\omega_{i,j}(t)$ suit la règle discutée en section 1.4.5, et si la coopération apporte un gain, on obtient un renforcement de la liaison $\omega_{i,j}$. Ainsi, une entité de type *texte* \mathcal{E}_{txt} n'est pas cantonnée à se fusionner uniquement avec une entité *image* \mathcal{E}_{img} dans un étage supérieur prédéfini : la **fusion** peut advenir à **n'importe quel moment**, entre n'importe quelles entités, pourvu qu'une **valeur ajoutée** soit constatée. Cette démarche favorise la **formation** naturelle de **clusters** multimodaux là où c'est nécessaire, et permet aux entités déjà

mixtes (par exemple combinant audio et vision) de se relier également à d'autres (texte, capteurs...), rendant le réseau plus **souple** et mieux à même de gérer la **pluralité** des sources de données.

1.5.2.2. Formation de Clusters Multimodaux

Le **Deep Synergy Learning (DSL)** emploie une règle de mise à jour décrite dans la section 1.4.5, qui se formalise ainsi :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Lorsque, par exemple, une **entité visuelle** $\mathcal{E}_{\text{visuelle}}^a$ (extraction d'images) et une **entité auditive** $\mathcal{E}_{\text{auditive}}^b$ (analyse du spectre sonore) constatent une **synergie** satisfaisante, la liaison $\omega_{a,b}$ se **renforce** progressivement. Les sections 1.4.3 et 1.4.4 évoquent déjà ce principe de consolidation, où toute **coopération** profitable se traduit par une augmentation de la pondération. Au fil des itérations, d'autres entités, qu'elles soient textuelles ou associées à une autre modalité, peuvent se **greffer** à ce duo si elles y perçoivent elles aussi un **gain**. S'organise alors un **cluster** multimodal $\{\mathcal{E}_{\text{vis}}, \mathcal{E}_{\text{aud}}, \mathcal{E}_{\text{txt}}\}$, destiné, par exemple, à la reconnaissance d'événements conjoints dans une vidéo, en y associant le son et des sous-titres.

Un exemple important survient lors de l'analyse synchronisée **vidéo + audio**. Une entité \mathcal{E}_{CNN} , spécialisée dans la détection d'objets ou de mouvements visuels, et une entité $\mathcal{E}_{\text{Audio}}$, orientée vers les fréquences sonores, découvrent une **corrélation** entre le contenu visuel (mouvements de l'orateur) et des sons spécifiques. Si la **fonction de synergie** $S(\mathcal{E}_{\text{CNN}}, \mathcal{E}_{\text{Audio}})$ affiche une valeur élevée, la pondération $\omega_{\text{CNN}, \text{Audio}}$ croît. Il en résulte la formation d'un **sous-réseau** constitué de ces entités, apte à identifier les vidéos où un son précis se déclenche simultanément avec un geste ou un mouvement labial.

Un autre scénario se rencontre dans la **fusion texte + image**. Si une entité $\mathcal{E}_{\text{visuel}}$ extrait les objets visibles dans une scène et qu'une entité $\mathcal{E}_{\text{langage}}$ génère des phrases descriptives (via un modèle de type RNN ou Transformers), la **co-information** qu'elles partagent peut montrer qu'en les associant, on obtient une légende automatique bien plus fidèle. Cette plus-value se reflète dans la liaison $\omega_{\text{visuel}, \text{langage}}$, qui se **solidifie** de manière autonome. Le **cluster** $\{\mathcal{E}_{\text{visuel}}, \mathcal{E}_{\text{langage}}\}$ se spécialise alors dans la génération de légendes, sans imposer de couche de fusion pré-déterminée.

On peut enfin imaginer un **cluster hybride** regroupant à la fois la "capture de mouvement" via des capteurs inertIELS, la "reconnaissance d'images" au moyen d'une entité visuelle, et l'"analyse du spectre sonore". Ce **triplet** se cristallise dès lors qu'il existe une forte complémentarité pour identifier des événements qui combinent aspects visuels, audio, et déplacements corporels. Le **SCN** encourage ainsi la création de tels **clusters** dès que les synergies apparaissent bénéfiques, ce qui permet au réseau de s'adapter aux environnements multimodaux et de tirer le meilleur parti de chacune des sources disponibles.

1.5.2.3. Avantages de la Co-Organisation Multimodale

Dans un **réseau neuronal** habituel, la fusion des différents **flux** (par exemple, l'image et le son) est généralement décidée à l'avance, souvent dans les étages supérieurs ou dans une **couche intermédiaire** prédéfinie. Le **Deep Synergy Learning (DSL)** introduit une perspective différente : chaque **entité** \mathcal{E}_i peut spontanément **rechercher** des partenaires synergiques, **évaluer** si la coopération améliore les performances ou apporte un gain en information mutuelle, et **créer** des liaisons directes $\omega_{i,j}$ sans qu'une couche de fusion imposée soit nécessaire. Cette démarche favorise l'apparition de **combinaisons** inhabituelles mais profitables, comme l'association de l'analyse audio, de la mesure de température et de l'intensité lumineuse, si un cluster pertinent émerge.

Concrètement, la **règle** de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

permet à chaque entité de constater, de façon autonome, si sa **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ s'avère suffisamment élevée pour justifier une **collaboration**. Ainsi, il n'existe pas d'étage unique où l'on agrégerait tous les flux, mais plutôt un **réseau** dont la topologie évolue pour intégrer toute coopération bénéfique.

Lorsqu'une modalité est **perturbée** ou moins fiable (par exemple l'audio, soumis à un **bruit** important ou une panne de capteur), la synergie associée $\omega_{\text{aud}, i}$ s'affaiblit, car les entités n'y trouvent plus de **valeur ajoutée**. Le **cluster** multimodal se réorganise donc en renforçant d'autres connexions, comme celles reliant la vision, le texte ou des capteurs alternatifs. Le système bénéficie ainsi d'une **résilience** accrue : il ne dépend pas d'un pipeline figé ni d'un module unique pour chaque modalité, mais s'autorégule en continu pour préserver une **robustesse** globale face aux changements du contexte ou aux défaillances de certaines sources.

1.5.2.4. Éléments Mathématiques : Synergie “Multi-Modal”

Dans le **Deep Synergy Learning (DSL)**, la mesure de **synergie** entre différentes **modalités** (comme l'audio, la vision ou le texte) peut prendre plusieurs formes selon le type de données ou la définition souhaitée de la **co-opération**. On considère deux grandes approches : l'usage d'une **co-information** (ou multi-information) tirée de la théorie de l'information, et le recours à une **distance** ou **similarité** inter-modale dans un espace latent commun.

A. Mesure de co-information

Lorsque l'on manipule un ensemble de modalités $\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m$, il est possible d'**estimer** une **co-information** ou **mutual information** globale :

$$S(\mathbf{X}_1, \dots, \mathbf{X}_m) = \text{MI}(\mathbf{X}_1, \dots, \mathbf{X}_m).$$

Dans certains cas, la **Partial Information Decomposition (PID)** permet de séparer la part strictement *synergique* de la part *redondante* ou *unique*. Dans le **DSL**, on s'intéresse à l'**analyse n-aire** au sens où si la **conjonction** de deux (ou plus) modalités, par exemple $\mathbf{X}_{\text{visuelle}}$ et $\mathbf{X}_{\text{auditive}}$, produit une **information** qu'aucune des modalités ne détenait isolément, alors on en conclut un **gain** pour la **synergie**. Ce **gain** s'exprime par un **renforcement** des pondérations : si l'usage conjoint d'un flux visuel et d'un flux audio s'avère utile, la liaison $\omega_{\text{vis}, \text{aud}}$ augmente selon la règle de mise à jour (section 1.4.5). L'**auto-organisation** du réseau valorise ainsi la coopération entre modalités lorsqu'elle apporte un surcroît d'information ou de performance.

B. Distance ou Similarité inter-modale

En alternative à la co-information, on peut quantifier la **synergie** via une **distance** (ou **similarité**) entre les modalités, à condition de se situer dans un **espace latent** commun. Dans la pratique, cela implique généralement les étapes suivantes :

(1) Embeddings partagés.

On forme une représentation vectorielle \mathbf{z}_{vis} pour la partie **visuelle** et \mathbf{z}_{aud} pour la partie **audio**, toutes deux dans \mathbb{R}^d . Ces embeddings peuvent provenir d'un réseau d'apprentissage dédié ou être appris conjointement.

(2) Distance ou similarité.

On calcule une **norme** $\|\mathbf{z}_{\text{vis}} - \mathbf{z}_{\text{aud}}\|$ ou une **similarité cosinus** $\langle \mathbf{z}_{\text{vis}}, \mathbf{z}_{\text{aud}} \rangle$. Cette quantité rend compte de la **proximité** ou de la **complémentarité** entre les deux modalités une fois projetées dans l'espace latent.

(3) Définition de la synergie.

En inversant ou en normalisant la distance, on obtient un **score de coopération** :

$$S(\mathbf{z}_{\text{vis}}, \mathbf{z}_{\text{aud}}) = \exp(-\|\mathbf{z}_{\text{vis}} - \mathbf{z}_{\text{aud}}\|^2 / 2\sigma^2) \quad \text{ou} \quad 1/(1 + \|\mathbf{z}_{\text{vis}} - \mathbf{z}_{\text{aud}}\|^2),$$

ou encore en recourant à une **similarité** normalisée dans [0,1]. Cette mesure, insérée dans la formule de mise à jour des pondérations $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$, renseigne sur la **pertinence** qu'il y a à **connecter** et **coordonner** ces deux modalités.

Synthèse et implication dans le DSL

Le **DSL** n'impose pas de définition unique de la synergie ; toute **fonction** mesurant la **coopération** peut s'appliquer. La **co-information** met en évidence des **complémentarités** complexes (non linéaires), en évaluant l'information que seules deux (ou plusieurs) modalités, **combinées**, parviennent à extraire. De son côté, la **distance** (ou la **similarité**) dans un **espace latent** offre un calcul plus direct et souvent moins coûteux, permettant d'établir rapidement une **coopération** si les embeddings s'avèrent proches ou orientés de façon compatible.

Dans tous les cas, on reste dans l'esprit du **DSL** : la synergie inter-modale agit comme un **signal** indiquant que les entités multimodales gagneraient à se **lier** et à former un cluster dédié. Cette logique s'applique à deux entités (binaire), mais peut s'étendre à des groupes plus vastes (section 1.4.7) lorsque plusieurs **modalités** s'enrichissent mutuellement. Dans une perspective **multimodale**, on voit ainsi se constituer des **clusters** audio-vidéo-texte, ou audio-vision-capteurs, sans qu'un concepteur doive imposer explicitement à quel niveau se fait la **fusion**. Le **SCN** se reconfigure et s'**auto-organise**, renforçant les **liens** rentables au sens de la synergie globale.

1.5.2.5. Perspectives : Plus de Fluidité, Plus de Découvertes

Dans un **Deep Synergy Learning (DSL)**, la fusion multimodale n'est pas confinée à une couche ou un module imposé à l'avance. Elle repose sur l'aptitude de chaque **entité** à forger des liens **synergiques** dès lors que la coopération procure un **gain**. Cette configuration autorise plusieurs évolutions particulièrement souples.

Un premier aspect concerne la **création spontanée** de **clusters multimodaux**. Lorsqu'un événement sollicite simultanément divers flux, par exemple un son particulier, un motif visuel distinctif et un mot-clé textuel, le **DSL** favorise la formation d'un **cluster** auto-organisé combinant toutes ces sources. Il n'existe pas l'exigence de décider, dans la conception du réseau, à quel étage ou sous-ensemble la fusion doit survenir : le **Synergistic Connection Network** se charge de **déetecter** les entités pertinentes, puis de renforcer leurs liaisons $\omega_{i,j}$ si la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ s'avère élevée. Cette auto-organisation, évoquée en section 1.4.3, rend la fusion bien plus fluide et adaptative.

Un deuxième enjeu apparaît lorsque des **modalités** ou sources de données nouvelles se présentent. Si un capteur inédit vient produire un nouveau flux, il suffit d'introduire une entité $\mathcal{E}_{capteur}$ correspondante. Les pondérations $\omega_{capteur,x}$ se mettent alors à jour selon la même loi que précédemment (section 1.4.5). Si les interactions avec des entités déjà existantes procurent un réel **bénéfice** à l'apprentissage ou à la représentation, la **synergie** grimpe ; sinon, le lien demeure faible et finit par s'éteindre. L'essentiel tient dans le fait qu'aucune **reconstruction** de l'architecture globale n'est nécessaire : le réseau s'**auto-adapte**, créant et dissolvant les connexions au gré des "opportunités" que confère le nouveau flux.

Un troisième point souligne la **réutilisation** des entités spécialisées. Dans un paradigme classique, un module dédié, par exemple à l'analyse de fréquences sonores, risque de ne pouvoir servir qu'à une tâche unique (identification de parole ou détection de musique). Au contraire, dans le **DSL**, si une entité \mathcal{E}_{audio} sait détecter des éléments discriminants dans le spectre sonore, elle peut être **sollicitée** par des **clusters** différents : on peut ainsi la retrouver dans un sous-réseau chargé de la reconnaissance de mots clés, un autre consacré à la détection de bruits inhabituels, ou encore un qui s'occupe d'associer des fragments de parole aux mouvements de lèvres. Chaque fois que la **synergie** le justifie, la liaison $\omega_{audio,\dots}$ se renforce dans le cluster concerné, sans imposer de duplication de la fonctionnalité ni de modification d'une couche fixe.

L'ensemble de ces mécanismes assure une **fluidité** et un **potentiel d'exploration** supérieurs par rapport aux modèles multimodaux figés. Le **DSL** favorise l'**émergence** et la **dissolution** dynamiques de clusters multimodaux, l'**intégration** graduelle de modalités nouvelles, et la **mutualisation** de blocs spécialisés au service de plusieurs tâches. Cette architecture, loin d'être imposée, se déploie au fil des itérations par simple adaptation des pondérations $\{\omega_{i,j}(t)\}$, conduisant à une plus grande **richesse** de découvertes et de combinaisons entre entités.

Conclusion

La **multi-modalité** constitue un **terrain privilégié** pour apprécier la **valeur ajoutée** du Deep Synergy Learning. Là où un **réseau neuronal** classique cloisonne les modalités dans des branches séparées, puis les fusionne **sur** commande, le **DSL** propose :

- Une **co-intégration libre** des entités,
- Une **dynamique** où le réseau détecte **de lui-même** quelles modalités se complètent,
- Une **robustesse** accrue lorsque certaines modalités sont indisponibles ou bruitées,
- Une **exploration plus approfondie** des interactions possibles (même celles qu'on n'avait pas anticipées).

C'est pourquoi, dans l'optique d'une IA **multi-sensorielle** ou traitant des **sources hétérogènes**, le DSL offre un **cadre naturel et organique** pour favoriser les **synergies**. On verra dans les points suivants (1.5.3 à 1.5.7) d'autres aspects tout aussi clés (comme la flexibilité face aux données incomplètes, la réduction du besoin de supervision, ou la possibilité d'inclure des dimensions symboliques).

1.5.3. Flexibilité vis-à-vis des Données Incomplètes ou Bruitées

Outre la capacité à gérer de multiples flux (voir 1.5.2), l'un des atouts majeurs du **Deep Synergy Learning (DSL)** réside dans sa **tolérance** accrue face aux **données incomplètes** ou **fortement bruitées**. Dans les approches d'apprentissage traditionnelles (réseaux neuronaux profonds inclus), on cherche souvent à **normaliser** ou **compléter** les données manquantes, voire on opte pour un **prétraitement** lourd afin de filtrer les bruits. Le DSL, de par sa **structure auto-organisée**, autorise un réseau à composer **localement** avec les lacunes et à **réajuster** ses connexions de manière à s'appuyer sur les sources les plus fiables.

Cette section (1.5.3) met en avant les principes qui confèrent au DSL sa **robustesse** et sa **souplesse** quand on fait face à des entrées partielles, hétérogènes ou bruitées.

1.5.3.1. Rôle de l'Auto-Organisation dans la Gestion du Bruit

Dans le **Deep Synergy Learning (DSL)**, les entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ s'**agglomèrent** en **clusters** (tels que décrits en section 1.4.3) lorsque leurs synergies bilatérales $S(\mathcal{E}_i, \mathcal{E}_j)$ atteignent un certain **niveau**. L'auto-organisation agit alors comme un **mécanisme** filtrant : si une entité \mathcal{E}_k se révèle trop **bruyante**, ou plus généralement peu profitable (synergie insuffisante, voire négative, avec les autres), ses connexions $\omega_{k,j}$ **dépérissent** naturellement, selon la **mise à jour** :

$$\omega_{k,j}(t+1) = \omega_{k,j}(t) + \eta [S(\mathcal{E}_k, \mathcal{E}_j) - \tau \omega_{k,j}(t)].$$

Si la synergie $S(\mathcal{E}_k, \mathcal{E}_j)$ reste **faible** ou négative, la liaison $\omega_{k,j}$ diminue itération après itération jusqu'à se rapprocher d'un **seuil** ω_{\min} en deçà duquel elle est jugée **inexistante**. L'entité \mathcal{E}_k tend dès lors à **s'isoler** de la structure principale et ne contribue plus de manière significative aux décisions ou au regroupement de clusters, limitant l'**impact** du bruit ou des mesures peu fiables sur l'ensemble du **Synergistic Connection Network**. Cette propriété confère au **DSL** une **robustesse** notable, car il n'est pas nécessaire de filtrer d'emblée les données douteuses : c'est la **dynamique** des pondérations qui effectue ce filtrage de manière locale et autonome.

De façon analogue, une entité ne possédant que des **données incomplètes** (par exemple, un capteur qui ne fonctionne que par intermittence) peut tout de même établir des **liens forts** avec d'autres entités si la partie (même partielle) de ses observations est **pertinente** et apporte un gain synergique réel. L'apprentissage ne condamne pas a priori les capteurs partiellement défaillants : il évalue la **valeur ajoutée** qu'ils procurent (section 1.4.4 sur la définition de S), et **renforce** ou **dissout** les liaisons en conséquence. Ainsi, le **DSL** teste localement l'**efficacité** des données partielles et décide, via la seule dynamique adaptative de $\omega_{k,j}(t)$, si l'entité doit être conservée dans un cluster ou rester en marge du réseau.

Ce procédé évite la nécessité d'une **stratégie** spécifique pour rejeter le **bruit** ou manipuler les **données manquantes**. Le **système** s'auto-regularise, en quelque sorte, grâce à la diminution des pondérations non rentables, ce qui se traduit par une **atténuation** de l'influence des sources imprécises. L'**auto-organisation** réalise donc une forme de **gestion du bruit** distribuée, où chaque entité (ou module) se maintient ou s'éclipse selon la **synergie** réellement engendrée avec les autres composantes du **SCN**, évitant ainsi de nuire à la **performance** globale.

1.5.3.2. Interprétation dans un Cadre Bruité

Lorsqu'une entité \mathcal{E}_k se trouve exposée à un **bruit** important dans ses représentations \mathbf{x}_k , sa **similarité** ou son **information mutuelle** avec les autres entités demeure faible ou fluctuante. On peut considérer par exemple la fonction

$$S(\mathcal{E}_k, \mathcal{E}_j) = \frac{1}{1 + \|\mathbf{x}_k - \mathbf{x}_j\|^2}.$$

Si \mathbf{x}_k varie de manière aléatoire (bruit élevé), alors $\|\mathbf{x}_k - \mathbf{x}_j\|$ reste souvent grande ou instable, réduisant ainsi la valeur de $S(\mathcal{E}_k, \mathcal{E}_j)$. En se référant à la loi de mise à jour des pondérations (section 1.4.5), on voit que

$$\omega_{k,j}(t+1) = \omega_{k,j}(t) + \eta [S(\mathcal{E}_k, \mathcal{E}_j) - \tau \omega_{k,j}(t)].$$

La valeur de $\omega_{k,j}(t)$ décroît donc si la **synergie** $S(\mathcal{E}_k, \mathcal{E}_j)$ ne compense pas suffisamment le terme de régulation $\tau \omega_{k,j}(t)$. Cette **diminution** progressive des liaisons $\omega_{k,j}$ conduit l'entité \mathcal{E}_k à s'**isoler** si le bruit la rend incohérente vis-à-vis des entités du cluster. Il n'est pas nécessaire d'introduire un algorithme d'exclusion particulier : le **DSL** gère ce phénomène de façon **auto-organisée**, en laissant le lien s'affaiblir jusqu'à potentiellement s'éteindre lorsque

$$\omega_{k,j}(t) < \omega_{\min}.$$

Dans l'hypothèse où, plus tard, \mathcal{E}_k retrouve une plus grande **fiabilité** (par exemple si un paramètre interne $\mathbf{s}_k(t)$ se recalibre, ou si le flux de données s'assainit), on observe que la **distance** $\|\mathbf{x}_k - \mathbf{x}_j\|$ peut redevenir modérée, augmentant $S(\mathcal{E}_k, \mathcal{E}_j)$. La mise à jour $\omega_{k,j}(t+1)$ peut alors reprendre une tendance à la hausse, recréant ou solidifiant des liaisons $\omega_{k,j}$. Cette **réversibilité** illustre l'une des forces du **Synergistic Connection Network** : une entité qui s'était tenue à l'écart en raison d'un bruit excessif peut retrouver sa place dès que ses **observations** redeviennent pertinentes, sans nécessiter la restructuration globale du réseau. L'entité revient alors dans le **cluster**, démontrant la **capacité** du DSL à s'**adapter** en continu à la qualité fluctuante des données.

1.5.3.3. Données Partielles : Complétion Progressive via Synergie

Dans de nombreux **scénarios** pratiques, certaines entités ne disposent que d'une **fraction** de leurs attributs. Par exemple, dans un cadre médical, un **patient** peut présenter des **données incomplètes** : quelques tests sanguins manquants, une imagerie non réalisée, ou un historique lacunaire. Les approches neuronales classiques exigent souvent une **imputation** ou un **remplissage** préalable des valeurs absentes, ce qui nécessite des hypothèses ou des méthodes de substitution. Il n'est pas rare non plus que l'on décide simplement d'**écartier** les exemples incomplets pour ne pas perturber l'apprentissage.

Dans le **Deep Synergy Learning (DSL)**, une telle entité — par exemple $\mathcal{E}_{\text{patient}}$ — est autorisée à **coopérer** partiellement avec d'autres entités. Si l'entité patient ne possède que certaines **mesures** (tests sanguins, variables cliniques) mais en ignore d'autres, elle peut néanmoins évaluer sa **synergie** avec, par exemple, d'autres patients $\mathcal{E}'_{\text{patient}}$ ou des **variables globales** (moyennes de cohortes, règles médicales) en se basant sur les **dimensions** ou **attributs** qu'elle détient réellement. Une fonction de synergie $S(\mathcal{E}_k, \mathcal{E}_j)$ adaptée aux **dimensions communes** ou à une distance partielle peut alors guider la mise à jour de la pondération :

$$\omega_{k,j}(t+1) = \omega_{k,j}(t) + \eta [S(\mathcal{E}_k, \mathcal{E}_j) - \tau \omega_{k,j}(t)].$$

Si, malgré des données incomplètes, l'entité \mathcal{E}_k dégage une **valeur ajoutée** non négligeable pour le réseau (par exemple, la portion de tests sanguins disponibles est très informative), ses **liaisons** $\omega_{k,j}$ avec d'autres entités \mathcal{E}_j se **consolident** progressivement. Cela peut se traduire par la **constitution** d'un **cluster** rassemblant d'autres patients, ou bien des entités spécialisées (un bloc fonctionnel détectant une tendance clinique). L'**incomplétude** des données n'empêche pas la **coopération** : seul importe le **gain** mesuré sur les attributs effectivement partagés.

Au fur et à mesure que l'entité \mathcal{E}_k acquiert (ou reconstitue) de nouveaux attributs manquants, il est envisageable que sa **distance** ou sa **similarité** avec d'autres entités évolue, rehaussant la pondération $\omega_{k,j}$. Le réseau y trouve un bénéfice : la réévaluation de la synergie incorpore cette **nouvelle information**, favorisant le **réajustement** des liens. On évite ainsi l'**exclusion** systématique des cas incomplets ou la nécessité d'une **imputation** globale : le **DSL** se contente de « tester » la **collaboration** sur la partie disponible des données. Cette logique de **liaison partielle** confère au **DSL** une **souplesse** précieuse pour manipuler des **jeux de données** irréguliers, tout en conservant la capacité d'un **regroupement** (cluster) auto-organisé autour des **similarités** ou **complémentarités** réellement détectées.

1.5.3.4. Exemples Concrets et Bénéfices

Dans un **environnement industriel**, certains capteurs (de température, de pression...) se révèlent souvent **défaillants** ou trop **bruités**. Dans un **Deep Synergy Learning (DSL)**, les entités correspondantes $\mathcal{E}_{\text{temp}}$ ou $\mathcal{E}_{\text{press}}$ voient leurs **pondérations** $\omega_{\text{temp},j}$, $\omega_{\text{press},j}$ diminuer dès lors que la **synergie** avec d'autres capteurs (comme $\mathcal{E}_{\text{vibration}}$ ou $\mathcal{E}_{\text{débit}}$) ne justifie plus la coopération. Le **cluster** principal, dédié par exemple à la **détection d'anomalies**, se centre alors sur les capteurs les plus fiables, sans qu'il faille explicitement exclure les capteurs défaillants : ceux-ci s'isolent **d'eux-mêmes** car leurs liens $\omega_{i,j}$ deviennent insignifiants. Si, par la suite, un capteur perturbé se **rétablissement** (ou se recèle), les pondérations se remettent à monter, lui permettant de **réintégrer** le cluster d'intérêt. Ce **mécanisme** procure une **robustesse** naturelle face aux dérives ponctuelles de certains capteurs.

Dans un autre contexte, celui d'une **base utilisateur** (profilage, recommandation), il est courant que les **profils** ne renseignent pas l'intégralité des informations attendues. En pratique, un **pipeline** neuronal classique suppose souvent une **imputation** des champs manquants ou l'exclusion des données incomplètes. Le **DSL**, au contraire, autorise l'insertion d'entités représentant « Utilisateur U1 » avec un vecteur \mathbf{x}_{U1} partiel, ainsi que d'entités « Contenu C1, C2 » définies par divers attributs. La **synergie** $S(\mathcal{E}_{U1}, \mathcal{E}_{C2})$ tient compte des attributs effectivement présents. Si, même partiellement, l'utilisateur U1 et un contenu C2 trouvent un **recouplement** significatif, la pondération $\omega_{U1,C2}$ s'accroît sans imposer que tous les champs soient renseignés. Le **réseau** s'**auto-structure** donc malgré l'incomplétude, sans avoir à multiplier les procédures d'imputation ou à écarter systématiquement les exemples incomplets. Cela permet de **préserver** une large portion d'information disponible et d'exploiter la **valeur ajoutée** dès qu'elle se présente.

1.5.3.5. Comment le DSL Surpasse la Rigidité des Réseaux Profonds

Les **réseaux neuronaux profonds** présentent une sensibilité notable aux **incohérences** ou aux cas situés hors de la distribution rencontrée lors de l'entraînement. Si, par exemple, un flux de capteurs n'a jamais inclus des observations particulièrement bruitées, le **modèle** peine à gérer ces perturbations lorsqu'elles apparaissent. En revanche, le **Deep Synergy Learning (DSL)** se caractérise par une **auto-régulation** : lorsqu'une entité \mathcal{E}_k se montre trop incertaine (synergie très basse ou négative avec d'autres entités), les pondérations $\omega_{k,j}(t)$ s'amoindrissent, et la contribution de \mathcal{E}_k se trouve mécaniquement **limitée** ou mise à l'écart, au moins de manière temporaire. Le **SCN** continue malgré tout de fonctionner avec les entités restantes, potentiellement plus fiables.

Dans bien des cas classiques, on recourt à un **modèle d'imputation** (pour combler les données manquantes) ou à des heuristiques rigides visant à traiter le **bruit**. Dans le **DSL**, à l'inverse, chaque entité **évalue** localement la **synergie** possible, sans qu'un algorithme unique de “remplissage” s'impose. Cette démarche revient à laisser chaque **liaison** $\omega_{k,j}$ apprendre la **compatibilité** ou l'**incompatibilité** entre \mathcal{E}_k et \mathcal{E}_j , de sorte que, lorsque le bruit ou l'incohérence survient, la pondération se détériore progressivement, isolant l'entité problématique du **cluster** principal.

Si la **qualité** des données associées à une entité fluctue dans le temps — qu'il s'agisse d'un capteur dont la précision varie ou d'une source intermittente de bruit — la **dynamique** d'auto-organisation gère cette évolution. Les **liaisons** $\omega_{k,j}$ s'étirent ou se resserrent selon la **synergie** présente, et aucune **réinitialisation** globale ni refonte de l'**architecture** n'est requise. Le réseau d'entités conserve sa topologie, ajustant simplement les **connexions** nécessaires. Cette **flexibilité** évite l'extrême rigidité à laquelle aboutit souvent un réseau profond traditionnel, où l'adaptation à de nouveaux modes de bruit ou d'incomplétude implique souvent un **réentraînement** sur un large jeu de données.

Conclusion

La flexibilité du DSL face aux **données incomplètes ou bruitées** repose principalement sur :

- L'**auto-organisation**, qui favorise les connexions fructueuses et exclut temporairement les sources peu fiables,
- Le **renforcement local**, où les entités évaluent directement leur synergie avec celles qui détiennent des informations compatibles,
- L'**adaptation continue**, permettant à un capteur rétabli ou à des données initialement partielles de reprendre de la valeur au sein du réseau.

Ce fonctionnement marque un écart fondamental par rapport à la plupart des **réseaux neuronaux profonds**, qui requièrent souvent une **qualité de données** homogène ou un **prétraitement** contraignant. Dans le DSL, la structure est assez **souple** pour intégrer ou ignorer (transitoirement) les entités déficientes, garantissant ainsi une forme de **robustesse** intrinsèque. Après avoir vu (1.5.2) comment le DSL absorbe naturellement la **multi-modalité**, et (1.5.3) comment il gère des données incomplètes ou bruitées, on peut poursuivre (1.5.4) sur son **potentiel d'auto-évolution** et d'adaptation continue, approfondissant encore la dimension dynamique du réseau.

1.5.4. Potentiel d'Auto-Évolution et d'Adaptation Continue

Un des aspects les plus novateurs du **Deep Synergy Learning (DSL)** concerne sa capacité à **évoluer** au fil du temps, sans nécessairement repasser par une phase d'entraînement globale et figée. Plutôt que de geler l'architecture après avoir ajusté quelques poids, le DSL propose une dynamique **en continu** où la structure interne (liens synergiques, clusters, etc.) se reconfigure régulièrement. Cette **auto-évolution** rend le réseau apte à faire face à des changements de distribution (domain shift), à l'apparition de nouvelles entités (nouvelles sources de données) ou à la nécessité d'exploiter un flux continu de données (*streaming data*). Dans cette section (1.5.4), nous examinons :

19. Les principes d'**auto-évolution** du DSL,
20. Comment cette dynamique repose sur l'**adaptation continue** des pondérations (déjà décrite en partie en 1.4.5),
21. Les bénéfices en termes de **Lifelong Learning** (apprentissage tout au long de la vie) et de **plasticité** comparable à des systèmes vivants.

1.5.4.1. L'Auto-Organisation comme Moteur d'Évolution

Dans le **Deep Synergy Learning (DSL)**, la **pondération** $\omega_{i,j}(t)$ reliant deux entités \mathcal{E}_i et \mathcal{E}_j suit la règle de mise à jour décrite en section 1.4.5. La pondération se met à jour selon

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où $S(\mathcal{E}_i, \mathcal{E}_j)$ correspond à la **synergie** (distance, similarité, co-information, etc.). Si cette **synergie** demeure suffisamment **positive**, la liaison $\omega_{i,j}$ se **renforce** ; dans le cas contraire, elle **décroît** et peut passer en deçà d'un **seuil** (voir ω_{\min} en section 1.4.5). Un lien sous ce seuil est généralement jugé **inexistant**, reflétant un **désintérêt** ou une **incohérence** mutuelle.

Ce **principe d'évolution dynamique** des connexions fait émerger des **clusters**. Deux entités qui constatent un **gain mutuel** se **rapprochent**, et des sous-groupes peuvent dès lors **fusionner** si la synergie inter-clusters croît au fil des itérations. À l'inverse, un **cluster** se **scinde** quand des sous-groupes décelant davantage de synergie interne choisissent de s'éloigner du groupe initial, leurs connexions inter-groupes ω s'abaissant peu à peu faute de bénéfice collectif.

Ce mécanisme confère au **DSL** une **dimension** “vivante” : le **Synergistic Connection Network** évolue en continu, s’**adapte** à l’arrivée de nouvelles données ou à la modification d’une source d’information, tout en **préservant** les clusters confirmés et en **permettant** l’émergence de combinaisons inédites. À aucun moment il n’est obligatoire de recommencer un **entraînement** exhaustif du réseau : la **dynamique** locale des pondérations $\{\omega_{i,j}\}$ assure la **réorganisation** progressive, mettant en place ou dissolvant les **liaisons** selon l’opportunité détectée dans les données.

1.5.4.2. Lifelong Learning et Intégration de Nouvelles Entités

Dans de nombreux environnements dynamiques (robotique, systèmes d’information évolutifs, etc.), il est courant de rencontrer des **données** ou des **entités** inédites au fil du temps. Les réseaux neuronaux profonds classiques se heurtent alors à diverses difficultés : l’ajout d’une nouvelle fonctionnalité peut réclamer la création d’un **module** annexe et un **réentraînement**, parfois complet ou via un **fine-tuning**, avec le risque de compromettre l’équilibre établi. Il arrive aussi que l’on doive réorganiser l’architecture en profondeur, ce qui affecte la stabilité du modèle.

Le **Deep Synergy Learning (DSL)** propose une alternative plus organique. Pour incorporer une **entité** $\mathcal{E}_{\text{nouvelle}}$ correspondant à un nouveau flux de données, on se contente de l’**introduire** dans le graphe existant. Les pondérations la reliant à d’autres entités $\{\omega_{\text{nouvelle},i}\}$ s’ajustent alors en fonction de la synergie détectée. Si une **coopération** apparaît avantageuse, ces liens augmentent et conduisent à l’**insertion** naturelle de l’entité dans un cluster. Il n’est donc pas nécessaire de reconfigurer toute l’architecture : seules les connexions pertinentes se consolident.

Ce mécanisme autorise une **évolution progressive**. Les clusters formés antérieurement conservent leurs **liaisons** tant qu’ils procurent un **gain** : leurs synergies internes demeurent stables. Le réseau n’est pas obligé de sacrifier ses acquis pour introduire de nouvelles entités. Ainsi, on **atténue** le phénomène de “catastrophic forgetting” caractéristique des modèles neuronaux, lesquels risquent d’effacer de vieilles connaissances lorsqu’on leur enseigne de nouvelles tâches. Le DSL, lui, ne **remplace** pas brutalement un jeu de poids par un autre : il **réorganise** plutôt les connexions de manière **locally adaptive**, maintenant la pertinence des anciens clusters et accueillant dans le graphe toute entité inédite jugée utile.

1.5.4.3. Adaptation Continue aux Changements de Distribution

Dans un **réseau** neuronal traditionnel, lorsqu’apparaît un **changement de distribution** dans les données (par exemple l’évolution des conditions de capteurs ou l’apparition de nouvelles caractéristiques), il est souvent indispensable de **réentraîner** ou de **fine-tuner** le modèle, ce qui peut provoquer un ajustement délicat, risqué pour la stabilité ou la précision déjà acquises. En **Deep Synergy Learning (DSL)**, la situation se gère plus souplement grâce à la **dynamique** auto-organisée qui régit la formation et la délaisson des connexions.

Au fil du temps, la **synergie** entre entités se réévalue en continu. Si un groupe de capteurs cesse de fournir une information fiable, les pondérations associées baissent (faible synergie), déconnectant progressivement l’entité devenue obsolète de la structure ; à l’inverse, si de nouvelles variables ou de nouvelles relations s’avèrent profitables, elles consolident leurs liaisons. Le réseau se remodelle donc de manière **locale**, sans nécessiter la rétropropagation globale ni un réapprentissage exhaustif.

Cette **localité** de la mise à jour, où chaque entité s’occupe prioritairement de ses **liens directs** (et s’ajuste indirectement via les liaisons d’autres entités), confère au **SCN** une forme de **dynamique distribuée**. Les noeuds décident eux-mêmes de renforcer ou d’affaiblir leurs coopérations, suivant l’intérêt (synergie) rencontré, sans orchestration centrale. Le système **s’adapte** ainsi de manière fluide, maintenant une relative **stabilité** pour les groupes demeurant utiles, tout en acceptant l’insertion ou la réduction de connexions pour refléter l’évolution réelle de la **distribution** des données.

1.5.4.4. Comparaison avec l’Architecture Figée d’un Réseau Profond

Dans la majorité des **réseaux neuronaux** classiques (CNN, RNN, MLP, Transformer), l’entraînement s’effectue via **rétropropagation** sur un **ensemble de données** donné a priori, aboutissant à un **modèle** fini. Lorsque la distribution

des données évolue ou qu'une nouvelle tâche survient, on se retrouve souvent contraint de réentraîner ou de fine-tuner l'architecture, avec un **risque** d'oubli des acquis précédents (catastrophic forgetting) et un **coût** en calcul potentiellement élevé. Cette rigidité résulte du fait que les **couches** et la **connectivité** du réseau demeurent invariables : les couches convolutionnelles ou self-attention restent fixes, et l'on n'a pas de mécanisme natif pour ajouter ou ôter des neurones ou des flux de données.

Le **Deep Synergy Learning (DSL)** propose une philosophie différente. La structure du **Synergistic Connection Network** est en **évolution continue**. Les entités \mathcal{E}_i peuvent mettre à jour leurs **représentations** internes $\mathbf{x}_i, \mathbf{s}_i$ et les **liaisons** $\omega_{i,j}$ s'ajustent (ou se dissolvent) selon le niveau de **synergie**. On peut donc introduire une **nouvelle entité** lorsqu'un flux de données inédits émerge, ou en ôter une qui ne sert plus, sans devoir **réentraîner** le système dans son intégralité. L'**apprentissage** n'est pas un processus qui se clôt : le **réseau** conserve un état de plasticité lui permettant de **s'adapter** en continu aux transformations de l'environnement. Cette approche relève d'un véritable **lifelong learning**, affectant aussi bien les **paramètres** que la **topologie** du SCN. C'est précisément cette capacité de reconfiguration qui différencie la souplesse du **DSL** de la rigidité d'un réseau profond figé, et qui évite de longues phases de réapprentissage global ou de coûteux correctifs pour incorporer de nouveaux concepts ou signaux.

1.5.4.5. Exemples d'Application Pratique

Il existe plusieurs scénarios où l'**approche de Deep Synergy Learning (DSL)**, avec son **réseau** auto-organisé et ses liens **adaptatifs**, se révèle particulièrement puissante pour faciliter un **apprentissage continu**. On peut citer :

Dans un contexte de **robotique autonome**, l'environnement et les capteurs peuvent évoluer : la luminosité change, des obstacles inédits surviennent ou un nouveau type de capteur (caméra additionnelle, laser de plus longue portée) est introduit. Au sein d'un **SCN**, chaque capteur ou module de traitement est représenté par une **entité** ; s'il devient peu fiable (par exemple à cause du bruit), ses connexions ω s'affaiblissent naturellement, ce qui réduit son influence. Si un nouveau dispositif émerge, il suffit de l'ajouter comme nouvelle entité $\mathcal{E}_{\text{nouvelle}}$, et la **synergie** qu'il partage avec d'autres modules s'évaluera de façon autonome. Sans reconstruire l'architecture de zéro, le **DSL** réorganise les liaisons $\{\omega_{i,j}\}$ et **réalloue** l'importance de chaque capteur selon sa **fiabilité** ou sa pertinence pour la tâche courante.

Dans un **système de recommandation évolutif**, on accueille sans cesse de nouveaux utilisateurs et de nouveaux items (films, livres, produits...). Les méthodes classiques (collaborative filtering, par exemple) requièrent souvent un **réapprentissage** complet ou un "retroufrage" à intervalles réguliers. Un **DSL**, quant à lui, peut "**brancher**" toute entité \mathcal{E}_{new} (nouvel utilisateur ou nouvel objet) au **réseau**, laissant la dynamique de **synergie** ajuster les connexions $\omega_{\text{new}, \dots}$. Les liens vers des utilisateurs ou items similaires se verront renforcés au fil des interactions, formant un **cluster** autour de la nouvelle entité. Le coût de recalculation demeure local, et le **réseau** évolue en continu, sans nécessiter un réentraînement global.

Dans l'**analyse contextuelle** (par exemple l'observation de flux de données urbaines ou sociales), la **distribution** peut changer selon l'heure de la journée, la saison, ou les événements. Un réseau traditionnel devrait incorporer différents "régimes" ou recalculer ses poids quand on bascule de "jour" à "nuit" ou d'"hiver" à "été". Dans un **SCN**, les **clusters** pertinents se forment pour un contexte donné, puis les pondérations ω se réactualisent quand le contexte se modifie. Certains sous-groupes se dissolvent si la synergie n'est plus présente, d'autres réapparaissent, et le **DSL** peut ainsi commuter de manière fluide d'un **cluster** à l'autre en fonction de la situation, sans devoir se relancer dans un **apprentissage** lourd.

Conclusion

Le **potentiel d'auto-évolution** et d'**adaptation continue** du DSL le dote d'une **plasticité** rarement vue dans les architectures d'apprentissage classiques. Son **paradigme** :

- Ne gèle **pas** les liens après entraînement,
- **Accepte** l'arrivée de nouvelles entités ou la disparition d'entités obsolètes,

- **Réajuste** la structure pour épouser l'évolution des données ou du contexte,
- **Garantit** la conservation de l'existant (évite l'oubli brutal) s'il reste utile.

Cette **flexibilité** soutient une vision “**lifelong learning**” où le réseau se maintient en **auto-organisation** permanente, capable d'absorber et d'exploiter des changements de distribution ou des sources inédites. On verra, dans les prochaines sections (1.5.5 à 1.5.7), d'autres arguments majeurs pour une approche synergique (diminution de la supervision, représentations plus riches, intégration de dimensions symboliques).

1.5.5. Réduction de la Dépendance à la Supervision Humaine

Au-delà de la multi-modalité (1.5.2), de la gestion des données incomplètes (1.5.3) et de l'adaptation continue (1.5.4), un autre avantage essentiel du **Deep Synergy Learning (DSL)** réside dans sa **relative autonomie** vis-à-vis d'une supervision humaine intensive. Les méthodes traditionnelles de l'IA, notamment les réseaux neuronaux profonds, reposent souvent sur l'accès à d'importants **jeux de données labellisés**, et requièrent un entraînement supervisé (ou semi-supervisé) afin d'ajuster des poids internes. Or :

- Préparer ces labels ou configurations peut être très **coûteux** et **chronophage**,
- L'absence de labels, ou la nécessité d'un apprentissage plus **auto-dirigé**, se fait de plus en plus ressentir dans des contextes réels (big data non étiqueté, environnements inconnus, etc.).

Le **DSL**, du fait qu'il encourage des **relations auto-organisées** entre entités, apporte des mécanismes intrinsèques pouvant réduire la besoin d'étiquettes externes. Cette section (1.5.5) explique :

22. Comment le DSL peut s'**auto-structurer** en s'appuyant sur la synergie plutôt que sur un label,
23. En quoi cela diminue la **dépendance** à la supervision,
24. Quelles **implications** pour l'IA autonome ou non supervisée.

1.5.5.1. L'Auto-Organisation sans Label

Dans les paradigmes d'**apprentissage supervisé**, on dispose en général pour chaque donnée \mathbf{x} d'un **label** y (classe, valeur numérique à prédire), et l'entraînement consiste à **minimiser** une fonction de coût $\mathcal{L}(f_\theta(\mathbf{x}), y)$. On **ajuste** alors les poids du réseau neuronal pour améliorer la correspondance entre l'entrée \mathbf{x} et la sortie souhaitée y .

Dans le **Deep Synergy Learning (DSL)**, la dynamique s'appuie sur la **synergie** entre paires d'entités \mathcal{E}_i et \mathcal{E}_j . On définit une quantité $S(\mathcal{E}_i, \mathcal{E}_j)$ qui mesure le **gain** (ou la complémentarité) qu'elles retirent de leur coopération. Cette fonction de synergie peut reposer sur une **distance** ou une **similarité**, sur la **co-information** en théorie de l'information, ou encore sur tout **critère** mesurant l'utilité mutuelle. Dans un **réseau** purement non supervisé, on ne dispose d'aucune annotation : la **structure** émerge alors de la manière dont les entités s'**agrègent** ou se **séparent** en fonction de leur synergie locale. Un **objectif** global, le cas échéant, peut exister si un signal partiel ou une contrainte externe oriente la cohérence, mais il n'est pas obligatoire. L'auto-organisation résulte de la **conjonction** des ajustements locaux, sans référence à un label y .

Cette logique se reflète dans la **formation** de clusters (section 1.4.3). Les entités se regroupent par **renforcement** de leurs connexions si elles détectent un **gain**, et se dissocient sinon. Il n'est pas nécessaire de leur **imposer** des étiquettes comme “classe C1” ou “classe C2” : elles se rassemblent d'elles-mêmes, **non supervisées**, autour de la synergie perçue. Le processus aboutit à un **clustering** naturel, que l'on peut ensuite interpréter comme un groupement de similarités ou de dépendances, et qui peut se révéler **faiblement supervisé** si l'on dispose, en parallèle, de quelques **signaux** ou **contraintes** externes. Dans ce cadre, le **DSL** encourage un **apprentissage local** des coopérations, structurant le **Synergistic Connection Network** sans recourir à un label global pour chaque exemple.

1.5.5.2. Quand un Signal de Supervision Existe...

Le Deep Synergy Learning (**DSL**) n'exclut pas l'éventualité de **labels** ou d'**objectifs** supervisés. Au contraire, il est tout à fait envisageable de disposer d'un **objectif** supervisé pour prédire un label y , tout en conservant le mécanisme d'**auto-organisation** qui ajuste les liens $\omega_{i,j}$ en fonction de la synergie. Dans ce scénario, le **DSL** cohabite avec un algorithme de rétropropagation ou d'autre forme de supervision, mais n'est pas entièrement asservi à cet unique signal étiqueté.

Concrètement, on peut imaginer qu'un **objectif** tel que la minimisation d'une fonction de perte $\mathcal{L}(f_\theta(\mathbf{x}), y)$ se déroule en parallèle de la **dynamique** locale guidée par la synergie. Les **liaisons** $\omega_{i,j}(t)$ poursuivent la règle adaptative (section 1.4.5), tandis que le **label** y n'intervient que pour évaluer la cohérence globale ou orienter certains choix de structure. Une partie du réseau peut ainsi se spécialiser dans la tâche supervisée, la **synergie** jouant un rôle déterminant dans l'**organisation** des entités restantes.

Cette architecture présente un **double** bénéfice. D'abord, l'**auto-organisation** ne dépend pas nécessairement du label y , ce qui autorise un fonctionnement **semi-supervisé** ou même **non supervisé** : la plupart des liaisons $\omega_{i,j}$ se mettent en place en fonction de la **coopération** intrinsèque, et l'objectif supervisé n'est mobilisé qu'en appoint. Ensuite, si le nombre de labels est **limité**, l'apprentissage reste largement alimenté par la **synergie** interne, évitant la nécessité de disposer d'un jeu de données richement annoté. Le **réseau** peut ainsi, en un premier temps, **structurer** ses entités par pure auto-organisation, puis tirer parti de quelques **labels** additionnels pour affiner la prédiction d'une variable ou la détection d'anomalies.

L'important est que les **labels**, quand ils existent, ne contrôlent pas toutes les **liaisons** du réseau. Seule une fraction des entités ou des connexions peut être orientée par la performance supervisée, tandis que les **clusters** naissent en grande partie de la **synergie** (sections 1.4.3 et 1.4.4). Le réseau se montre donc apte à **apprendre** et **s'adapter** via son auto-organisation, puis à **exploiter** un signal supervisé pour ajuster un module final ou un sous-ensemble d'entités, évitant ainsi la contrainte d'une rétropropagation exhaustive sur toute l'architecture.

1.5.5.3. Exemple : Découverte Spontanée de Catégories

Pour illustrer la manière dont un Deep Synergy Learning (**DSL**) peut structurer les données **sans** s'appuyer sur des labels explicites, on peut considérer un ensemble de **documents** non annotés. Chaque document est alors représenté par un **embedding** ou un **bag-of-words**. Dans le **Synergistic Connection Network**, on introduit autant d'**entités** \mathcal{E}_{doc} que de documents, en laissant le mécanisme de **synergie** déterminer comment ces entités se relient.

Les entités commencent par **évaluer** leur proximité ou leur complémentarité : des vecteurs \mathbf{x}_{doc} partageant des thèmes (mots récurrents, vocabulaire semblable, distributions de topics) peuvent afficher une **synergie** notable. Les pondérations $\omega_{i,j}$ entre entités se renforcent si la **similarité** est jugée élevée, ou si la co-information (section 1.4.4) révèle un **gain** dans la mise en commun de leurs contenus. Progressivement, un ou plusieurs **clusters** apparaissent, chaque cluster rassemblant des documents sur des sujets proches, sans qu'on ait besoin de spécifier de labels ou de classes. Ainsi, certains regroupements peuvent porter sur la **politique**, d'autres sur le **sport**, d'autres encore sur la **santé**, mais rien n'impose que le DSL les nomme ou les identifie explicitement.

Si, plus tard, un **expert** fournit quelques **labels** indiquant, par exemple, qu'un certain cluster correspond au domaine sportif, on n'a pas besoin d'annoter l'**intégralité** des documents. Le **DSL** s'était déjà **auto-organisé** en clusters cohérents : il suffit de "taguer" postérieurement la communauté d'entités concernée. Ce procédé montre comment un système **non supervisé** ou **faiblement supervisé** peut accomplir une **structuration** fine des données, sur laquelle un label partiel ne fait qu'ajouter un niveau de **description** plus explicite. Les sections antérieures (1.4.3, 1.5.5.1 et 1.5.5.2) soulignent ainsi la flexibilité du DSL pour fonctionner dans des contextes sans labels, tout en permettant une intégration de signaux supervisés quand ils existent.

1.5.5.4. Vers une IA plus Autonome

Les **interactions indirectes** (discutées en section 1.4.6) font en sorte qu'une entité peut être **influencée** par une autre, même en l'absence de lien direct, via des chemins de coopération au sein du **Synergistic Connection Network**. Ce mécanisme enrichit le potentiel d'**auto-découverte** de catégories ou de concepts de manière **non supervisée**, puisque l'information peut circuler librement à travers divers **chemins**, et les **clusters** émergent en valorisant l'intégralité des signaux disponibles.

Dans une perspective de **IA** plus générale, appelée à affronter des **domaines inconnus** et des **contextes** évolutifs, la **réduction** du besoin de supervision revêt un caractère décisif. Le **Deep Synergy Learning (DSL)**, en misant sur un **apprentissage local** et **continu**, se montre apte à fonctionner avec des annotations partielles ou inexistantes, tout en conservant la capacité de **structurer** et **organiser** les entités. Il s'agit d'une étape vers une **IA** plus autonome, qui, sans nécessité d'un étiquetage exhaustif ou de guides fortement supervisés, s'adapte aux environnements **faiblement annotés** (voire non annotés) et continue à explorer la structure interne des données pour faire émerger **concepts** et **catégories** spontanément.

1.5.5.5. Limites et Solutions Partielles

Le **Deep Synergy Learning (DSL)** ne s'appuie pas massivement sur les **labels**, ce qui lui confère une grande liberté dans la formation des **clusters** et dans la découverte de structures internes. Néanmoins, un **petit** signal externe, issu par exemple d'une **semi-supervision**, peut être requis lorsque l'on souhaite orienter la **dynamique** vers des objectifs spécifiques. Sans cette impulsion, le réseau se contente de former des groupes qui font sens pour les données elles-mêmes, sans garantir qu'ils correspondent aux besoins d'une application particulière. Il peut arriver que les clusters, très cohérents sous un angle statistique, ne concordent pas avec la segmentation ou les catégories souhaitées par l'utilisateur.

L'**auto-organisation** peut en effet aboutir à des **partitions** intrinsèquement pertinentes, mais qui ne recouvrent pas exactement les **objectifs** fixés en pratique. Pour rectifier ce phénomène, il est possible d'introduire un **minimum** de **contraintes** ou de **labels** annotés, lesquels jouent le rôle de **pénalités** ou de **feedback** dans la mise à jour des liaisons $\{\omega_{i,j}(t)\}$. De la sorte, le **DSL** ne se limite plus à la seule synergie intrinsèque, mais tient également compte de la finalité appliquée (une tâche métier, une classification imposée), ce qui oriente plus étroitement la **constitution** des clusters.

La **complexité** du calcul de la synergie, surtout pour des données brutes très dimensionnelles, constitue un autre défi. Dans certains cas, on envisage des mesures plus poussées telles que la **co-information** ou la **Partial Information Decomposition** (sections 1.4.4 et 1.4.7), dont l'estimation peut s'avérer lourde. Des **estimateurs** non paramétriques requièrent souvent un échantillonnage important et des ressources conséquentes. Il est alors crucial de se doter de **stratégies** d'approximation ou d'**échantillonnage** (sampling parcimonieux, heuristiques de calcul) pour maintenir la faisabilité sur des volumes de données massifs (big data). Cette solution, certes partielle, demeure incontournable si l'on veut étendre le **DSL** à des scénarios industriels exigeants, tout en préservant un **coût** de calcul acceptable.

Conclusion

Le **Deep Synergy Learning** offre un **cadre** qui diminue la **dépendance** à la **supervision humaine** en se basant principalement sur :

- Des **fonctions de synergie** (distance, similarité, information mutuelle, etc.) pour guider la structuration,
- Une **auto-organisation** où les entités coopèrent ou se séparent localement,
- La possibilité d'un **apprentissage** non supervisé, ou faiblement supervisé, via la croissance / décroissance des pondérations $\omega_{i,j}$.

Cette **indépendance** accrue vis-à-vis d'un étiquetage exhaustif, couplée à la **plasticité** et à l'**adaptation continue** (1.5.4), fait du **DSL** un **candidat** privilégié pour les systèmes d'**IA** évolutifs, explorant des environnements peu annotés

ou devant se reconfigurer en permanence. Par la suite (1.5.6, 1.5.7), nous verrons comment cette approche peut encore être renforcée pour créer des représentations **riches** et éventuellement **cognitives**, en incorporant des idées de **conceptualisation** ou de **symbolique**.

1.5.6. Création de Représentations Riches et plus Interprétables

En plus de la **réduction de la dépendance à la supervision** (1.5.5), le **Deep Synergy Learning (DSL)** présente un avantage notable pour la **création de représentations** à la fois **riches** et **interprétables**. Contrairement aux réseaux neuronaux profonds classiques, qui produisent souvent des représentations “boîte noire” difficilement explicables, le DSL mise sur l'**auto-organisation** et la **coopération** entre entités d’information, ce qui peut aboutir à des structures internes (clusters, macro-clusters, liens synergiques) plus compréhensibles pour un humain. Cette section (1.5.6) met en évidence :

25. **Pourquoi** la synergie favorise l’émergence de **représentations complexes**,
26. Comment le principe de **clusters** ou de **macro-clusters** rend l’organisation plus **lisible**,
27. En quoi le DSL peut faciliter l’**interprétabilité** par rapport aux architectures profondes traditionnelles.

1.5.6.1. Logique “par Entités” plutôt que “en Couches Opaques”

Dans les **réseaux neuronaux profonds** (CNN, RNN, Transformers...), la **représentation** interne s’étage à travers plusieurs **couches** successives de transformations non linéaires. Malgré certaines avancées, telles que la visualisation de **filtres** dans un CNN ou l’analyse de **matrices d’attention** dans un Transformer, comprendre comment le réseau aboutit à sa décision globale demeure un défi majeur. Les poids ne sont pas naturellement groupés sous forme de blocs “légibles” et, hors du cadre spécifique d’un neurone ou d’une couche, il est difficile de rattacher un sous-ensemble de paramètres à un **concept** clairement identifiable. Les techniques d’**interprétabilité** (comme Grad-CAM ou LIME) tentent de mettre en évidence l’influence de certains pixels ou de certaines dimensions, mais elles proposent souvent des **heuristiques** ne fournissant pas nécessairement une vue d’ensemble de la logique interne.

Le **Deep Synergy Learning (DSL)** adopte une approche sensiblement différente en structurant l’information autour d’**entités** (nœuds du **Synergistic Connection Network**). Chaque entité peut représenter un **flux sensoriel** ou un **descripteur** plus abstrait (embedding, concept partiel). Les **liaisons** entre entités indiquent la **synergie** détectée, c’est-à-dire la plus-value mutuelle de leur coopération. Ainsi, la représentation n’est plus seulement un pipeline linéaire ou une série de couches opaques : elle se formalise comme un **graphe** de modules où l’on identifie plus aisément les fragments d’information et où l’on constate quelles entités collaborent avec quelles autres. Les **clusters** qui émergent se trouvent alors naturellement interprétables : un sous-groupe peut regrouper la “détection de visages” (vision), l’“audio de voix” (son) et un “module de reconnaissance textuelle” (parole transformée en mots-clés), offrant un tableau beaucoup plus modulaire et segmenté qu’un simple empilement de couches neuronales classiques.

Cette logique “par entités” rend donc la représentation moins opaque : on sait, au sein du **SCN**, quelles entités existent, quelles sont leurs fonctions ou données principales et comment elles s’**agencent** via les pondérations de synergie. Le **DSL** propose ainsi un degré d’**explicabilité** supplémentaire, puisque la structure d’ensemble apparaît comme un **réseau** plus proche d’une **cartographie** du flux d’information, plutôt qu’un empilement difficile à démêler d’unités et de poids dispersés.

1.5.6.2. Clusters et Macro-Clusters comme Briques de Sens

Dans le **Deep Synergy Learning (DSL)**, lorsqu’un **cluster** d’entités se forme (section 1.4.3), il peut refléter un **concept** ou un **thème** commun aux données. Imaginons un **cluster** multimodal $\{\mathcal{E}_{\text{image}}, \mathcal{E}_{\text{audio}}, \mathcal{E}_{\text{texte}}\}$ qui se spécialise dans la détection d’un “événement conférence” parce que l’entité image renvoie à une scène d’orateur faisant face à un public, l’entité audio capture un bruit de voix ou d’applaudissements, et l’entité texte détecte des mots comme “bienvenue” ou “question”. La **cohérence** ainsi dégagée à l’intérieur du cluster n’est pas un simple effet de poids numériques : on

sait **quelles** entités s’agrègent, **pourquoi** (à travers la synergie évaluée localement), et cela confère à l’ensemble une **lisibilité** accrue.

Du point de vue de l'**explicabilité** ou de l’“Explainable AI”, un **cluster** dans le **Synergistic Connection Network** n’équivaut pas à un bloc opaque de neurones comme dans un réseau profond traditionnel. Il s’agit plutôt d’un **sous-graphe** composé d’entités portant une signification distincte (capteurs visuels, flux audio, modules textuels, ou encore vecteurs symboliques...). Les **liaisons synergiques** $\omega_{i,j}$ reflètent la force de leur coopération : on peut par exemple examiner la distribution de la co-information ou de la similarité entre chaque paire. Les regroupements (macro-clusters) où plusieurs clusters se **fusionnent** indiquent qu’un ensemble plus large d’entités se renforcent réciproquement dans leur coopération. Dans un **contexte e-commerce**, on pourrait alors voir un macro-cluster associer plusieurs catégories de produits, des groupes d’utilisateurs et des attributs de profil, dessinant un **sous-réseau** particulièrement actif et riche en **synergie**.

Ces **briques** (clusters, macro-clusters) constituent autant de **modules** que l’on peut **interpréter** et **commenter** : chaque entité, de par son lien avec un flux de données ou un vecteur caractéristique particulier, donne un **ancrage** explicite à son rôle. Les liens montrent la proportion de **synergie** interne, et le **DSL** scinde le graphe global en plusieurs **sous-ensembles** correspondant à des **concepts** ou **sujets**. Cette granularité rend la représentation plus modulaire, invitant à un examen plus détaillé de l’organisation du réseau et offrant des pistes de justification plus tangibles que la simple consultation de poids dans une couche neuronale abstraite.

1.5.6.3. Représentations Multiples et Non Linéaires

Le **Deep Synergy Learning (DSL)** ne se borne pas à des représentations linéaires ou statiques. Au contraire, chaque **entité** \mathcal{E}_i peut conserver une **représentation interne** $\mathbf{x}_i(t)$ ou $\mathbf{s}_i(t)$ qui évolue au fil du temps, ou être associée à un petit réseau neuronal local (par exemple un autoencodeur) calculant

$$\mathbf{x}_i(t+1) = F_i(\mathbf{x}_i(t), \{\omega_{i,k}(t)\}, \dots).$$

La **non-linéarité** inhérente à ce type de module (RNN local, MLP, autoencodeur) élargit considérablement l’espace des **représentations** envisageables, comparativement à un simple cadre vectoriel fixe. Les entités peuvent ainsi affiner leurs **features** internes ou leurs **états** \mathbf{s}_i , tout en mettant à jour leurs **liaisons** $\omega_{i,j}(t)$ d’après la **synergie** nouvellement constatée. Cette approche ne constitue pas un pipeline rigide : chaque entité, considérée comme un bloc fonctionnel autonome, maintient sa propre logique d’évolution, tandis que la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ se réévalue localement.

Si deux entités en viennent à **coïncider** fortement (en partageant des représentations très similaires ou très complémentaires), leurs pondérations $\omega_{i,j}$ peuvent atteindre un niveau **saturé**, conduisant potentiellement à une **fusion** (au sens de la fusion de clusters, section 1.4.3). Inversement, d’autres entités peuvent demeurer **indépendantes**, préférant garder leurs liaisons minimales. Le résultat global est une **représentation** dite “multi-foyers”, où plusieurs **clusters** se spécialisent dans des sous-ensembles de la tâche ou des données. Chacun possède son propre noyau fonctionnel, et ils coopèrent occasionnellement si la **synergie** l’exige. Cette organisation **modulaire** tranche avec un réseau profond monolithique dans lequel toutes les informations finissent entremêlées au sein de couches successives. Elle favorise un **équilibre** entre la **richesse** (via la non-linéarité) et la **lisibilité** (chaque entité reste limitée à un bloc fonctionnel clairement cerné), tout en laissant l’auto-organisation diriger la consolidation ou la séparation des diverses composantes.

1.5.6.4. Comparaison avec l’Explicabilité dans les Réseaux Profonds

Dans les approches classiques d'**explainability** appliquées aux réseaux neuronaux profonds (CNN, Transformers, etc.), il est habituel de recourir à des méthodes “post-hoc”, comme la visualisation d’**activation maps**, l’explication locale (LIME, SHAP) ou l’examen des **attention maps** dans un Transformer. Ces techniques fournissent un aperçu de la raison pour laquelle le réseau met en avant tel pixel, telle dimension ou tel token, mais elles ne suppriment pas pour autant le caractère opaque de la structure interne. Un CNN reste un empilement de couches, au sein desquelles la signification des poids demeure largement cachée, et un Transformer recèle une superposition de blocs d’attention dont la lisibilité globale n’est pas toujours aisée.

Le Deep Synergy Learning (DSL) offre une **explication intrinsèque** plus directe, car l'information s'organise en entités reliées par des **liaisons** $\omega_{i,j}$ qui traduisent la synergie entre ces entités. Le graphe obtenu est alors lisible : on distingue les **clusters** et macro-clusters correspondant à des groupes d'entités coopératives, et chaque entité garde une identité claire (modalité, bloc fonctionnel, concept partiel). L'utilisateur peut examiner un cluster C pour déterminer quelles entités le composent, quelles caractéristiques portent chacune, comment se distribuent les pondérations $\omega_{i,j}$. De là, il est possible de déduire la logique d'assemblage : on voit, par exemple, qu'un sous-graphe associe la détection de panneaux de signalisation à l'enregistrement de la vitesse et au suivi de la trajectoire, ce qui éclaire la raison d'être de ce regroupement. Un tel niveau de transparence demeure rarement accessible dans un réseau profond standard, où l'on ne peut guère rattacher des ensembles de neurones internes à une fonction ou à un concept identifié, si ce n'est par l'ajout ultérieur d'artifices explicatifs. Le DSL rend la structure explicite et modulable, montrant clairement **qui** coopère avec **qui**, et dans quelle mesure, par l'intermédiaire de pondérations synergiques réparties sur l'ensemble du **Synergistic Connection Network**.

1.5.6.5. Exemple d'Application : Représentation Sémantique Évolutive

Dans un **agent conversationnel**, plusieurs **entités** peuvent correspondre à différents volets de l'interaction : ainsi, une entité $\mathcal{E}_{\text{linguistique}}$ traite la compréhension du langage, une entité $\mathcal{E}_{\text{contexte}}$ se réfère aux informations contextuelles (heure, lieu, historique de conversation), et une entité $\mathcal{E}_{\text{profil utilisateur}}$ incarne les préférences ou les caractéristiques de l'utilisateur. Au fil des échanges, les synergies entre ces composantes peuvent croître lorsque, par exemple, le $\mathcal{E}_{\text{profil}}$ apporte des indices qui concordent avec $\mathcal{E}_{\text{linguistique}}$. Le **DSL** favorise alors la formation d'un **cluster** $\{\mathcal{E}_{\text{ling}}, \mathcal{E}_{\text{context}}, \mathcal{E}_{\text{profil}}\}$ dont la **coopération** explicite se traduit par la hausse des pondérations $\omega_{i,j}$. Il en résulte un "sous-modèle" sémantique plus clair, articulé autour des thèmes, des intentions ou des préférences récurrentes du locuteur. En examinant la façon dont les liens se sont construits, on peut identifier les **topics** qui émergent et comprendre comment s'est bâtie cette synergie.

Dans un autre registre, l'**analyse d'images documentaires** peut tirer profit de ce principe. Certaines **entités** $\mathcal{E}_{\text{patch}}$ décrivent les patchs ou segments visuels (éléments d'architecture, styles picturaux), alors que d'autres entités regroupent des **mots-clés** extraits de légendes ou de métadonnées, ou encore des **concept**s plus abstraits liés à une époque ou un auteur. Il se crée ainsi un **cluster** combinant, par exemple, un patch visuel P1, un ensemble de mots-clés M2 et une entité conceptuelle C3 qui indique la période médiévale. L'auto-organisation du **Synergistic Connection Network** montre que la **co-information** est élevée : les motifs visuels rappellent l'architecture gothique, les mots-clés renvoient à un style médiéval, et la dimension conceptuelle (C3) représente cet intervalle historique. La **coopération** entre ces entités confère une **lisibilité** accrue : on identifie un regroupement signifiant « document archéologique médiéval » sans avoir dû imposer de règle au préalable. Le **DSL** opère donc comme un **pilier** sémantique dynamique, consolidant les liens lorsqu'il perçoit un bénéfice mutuel et autorisant une **représentation** riche et modulable.

Conclusion

La **création de représentations riches et plus interprétables** fait partie des **points forts** du DSL, grâce à :

- Son **organisation par entités** (nœuds clairs, porteurs d'une modalité ou d'une fonction),
- Sa **clusterisation** auto-organisée, permettant d'identifier *qui* coopère avec *qui*,
- La **possibilité** d'observer directement les liaisons $\omega_{i,j}$ et leurs pondérations,
- Un **niveau de modularité** qui dépasse l'approche "monobloc" des réseaux neuronaux profonds.

De ce fait, le DSL se prête mieux à des besoins de **transparence**, de **compréhension** et de **diagnostic**. Couplée aux atouts précédents (multi-modalité, gestion du bruit, adaptation continue, moindre dépendance à la supervision), cette caractéristique renforce l'idée que l'approche synergique peut offrir une **IA plus agile**, pouvant aboutir à des **configurations** internes plus **accessibles** à l'analyse humaine. Dans la dernière section (1.5.7), nous verrons un autre

aspect crucial : l'**intégration** de **dimensions symboliques** ou **cognitives** dans un cadre DSL, promettant une forme encore plus avancée d'**IA unifiée**.

1.5.7. Intégration de Dimensions Symboliques ou Cognitives

Les sections précédentes (1.5.1 à 1.5.6) ont souligné diverses forces du **Deep Synergy Learning (DSL)**, telles que la **capacité à gérer la multi-modalité**, à s'**adapter** en continu, ou à **créer** des représentations plus **lisibles**. Un autre point capital, souvent abordé dans l'IA contemporaine, est la possibilité de **mélanger** des composantes **sub-symboliques** (issues des méthodes connexionnistes) et des composantes **symboliques** (notions logiques, sémantiques, règles expertes). Alors que les réseaux neuronaux profonds "classiques" ont généralement peiné à intégrer ces dimensions symboliques, le **DSL**, par sa structure en entités et liens synergiques, peut être un **cadre** propice pour unir symbolique et sub-symbolique.

Cette section (1.5.7) explique :

28. Comment le DSL autorise l'**introduction** d'entités à **portée symbolique** (règles, concepts, modèles cognitifs),
29. Pourquoi cette intégration est **plus fluide** que dans un réseau hiérarchique figé,
30. Les **applications** possibles (raisonnement, inférence, cognition proche de l'humain),
31. Les **défis** mathématiques associés à la cohabitation de ces deux modes de représentation.

1.5.7.1. Aperçu des Approches Symboliques et Connexionnistes en IA

Dans le domaine de l'**IA**, deux grands courants se distinguent historiquement. Le premier, souvent qualifié de **IA symbolique**, s'appuie sur des **représentations logiques** et des **moteurs d'inférence** capables de manipuler des symboles pour résoudre des problèmes. Cette approche offre une **explicabilité** naturelle, car les **règles logiques** sont décrites dans un langage clair et les chaînes de raisonnement peuvent être retracées. Il en découle une forme de **transparence** : on peut savoir précisément pourquoi une conclusion a été tirée. Cependant, ce paradigme symbolique se heurte à plusieurs limites. Il gère mal l'**incertitude** et se révèle peu adapté à l'**apprentissage** automatique à partir de données massives ou bruitées. L'idée d'extraire directement des **motifs** complexes d'un grand ensemble d'observations, sans intervention humaine pour écrire les règles, demeure délicate dans ce cadre.

À l'opposé, les méthodes **connexionnistes**, dont les **réseaux neuronaux** constituent l'archétype, excellent dans l'**apprentissage** à partir de larges volumes de données hétérogènes. Elles reposent sur des **paramètres** réels ajustés par descente de gradient ou par d'autres algorithmes, permettant de capturer des régularités complexes dans l'**espace** des entrées. L'exemple canonique est la classification d'images, où un réseau convolutif identifie des motifs tout en s'ajustant aux écarts et aux bruits des données. Toutefois, ce cadre connexionniste se heurte à une **opacité** interne, souvent résumée par l'expression "boîte noire". Les poids et activations qui conduisent à la décision sont difficiles à **expliquer**, et la notion de **règles logiques** ou de **concepts** abstraits y est absente dans la formulation de base. Il n'existe pas de mécanisme formel de **raisonnement** symbolique, ni de vérification de **cohérence** par rapport à un ensemble de connaissances explicites.

Dans les sections suivantes, le **Deep Synergy Learning (DSL)** sera envisagé comme un **cadre unificateur** susceptible d'exploiter les forces des deux approches. L'accent sera mis sur la manière de tirer parti de la **plasticité** et de la **puissance** d'apprentissage sub-symbolique tout en intégrant, au sein d'un **Synergistic Connection Network**, des **entités** représentant des **règles** ou **concepts** symboliques. Ainsi, on peut espérer concilier l'**adaptation** continue des réseaux neuronaux avec la **lisibilité** et la **transparence** associées aux logiques symboliques.

1.5.7.2. Principes d'une Approche Hybride Symbolique–Connexionniste

Dans la continuité des sections précédentes, et en particulier au regard des principes généraux du **Deep Synergy Learning (DSL)**, on constate que les approches **symboliques** et **sub-symboliques** apparaissent de plus en plus **complémentaires**. Les récentes avancées en **IA** soulignent cette complémentarité : d'un côté, les méthodes sub-symboliques (réseaux neuronaux classiques, **DSL** et paradigmes voisins) démontrent une **puissance d'apprentissage** remarquable lorsqu'il s'agit de traiter de grandes quantités de données, ou d'extraire des **caractéristiques** non triviales dans un **espace** de dimension élevée. De l'autre, les méthodes symboliques (appuyées sur des **logiques**, des **règles** et des **ontologies**) assurent une **rigueur** et une **explicabilité** supérieures, notamment lorsqu'il convient de formuler des raisonnements structurés ou de décrire des **connaissances** de manière **interprétable**.

La question est donc de **concevoir** une architecture **hybride** qui mobilise simultanément la **puissance d'apprentissage** sub-symbolique et la **cohérence symbolique**. Dans le cadre d'un **Synergistic Connection Network (SCN)** tel que défini dans les sections précédentes, il est possible d'**introduire** des entités de différents types. Il est ainsi envisageable de faire cohabiter des **entités** $\mathcal{E}_{\text{symb}}$ représentant des **concept logiques**, des **règles formelles**, ou des **faits** structurés, et des **entités** $\mathcal{E}_{\text{data}}$ se fondant sur des **vecteurs sub-symboliques** $\mathbf{x}_i \in \mathbb{R}^d$. On peut imaginer que la représentation symbolique contienne des énoncés tels que « Un véhicule se déplace sur roues » ou « Si la température excède 100°C, on suspecte une surchauffe », tandis que la partie sub-symbolique gère des **flux de données** (images, séries temporelles, signaux sensoriels) ou des **features** issues de réseaux de neurones.

Dans cette optique, on peut noter $\mathcal{E}_{\text{symb}}$ pour une **entité symbolique** et $\mathcal{E}_{\text{data}}$ pour une **entité sub-symbolique**. La **pondération** $\omega_{\text{symb},\text{data}}(t)$ s'adapte alors par la règle classique du **DSL** :

$$\omega_{\text{symb},\text{data}}(t+1) = \omega_{\text{symb},\text{data}}(t) + \eta [S(\mathcal{E}_{\text{symb}}, \mathcal{E}_{\text{data}}) - \tau \omega_{\text{symb},\text{data}}(t)],$$

où la **synergie** $S(\mathcal{E}_{\text{symb}}, \mathcal{E}_{\text{data}})$ traduit à quel point la **règle** ou le **concept** véhiculé par $\mathcal{E}_{\text{symb}}$ s'applique utilement au **contenu** sub-symbolique porté par $\mathcal{E}_{\text{data}}$. La valeur de S peut tenir compte du **taux de succès** lors de la confrontation aux données, ou évaluer la **compatibilité** sémantique entre la règle symbolique et les caractéristiques identifiées dans le **flux sub-symbolique**.

On peut approfondir cette **coopération** à l'aide de **formules** décrivant la mise en correspondance entre un **concept** symbolique et des **features** extraites. Par exemple, on peut écrire :

$$S(\mathcal{E}_{\text{symb}}, \mathcal{E}_{\text{data}}) = \text{score} \left(\text{matching}(\text{rules}(\mathcal{E}_{\text{symb}}), \mathbf{x}_{\text{data}}) \right),$$

où $\text{rules}(\mathcal{E}_{\text{symb}})$ dénote l'ensemble des règles ou faits associés à la représentation symbolique, et \mathbf{x}_{data} la **représentation vectorielle** de l'entité sub-symbolique. La fonction **matching** évalue la cohérence entre la **logique symbolique** et les **observations** sub-symboliques, et **score** traduit le **gain** ou la **conformité** obtenu.

Dans cette approche **hybride**, il y a un **avantage** notable. Le réseau **auto-organisé** du **DSL** peut **identifier** quelles **règles** ou **concept**s sont les plus pertinents pour tel **type de données**, sans nécessiter une supervision massive. Les entités symboliques, en s'activant sur certaines conditions, peuvent **expliquer** plus aisément les **décisions** ou **groupements** effectués. La **transparence** s'en trouve améliorée : si la **pondération** $\omega_{\text{symb},\text{data}}$ monte en régime, cela indique que la règle symbolique s'applique fréquemment et avec succès à la modalité de données considérée.

La **limite** principale repose sur la **cohérence** nécessaire entre la **logique** et les **données**. L'entité symbolique doit être suffisamment bien définie et adaptée à la réalité sub-symbolique ; en cas de décalage, la pondération associée risque de rester faible, rendant la règle peu intégrée. Sur le plan de l'implémentation, il faut également gérer la **complexité** liée à l'activation symbolique, potentiellement coûteuse s'il faut évaluer de nombreuses règles sur un flux volumineux de données.

Dans ce **paradigme**, on constate néanmoins que la **puissance** du **DSL** est préservée : le réseau peut se reconfigurer en continu, renforçant ou affaiblissant les liens entre entités symboliques et sub-symboliques selon la **synergie** détectée. En parallèle, les entités symboliques aident à **structurer** le raisonnement et à **rendre compte** de la logique sous-jacente, contribuant à l'**explicabilité** du système final. C'est cette double souplesse qui, dans la ligne de ce chapitre (section 1.5.7.2), semble prometteuse pour de nombreuses applications, allant de la **robotique cognitive** à l'**intégration** de vastes **bases de connaissances** et de **flux sensoriels**.

1.5.7.3. Mécanismes d'Auto-Organisation Intégrant la Logique

Dans la continuité de l'approche **hybride** symbolique–connexionniste présentée en section 1.5.7.2, il est envisageable d'introduire au sein d'un **Synergistic Connection Network (SCN)** des **règles symboliques** interagissant avec des **entités** sub-symboliques. On peut par exemple considérer une **règle** R stipulant :

Si la caméra détecte une forme circulaire, alors suspecter un panneau de signalisation.

Parallèlement, on introduit une **entité sub-symbolique** \mathcal{E}_{cam} chargée d'analyser des données visuelles. Cette entité exploite typiquement un **modèle** neuronique local pour reconnaître ou segmenter des formes dans un **flux** d'images.

La **pondération** $\omega_{R,\text{cam}}$ reliant la règle R à l'entité \mathcal{E}_{cam} suit la dynamique du **DSL**. On peut ainsi l'écrire :

$$\omega_{R,\text{cam}}(t+1) = \omega_{R,\text{cam}}(t) + \eta [S(R, \mathcal{E}_{\text{cam}}) - \tau \omega_{R,\text{cam}}(t)],$$

où

$S(R, \mathcal{E}_{\text{cam}})$ mesure la **synergie** entre la **règle symbolique** et la **détection sub-symbolique** opérée par \mathcal{E}_{cam} . Concrètement, si la **forme circulaire** repérée par la caméra coïncide fréquemment avec la proposition « panneau de signalisation », la quantité $S(R, \mathcal{E}_{\text{cam}})$ prend une **valeur positive**. Le terme $\eta [S - \tau \omega]$ reste alors **positif**, ce qui **renforce** la liaison $\omega_{R,\text{cam}}$. Progressivement, on voit s'**organiser** un **cluster** local $\{R, \mathcal{E}_{\text{cam}}\}$, soulignant la coopération entre la **règle symbolique** et la **détection** d'images.

Lorsque survient une **deuxième** entité, $\mathcal{E}_{\text{limit}}$, qui incarne un **concept** tel que « panneau de limitation de vitesse », la **coopération** peut encore se prolonger. Si un **module** \mathcal{E}_{ocr} découvre régulièrement le texte “30 km/h” à l'intérieur des formes circulaires détectées, la synergie $S(\mathcal{E}_{\text{ocr}}, \mathcal{E}_{\text{limit}})$ peut s'avérer **élevée**. La **pondération** $\omega_{\text{ocr},\text{limit}}$ croît alors, impliquant l'émergence d'un **sous-réseau** plus vaste :

$$\{R, \mathcal{E}_{\text{cam}}, \mathcal{E}_{\text{ocr}}, \mathcal{E}_{\text{limit}}\}.$$

À l'intérieur de ce **cluster**, les entités **symboliques** (par exemple R et $\mathcal{E}_{\text{limit}}$) formulent des règles logiques ou des connaissances à caractère explicite. Les **entités** \mathcal{E}_{cam} et \mathcal{E}_{ocr} se chargent au contraire d'un **traitement** sub-symbolique (déttection de formes, reconnaissance du texte “30”). La **dynamique** des pondérations, selon la règle linéaire ou d'autres variantes (voir section 1.4.5), veille à **consolider** cet ensemble si les corrélations s'avèrent régulières et profitables.

Dans cette **topologie**, le **raisonnement** reste en grande partie **distribué** et local : il n'y a pas une unique couche hiérarchique, mais une **auto-organisation** en **micro-réseaux** où règles et modules neuronaux s'**associent** dès que leur **synergie** atteint un **niveau** significatif. Les **avantages** de cette intégration résident dans la capacité à **exploiter** la puissance de l'apprentissage sub-symbolique pour reconnaître des patterns visuels complexes, tout en s'appuyant sur la **précision** et la **transparence** qu'offrent les entités symboliques. De surcroît, on peut ainsi relier de multiples règles, chacune s'appliquant à des **entités** sub-symboliques différentes, ce qui permet d'atteindre une **modularité** plus élaborée à l'intérieur du SCN.

Les **limites** et **inconvénients** tiennent essentiellement au besoin de concevoir des règles symboliques suffisamment précises pour bien refléter la réalité observée par les flux sub-symboliques. S'il existe un décalage important ou un **bruit** excessif dans la détection, la synergie peut rester faible et empêcher la formation d'un cluster stable. Un autre **défi** réside dans la gestion de la **scalabilité** : si un trop grand nombre de règles ou d'entités sub-symboliques sont présentes, leur mise en correspondance peut devenir coûteuse à calculer, ce qui impose des stratégies de **sparsification** ou de **pruning** (cf. section 1.4.5).

Dans l'ensemble, cette **intégration de la logique** dans le **DSL** illustre la possibilité de **fusionner** la robustesse neuronale ou statistique et la **structuration** symbolique. L'**auto-organisation** garantit que seules les entités **règle-module** qui partagent un gain effectif conservent un lien fort, aboutissant à une **cohérence** émergente entre les **concepts** logiques et les **vecteurs** de features dans le **Synergistic Connection Network**.

1.5.7.4. Avantages pour la Cognition et l'Explicabilité

Dans la continuité des sections précédentes, le **Deep Synergy Learning (DSL)** se présente comme un **pont** unifiant deux grandes approches du traitement de l'information. D'une part, la **dimension sub-symbolique** s'attache aux aspects de **perception** et de **features** extraits directement des données (images, sons, textes bruts), héritant de la puissance d'**apprentissage** de type connexionniste. D'autre part, la **dimension symbolique** articule des **concepts**, des **règles** et des **faits** logiques qui ne seraient pas forcément découverts par simple analyse statistique, par exemple : « un véhicule possède quatre roues, un moteur et est conçu pour se déplacer sur la route ».

Lorsqu'un **cluster** (au sens de la section 1.4.3) associe une **règle** formalisée à des **entités perceptives** ancrées dans des vecteurs sub-symboliques, on obtient une **forme d'explication** plus directe de la décision. En effet, plutôt que de faire référence à un poids numérique situé dans une couche profonde du réseau, on peut pointer vers le **signal x** (issu de la représentation sub-symbolique) et la **règle R** (exprimée symboliquement) pour justifier la conclusion. Cette **transparence** s'apparente à une forme d'**explicabilité** (souvent recherchée en IA), où l'on peut réellement indiquer quelles **connaissances** et quelles **observations** ont motivé l'activation finale.

Pour formaliser mathématiquement cette idée, on peut écrire que la **pondération** $\omega_{R,\text{data}}(t)$ reliant la règle symbolique \mathcal{E}_R à l'entité sub-symbolique $\mathcal{E}_{\text{data}}$ évolue selon :

$$\omega_{R,\text{data}}(t+1) = \omega_{R,\text{data}}(t) + \eta [S(\mathcal{E}_R, \mathcal{E}_{\text{data}}) - \tau \omega_{R,\text{data}}(t)],$$

où $S(\mathcal{E}_R, \mathcal{E}_{\text{data}})$ mesure l'**adéquation** entre la règle symbolique et les **caractéristiques** perçues dans les données. Une fois cette liaison suffisamment renforcée, on peut interpréter la **décision** ou la **catégorie** apprise en mentionnant explicitement la règle R et le segment de données x qui la **valide**.

Ce mécanisme illustre la notion de **cognition synergique** : les **entités** symboliques **codent** la **sémantique** et les **règles** explicites, tandis que les **entités** sub-symboliques **apprennent** à partir des données brutes, en extrayant des **patterns** ou des **features** utiles. Dans le **Synergistic Connection Network (SCN)**, la **dynamique** de mise à jour des **pondérations** relie directement ces deux pôles. On aboutit alors à une forme d'**intégration** qui se rapproche de la **cognition humaine**, où des **concepts** explicitement formulés cohabitent avec des **associations perceptives** acquises à travers l'expérience.

En termes d'**avantages**, on dispose d'une **robustesse** caractéristique des méthodes sub-symboliques, puisque le **réseau** peut s'adapter aux données, tout en bénéficiant de la **précision** et de la **lisibilité** qu'offre la **logique symbolique**. Cela se traduit par un **apprentissage** plus puissant combiné à une **explication** plus accessible : la conclusion ne dépend plus seulement d'une somme opaque de poids, mais peut être reliée à un **ensemble** (règle, concept, fait) qui rend compte de sa **signification**. Dans le même paragraphe, on peut souligner que la **cohabitation** de règles symboliques et de flux sub-symboliques exige une **cohérence** de haut niveau. Les règles doivent être appropriées pour décrire les phénomènes observés, et les entités sub-symboliques doivent parvenir à traduire efficacement le **signal** en indices qui **valident** ou **invalident** les **règles**.

Cette **fusion** entre ces deux mondes (symbolique et sub-symbolique) au sein du **DSL** peut ainsi instaurer un **climat** proche de ce qu'on appelle la "**cognition synergique**" : à la manière de l'esprit humain, le système mixe des **concepts** explicites et des **représentations** perceptives implicites, avant de les **coordonner** dynamiquement pour décider de l'interprétation finale.

1.5.7.5. Défis Mathématiques et Implémentation

Dans la poursuite des principes décrits en sections 1.5.7.2 à 1.5.7.4, se pose la question de la **représentation** mathématique des **règles** symboliques à l'intérieur du **Deep Synergy Learning (DSL)** et, plus généralement, les difficultés techniques qu'implique l'introduction de composantes **logiques**. Un premier enjeu consiste à décider comment **encoder** une règle $\mathcal{E}_{\text{rule}}$. Certains travaux préconisent d'associer à chaque règle un **vecteur** ou un **arbre** logique compact, ce qui permet d'attribuer à $\mathcal{E}_{\text{rule}}$ un **nœud** du réseau muni de **paramètres** θ_{rule} . On peut alors définir

la **synergie** $S(\mathcal{E}_{\text{rule}}, \mathcal{E}_{\text{feature}})$ de façon à intégrer à la fois la **structure symbolique** (par exemple, un parse-tree) et la **partie numérique** (par exemple, un embedding vectoriel ou un ensemble de features \mathbf{x}_{data}).

Il est possible d'écrire, de manière formelle, une fonction

$$S_{\text{mix}}(\mathcal{E}_{\text{rule}}, \mathcal{E}_{\text{data}}) = F(\text{logic}(\theta_{\text{rule}}), \text{embedding}(\mathbf{x}_{\text{data}})),$$

où logic exprime la partie symbolique (règles, connecteurs logiques, etc.), tandis que embedding encode la portion sub-symbolique. La fonction F peut évaluer une **co-information** ou un **score** d'adéquation entre la règle et les données. Cette démarche reprend les principes du **DSL** (sections 1.5.7.2 et 1.5.7.3) en donnant un **cadre** uniifié pour la synergie entre une entité symbolique et une entité sub-symbolique.

Au fur et à mesure que le **nombre de règles** logiques augmente, on observe que la taille du **réseau** s'accroît, car chaque $\mathcal{E}_{\text{rule}}$ ou $\mathcal{E}_{\text{concept}}$ vient s'ajouter aux nœuds. Les **calculs** de synergie $S_{\text{mix}}(\mathcal{E}_{\text{rule}}, \mathcal{E}_{\text{data}})$ peuvent devenir coûteux si l'on doit évaluer régulièrement de nombreuses règles sur de vastes flux sub-symboliques. Il est alors crucial de recourir à des **techniques de parsimonie**, en particulier celles décrites en section 1.4.5, qui permettent de **supprimer** ou de **couper** certaines liaisons $\omega_{i,j}$ dès lors que leur pondération demeure faible. Cette régulation limite la prolifération de liens inutiles, maintenant la **complexité** à un niveau gérable.

Lorsque plusieurs **règles** se **contredisent** au sein du **SCN**, des **tensions** apparaissent. Il est possible que l'une des entités sub-symboliques (par exemple un flux de capteurs) présente une **synergie** positive avec une première règle, mais se heurte à une **incompatibilité** logique avec la seconde. Dans un tel cas, on peut introduire un **score de cohérence** ou une **pénalité** $P(\mathcal{E}_{\text{rule}_1}, \mathcal{E}_{\text{rule}_2})$, reflétant le fait que les règles rule_1 et rule_2 ne peuvent valablement être activées en même temps. On peut alors modifier l'évolution de chaque pondération afin de **réduire** la force de liens conduisant trop souvent à cette contradiction, ou introduire un **terme** correctif dans la mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{mix}}(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)] - \alpha P(\mathcal{E}_i, \mathcal{E}_j),$$

où α contrôle l'intensité de la pénalité due à la contradiction.

D'un point de vue **implémentation**, les **limites** principales résident dans l'**explosion combinatoire** lorsque l'on souhaite intégrer non seulement des synergies binaires (section 1.4.4) mais également des synergies n-aires (section 1.4.7) et des **connaissances** logiques toujours plus nombreuses. De même, il peut être nécessaire de définir des **règles** plus modulaires ou paramétriques afin d'éviter la création d'une entité symbolique distincte pour chaque énoncé. Les entités pourraient alors coder des **schémas** logiques, associés à des **variables** substituables, pour diminuer le nombre total de nœuds.

Dans l'ensemble, ces **défis** mathématiques et de **mise en œuvre** ne remettent pas en cause le principe fondateur de l'approche hybride (sections 1.5.7.2 et 1.5.7.3), à savoir la possibilité de **fusionner** la rigueur symbolique avec la force d'apprentissage sub-symbolique. Ils incitent plutôt à développer des **algorithmes** de parsimonie adaptés, à concevoir des mécanismes de **cohérence** entre règles, et à prévoir des **heuristiques** ou **méthodes** de clustering pour gérer la croissance du réseau. Le **DSL**, dans ce contexte, se distingue par sa **capacité d'auto-organisation** : il veille à ce que seules les **liaisons** $\omega_{i,j}$ d'une **synergie** suffisante se stabilisent, et il répartit de façon **adaptative** les efforts de calcul selon les **bénéfices** perçus.

1.5.7.6. Conclusion

L'**incorporation** de composantes **symboliques** ou **cognitives** au sein du **Deep Synergy Learning (DSL)** ouvre un champ d'**IA hybride** dans lequel la **partie sub-symbolique** (traitement de données massives, bruitées et hétérogènes) est **complétée** par une **dimension symbolique** (règles, concepts, ontologies et raisonnement explicite). Cette combinaison se déploie naturellement dans le **Synergistic Connection Network (SCN)** qui, grâce à ses entités adaptatives et à ses liaisons synergiques évolutives, apparaît particulièrement adapté pour réunir et faire coévoluer ces deux approches.

On peut modéliser la **synergie** entre une **règle** R et un **module perceptif** $\mathcal{E}_{\text{data}}$ à l'aide d'une fonction

$$S(\mathcal{E}_{\text{rule}}, \mathcal{E}_{\text{data}}) = f(\theta_{\text{rule}}, \mathbf{x}_{\text{data}}),$$

où θ_{rule} encode l'information symbolique (logique, ontologie...) et \mathbf{x}_{data} représente les **features** sub-symboliques. La pondération $\omega_{\text{rule},\text{data}}(t)$ associée se met alors à jour de façon adaptative, selon

$$\omega_{\text{rule},\text{data}}(t+1) = \omega_{\text{rule},\text{data}}(t) + \eta [S(\mathcal{E}_{\text{rule}}, \mathcal{E}_{\text{data}}) - \tau \omega_{\text{rule},\text{data}}(t)].$$

Cette dynamique assure que seules les **connexions** réellement bénéfiques (au sens de la synergie mesurée) se consolident, et permet à des **clusters** associant règles symboliques et entités sub-symboliques d'émerger spontanément au sein du réseau.

Ce mécanisme procure de **multiples avantages**. Il **préserve** la capacité d'**apprentissage** sub-symbolique, cruciale pour dompter des données riches et bruitées, tout en **introduisant** des notions symboliques aisément **explicables** (règles, concepts, axiomes). Cette alliance favorise une **explication** plus claire des décisions, dans la mesure où la conclusion d'un cluster peut être reliée à la **règle R** (logique) et au **signal x** (apprentissage neuronal). De plus, le DSL conserve ses forces intrinsèques, telles que la **multimodalité**, l'**adaptation** continue face aux variations de distribution et la **résilience** aux données partielles ou contradictoires.

Il faut toutefois noter qu'en **élargissant** la base de **règles** ou de **concept**s, le réseau peut s'agrandir et accroître la **complexité** de calcul, comme évoqué en section 1.5.7.5. Des mécanismes de **parsimonie** ou de **régulation** (limitation des connexions inutiles, seuils dynamiques sur les liaisons) demeurent essentiels pour éviter une explosion combinatoire. On peut également ajouter des **pénalités** ou un **score de cohérence** afin de gérer des contradictions logiques entre plusieurs règles.

En définitive, on obtient ainsi un **DSL** apte à **raisonner** en s'appuyant sur sa **base symbolique**, tout en **apprenant** de manière continue via sa **dimension sub-symbolique**. Cette architecture apparaît comme un **paradigme plus complet**, se rapprochant de la **cognition humaine**, laquelle jongle entre des **concept**s explicites et des **associations** perceptives implicites. Les sections et principes exposés dans ce chapitre (1.5) soulignent la **flexibilité** du DSL et son aptitude à jeter des **ponts** entre la **puissance** d'apprentissage neuronale et la **lisibilité** de la connaissance symbolique.

Exercices Simples

Exercice 1 : Synergie vs. Architecture Figée

(Lien avec 1.5.1)

32. Rappelez ce qu'on entend par "architecture figée" dans un réseau neuronal profond (quel est son principe ?).
33. Expliquez en quoi une approche "synergique" (via des liens adaptatifs entre entités) peut rendre le réseau plus flexible sans modifier l'architecture d'ensemble.
34. Donnez un exemple concret où un réseau hiérarchique classique serait limité, alors qu'une approche DSL pourrait se reconfigurer spontanément.

Exercice 2 : Multi-modalité Naturelle

(Lien avec 1.5.2)

35. Qu'est-ce qu'on entend par "fusion tardive" et "fusion précoce" dans les approches multi-modales traditionnelles ?
36. Comparez ces méthodes à la "co-intégration libre" dans le DSL : en quoi la synergie facilite-t-elle la découverte de liens entre modalités ?
37. Proposez un cas (vision + audio + texte) où la multi-modalité est cruciale, et indiquez comment le DSL l'absorbe sans configurer manuellement une "couche de fusion".

Exercice 3 : Données Manquantes ou Bruitées

(Lien avec 1.5.3)

38. Dans un pipeline classique, citez au moins deux méthodes usuelles pour traiter les données incomplètes.
39. Expliquez la notion d'"auto-exclusion" d'une entité bruyante dans le DSL (comment se manifeste la chute de synergie ?).
40. Donnez un exemple imagé montrant comment un capteur défaillant peut être automatiquement mis à l'écart par l'auto-organisation synergique.

Exercice 4 : Adaptation Continue et Lifelong Learning

(Lien avec 1.5.4)

41. Définissez le "catastrophic forgetting" dans un réseau neuronal classique.
42. Montrez en quelques phrases pourquoi le DSL, avec ses liens ajustés en permanence, peut mieux gérer l'arrivée d'une nouvelle tâche ou d'un nouveau flux de données.
43. Donnez un exemple où l'on ajoute une entité $\mathcal{E}_{\text{nouvelle}}$ en milieu de parcours : comment se forment les connexions si la synergie est positive ?

Exercice 5 : Symbolique et Sub-symbolique

(Lien avec 1.5.7)

44. Qu'entend-on par "entité symbolique" vs. "entité sub-symbolique" dans un réseau ?

45. Donnez un exemple de règle symbolique qui pourrait être encodée dans une entité \mathcal{E}_{rule} .
46. Décrivez brièvement comment cette règle peut se lier à une entité perceptive \mathcal{E}_{vision} si un motif détecté correspond à la condition de la règle.

II. 5 Grands Problèmes Approfondis

Problème 1 : Avantages Structuraux et Comparaison aux Réseaux Neuronaux

(Couvre 1.5.1, avec références à 1.5.3 et 1.5.4 éventuellement)

Énoncé :

On veut analyser la différence entre une architecture figée de réseau profond et l'approche “auto-organisée” du DSL pour un scénario de classification complexe (images + quelques méta-information).

47. Question A :

Décrivez comment, dans un CNN classique, les couches sont conçues et pourquoi cela peut rendre l'apprentissage rigide quand on introduit de nouvelles classes d'images ou de nouvelles caractéristiques (méta-information).

48. Question B :

Montrez, en termes de synergie, comment un bloc $\mathcal{E}_{metainfo}$ peut s'auto-connecter à un bloc \mathcal{E}_{vision} si l'on détecte une forte complémentarité. Quelles conséquences sur la structure du réseau ?

49. Question C :

Comparez la souplesse d'évolution du DSL lorsqu'on ajoute des “features” ou lorsqu'une partie du CNN classique doit être reconçue. Quelle conclusion tirer pour des tâches en flux continu (continual learning) ?

Problème 2 : Multi-Modalité et Synergie dans un Système Réel

(Couvre 1.5.2, avec liens vers 1.5.6 : représentations riches)

Énoncé :

On se place dans un système de reconnaissance d'événements (audio + vidéo + texte). Décrivez comment les entités correspondantes se forment et coévoluent.

50. Question A :

Définissez les entités \mathcal{E}_{vid} , \mathcal{E}_{aud} , \mathcal{E}_{txt} . Proposez une forme pour leur représentation interne (ex. vecteurs embedding).

51. Question B :

Comment la synergie “audio–texte” peut-elle se mesurer ? Donnez une idée (distance, co-information, etc.). Qu'est-ce que cela implique pour la création de liens $\omega_{aud,txt}$?

52. Question C :

Supposez qu'un 4^e flux apparaisse (capteur de mouvement). Quelles étapes se produisent pour que \mathcal{E}_{mouv} s'insère ou non dans le cluster multimodal formé ?

53. Question D :

Montrez en quoi cette intégration multi-modale **améliore** la “richesse” de la représentation globale, et commentez l’aspect d'**interprétabilité** (section 1.5.6) : comment visualiser et expliquer un cluster audio–vidéo–texte–mouvement ?

Problème 3 : Gestion du Bruit, Adaptation Continue et Réduction de Supervision

(Couverte 1.5.3, 1.5.4, 1.5.5)

Énoncé :

On s’intéresse à un contexte industriel où plusieurs capteurs surveillent une chaîne de production. Certains capteurs sont souvent défaillants (bruit élevé), et on n’a que peu de labels pour repérer les “pannes” effectives. On veut voir pourquoi le DSL s’en sort mieux qu’un réseau neuronal classique.

1. Question A :

Décrivez comment l'**auto-exclusion** d’un capteur bruyant se produit dans le DSL : quel rôle joue la baisse de synergie $\omega_{i,j}$? Expliquez la dynamique mathématique (rappel de la mise à jour $\omega \leftarrow \omega + \eta[S - \tau\omega]$).

2. Question B :

Illustrez la notion de “peu de labels” : comment le DSL peut structurer les capteurs et détecter des clusters anormaux (pannes potentielles) **sans** un ensemble massivement labellisé ?

3. Question C :

Montrez qu’en cas de changement de distribution (p. ex. nouvelle configuration de la chaîne), la **mise à jour continue** favorise la réorganisation progressive, alors qu’un réseau classique exigerait un réapprentissage partiel ou total (risque de catastrophic forgetting).

4. Question D :

Concluez en expliquant pourquoi cette approche diminue la **dépendance** à un jeu de labels exhaustif (mieux qu’un CNN supervisé standard).

Problème 4 : Représentations Interprétables et Conception d’un DSL “lisble”

(Couverte 1.5.6 : *représentations riches et interprétables, avec aspects multi-modal / clusters*)

Énoncé :

On veut mettre en place un DSL pour une application de recommandation de contenus (médias, articles, etc.). L'**interprétabilité** est cruciale : on souhaite comprendre pourquoi le réseau propose tel contenu à tel utilisateur.

1. Question A :

Proposez un modèle d’entités $\{\mathcal{E}_{\text{user}}\}$ et $\{\mathcal{E}_{\text{content}}\}$. Quelles informations contiennent-elles (profils, features, historique) ?

2. Question B :

Expliquez la formation spontanée de **clusters** : des groupes d’utilisateurs + contenus qui partagent une synergie. Comment lire ces clusters pour en extraire des explications (ex. “ce cluster aime la musique rock, ce contenu est du rock, etc.”) ?

3. Question C :

Peut-on imaginer un macro-cluster qui regroupe plusieurs sous-ensembles (genres musicaux, types de public) ? Montrez comment la synergie de second ordre (ou plus) peut révéler des regroupements plus larges.

4. **Question D :**

Mettez en avant la différence avec un système de recommandation standard (collaborative filtering) : en quoi la “coopération adaptative” du DSL facilite-t-elle l’explicabilité ? Donnez un exemple narratif.

Problème 5 : Intégration de Symbolique et de Sub-symbolique dans le DSL

(Couverte 1.5.7 : dimensions symboliques ou cognitives)

Énoncé : On cherche à intégrer un **module de règles logiques** dans un DSL qui, par ailleurs, gère des flux sub-symboliques (vision, audio). L’objectif : détecter un événement “danger” si plusieurs conditions symboliques et perceptives sont réunies.

1. **Question A :**

Donnez un **exemple** concret d’une règle symbolique \mathcal{E}_{rule} : “Si la température > 100°C ET la pression > 5 bars, ALORS suspecter surchauffe.”

2. **Question B :**

Expliquez comment cette règle peut être une entité \mathcal{E}_{rule} dans le DSL, et quelle forme prend la synergie $\omega_{rule,capteurTemp}, \omega_{rule,capteurPress}$.

3. **Question C :**

Décrivez la **formation** d’un cluster $\{\mathcal{E}_{rule}, \mathcal{E}_{capteurTemp}, \mathcal{E}_{capteurPress}\}$. Qu’adviennent-il si un 3^e paramètre (ex. “vibration”) s’ajoute ? Sous quelles conditions la règle évolue ou se connecte à cette nouvelle entité ?

4. **Question D :**

Quels **défis** (scalabilité, cohérence logique, etc.) se posent si on introduit plusieurs dizaines de règles symboliques dans le même DSL ? Comment gérer la contradiction entre deux règles ?

Conclusion

Ces **5 exercices** et **5 grands problèmes** couvrent l’essentiel des **motifs** pour lesquels une approche **synergique** (DSL) se révèle attrayante :

- **Comparaison** avec les réseaux neuronaux profonds (1.5.1),
- **Gestion** fluide de la **multi-modalité** (1.5.2),
- **Souplesse** face aux données bruitées ou incomplètes (1.5.3),
- **Adaptation** continue ou auto-évolution (1.5.4),
- **Réduction** de la supervision (1.5.5),
- **Représentations** plus riches et interprétables (1.5.6),
- **Dimensions** symboliques ou cognitives (1.5.7).

Les exercices offrent une **entrée rapide**, tandis que les problèmes approfondissent chacun plusieurs aspects, mettant en avant la **dynamique** du DSL et sa **philosophie** de “co-apprentissage” par la **synergie**.

Exercices et problèmes

I. 5 Exercices (courts, purement mathématiques)

Exercice 1 : Mesure de Synergie et Point Fixe

(Référence à 1.5.1 : Avantages face aux réseaux profonds, via une pondération adaptative)

Soit une entité \mathcal{E}_i et une entité \mathcal{E}_j , dont la pondération synergique suit la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{i,j} - \tau \omega_{i,j}(t)],$$

avec $\eta, \tau > 0$ et $S_{i,j} \in \mathbb{R}$ une constante (mesure de synergie binaire).

1. **Montrez** qu'il existe un point fixe $\omega_{i,j}^*$ pour cette dynamique et **calculez** $\omega_{i,j}^*$.
2. **Discutez** la condition de stabilité (linéarisation autour de $\omega_{i,j}^*$) menant à $\eta \tau < 1$.
3. **Concluez** brièvement pourquoi cet ajustement local favorise la flexibilité structurelle, en comparaison d'un poids figé dans un réseau profond.

Exercice 2 : Synergie Multi-modal par Distance

(Référence à 1.5.2 : multi-modalité)

On définit deux entités \mathcal{E}_A (domaine “visuel”) et \mathcal{E}_B (domaine “audio”), chacune représentée par un vecteur dans \mathbb{R}^d : $\mathbf{x}_A, \mathbf{x}_B$. On pose :

$$d(\mathbf{x}_A, \mathbf{x}_B) = \|\mathbf{x}_A - \mathbf{x}_B\|, \quad S(\mathcal{E}_A, \mathcal{E}_B) = \frac{1}{1 + \|\mathbf{x}_A - \mathbf{x}_B\|^2}.$$

1. **Prouvez** que $S(\mathcal{E}_A, \mathcal{E}_B)$ est inversément proportionnelle à la distance euclidienne au carré.
2. **Montrez** que $S(\cdot, \cdot)$ atteint son maximum quand $\mathbf{x}_A = \mathbf{x}_B$.
3. **Interprétez** mathématiquement la situation où \mathbf{x}_A et \mathbf{x}_B proviennent de modalités différentes : pourquoi $S(\cdot, \cdot)$ peut rester élevé même si $\mathbf{x}_A \neq \mathbf{x}_B$ au sens brut, mais “proches” dans un espace latent ?

Exercice 3 : Complétion Partielle via Synergie

(Référence à 1.5.3 : flexibilité face aux données incomplètes)

Soient deux vecteurs $\mathbf{u}, \mathbf{v} \in \mathbb{R}^d$, chacun avec des composantes manquantes (codées par NaN ou *). On définit un opérateur \oplus qui ignore les composantes manquantes et évalue la distance sur les seules dimensions communes. Par exemple :

$$d_{\oplus}(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{k \in I} (u_k - v_k)^2},$$

où $I \subset \{1, \dots, d\}$ est l'ensemble des composantes non manquantes à la fois chez \mathbf{u} et \mathbf{v} .

1. **Formulez** une mesure de synergie $S(\mathbf{u}, \mathbf{v})$ à partir de d_{\oplus} .
2. **Démontrez** que si \mathbf{u} et \mathbf{v} partagent un nombre restreint de dimensions, la variance sur ces dimensions peut suffire à créer un lien fort si la similarité est forte dessus.
3. **Commentez** la robustesse face au bruit : si \mathbf{u} est bruyante sur quelques composantes, pourquoi la définition par \oplus peut limiter l'influence de ces composantes défaillantes ?

Exercice 4 : Dynamique Continue et Auto-Évolution

(Référence à 1.5.4 : adaptation continue)

On considère la version “continue” de l'équation de mise à jour :

$$\frac{d\omega_{i,j}}{dt} = \eta[S_{i,j} - \tau\omega_{i,j}(t)].$$

1. **Résolvez** l'EDO pour $\omega_{i,j}(t)$ en supposant $\omega_{i,j}(0) = \omega_0$. Trouvez la forme explicite de la solution $\omega_{i,j}(t)$.
2. **Déterminez** la valeur limite lorsque $t \rightarrow \infty$.
3. **Interprétez** l'intérêt de cette dynamique dans un cadre “lifelong learning”, où $\omega_{i,j}(t)$ peut s'ajuster en temps réel sans repasser par un entraînement batch global.

Exercice 5 : Synergie Symbolique et Sub-symbolique

(Référence à 1.5.7 : dimensions symboliques)

On suppose deux types d'entités :

- $\mathcal{E}_{\text{rule}}$ (règle symbolique) décrite par un petit vecteur $\mathbf{r} \in \{0,1\}^m$ (chaque composante représentant la présence d'une condition),
- $\mathcal{E}_{\text{feat}}$ (feature sub-symbolique) décrite par un vecteur réel $\mathbf{x} \in \mathbb{R}^m$.

Définissez une fonction de synergie :

1. **Proposez** une mesure $S(\mathcal{E}_{\text{rule}}, \mathcal{E}_{\text{feat}})$ qui compare \mathbf{r} et \mathbf{x} (par ex. un produit mixte).
2. **Montrez** qu'on peut extraire une valeur de “compatibilité” logique, si $\mathbf{r}_k = 1$ implique $\mathbf{x}_k > \theta$.
3. **Concluez** sur l'idée d'**hyper-arêtes** si l'on intègre plusieurs règles ou features simultanément.

II. 5 Grands Problèmes (avec plusieurs questions mathématiques)

Problème 1 : Analyse Formelle des Avantages face aux Réseaux Profonds

(Couvre 1.5.1, éléments de 1.5.3 et 1.5.4)

Énoncé mathématique :

On modélise un réseau neuronal “classique” (à architecture figée) comme un ensemble de matrices de poids $\{W^{(l)}\}_{l=1}^L$. Pour incorporer un nouveau module ou une nouvelle tâche, il faut ajouter des poids ΔW et réentraîner partiellement. Dans le **DSL**, on a une matrice $\Omega(t)$ de taille $n \times n$, évoluant selon :

$$\Omega(t+1) = \Omega(t) + \eta [S(\Omega(t)) - \tau \Omega(t)],$$

pour une fonction “globale” $S(\cdot)$.

1. **Question A** (Comparaison paramétrique) :

Évaluez la **dimension paramétrique** d'un CNN figé (où seule la rétropropagation ajuste $\{W^{(l)}\}$) contre un DSL pouvant **augmenter ou réduire** la taille de $\Omega(t)$ (nouvelles entités, etc.). Quelle conclusion sur la capacité d'évolution ?

2. **Question B** (Stabilité d'un point fixe global) :

Supposons qu'on définisse une fonction énergie $J(\Omega) = -\sum_{i,j} \Omega_{i,j} S_{i,j} + \text{termes de régularisation}$. Montrez que la descente locale en J entraîne la **création** ou la **dissolution** de liens. Comment cela surpasse la rigidité d'un CNN ?

3. **Question C** (Scénario d'ajout d'entités) :

Mathématisez la situation où on ajoute Δn entités, agrandissant Ω . Définissez les conditions sous lesquelles ces nouvelles entités se connectent rapidement à des clusters existants.

Problème 2 : Multi-modalité et Mesure de Synergie Non Linéaire

(Couverte 1.5.2, aspects 1.5.6)

Énoncé mathématique :

Soit un jeu de deux modalités $\mathbf{x}_{\text{vis}} \in \mathbb{R}^d$ (image) et $\mathbf{x}_{\text{aud}} \in \mathbb{R}^p$ (spectre audio). On veut définir une **co-information** n-aire :

$$I(\mathbf{x}_{\text{vis}}, \mathbf{x}_{\text{aud}}) = H(\mathbf{x}_{\text{vis}}) + H(\mathbf{x}_{\text{aud}}) - H(\mathbf{x}_{\text{vis}}, \mathbf{x}_{\text{aud}}),$$

puis en déduire une synergie.

1. **Question A** (Discrétisation) :

Montrez que si on approxime \mathbf{x}_{vis} et \mathbf{x}_{aud} par des variables discrètes (binning, par ex.), la **complexité** de l'estimation d'entropie croît exponentiellement avec la dimension.

2. **Question B** (Définition de la synergie) :

Proposez une fonction $S(\mathbf{x}_{\text{vis}}, \mathbf{x}_{\text{aud}}) = \max(0, I(\mathbf{x}_{\text{vis}}, \mathbf{x}_{\text{aud}}))$. Montrez que cette synergie devient plus grande si les deux modalités sont statistiquement dépendantes.

3. **Question C** (Clusterisation auto-organisée) :

Supposez que $\mathbf{x}_{\text{vis}}, \mathbf{x}_{\text{aud}}, \mathbf{x}_{\text{txt}}$ forment un trio. Comment formuler la synergie n-aire

$$S_3(\mathbf{x}_{\text{vis}}, \mathbf{x}_{\text{aud}}, \mathbf{x}_{\text{txt}})$$

en termes d'information mutuelle conjointe ? Quelles difficultés (estimation, calcul) souligner ?

Problème 3 : Données Incomplètes, Bruit et Équations de Mise à Jour

(Couverte 1.5.3, 1.5.4, 1.5.5)

Énoncé mathématique :

Soit un ensemble de vecteurs $\{\mathbf{x}_i\}_{i=1}^n$ dans \mathbb{R}^d , dont certains composants sont manquants ou bruités. On définit une **matrice de similarités** $[S_{i,j}]$ adaptée : si \mathbf{x}_i ou \mathbf{x}_j sont incomplètes, on calcule la similarité sur les dimensions communes. On met à jour $\omega_{i,j}(t)$ par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{i,j} - \tau \omega_{i,j}(t)].$$

1. **Question A** (Blocage par bruit) :

Définissez un seuil ω_{\min} . Montrez que si \mathbf{x}_k est très bruyante, son $\omega_{k,j}(t)$ pour tout j reste durablement $< \omega_{\min}$. En déduire l'**isolement** de \mathcal{E}_k .

2. **Question B** (Renforcement partiel) :

Montrez qu'une entité \mathcal{E}_m partiellement incomplète peut quand même obtenir un $\omega_{m,j}$ élevé si sur les dimensions disponibles (communes), la similarité est grande.

3. **Question C** (Inclusion dans un cluster) :

Formalisez la condition d'entrée de \mathcal{E}_m dans un cluster $\mathcal{C} \subset \{1, \dots, n\}$: la somme interne $\sum_{j \in \mathcal{C}} \omega_{m,j}$ doit dépasser un seuil. Discutez de l'aspect "faible supervision".

Problème 4 : Représentations Riches, Clusters et Macro-Clusters

(*Couvre 1.5.6, aspects 1.5.2*)

Énoncé mathématique :

On suppose qu'on a un ensemble d'entités $\{\mathcal{E}_i\}_{i=1}^n$, chacune pouvant posséder sa représentation $\mathbf{x}_i \in \mathbb{R}^d$. La synergie binaire est

$$S(\mathcal{E}_i, \mathcal{E}_j) = f(\mathbf{x}_i, \mathbf{x}_j),$$

pour une fonction f quelconque (distance, similarité, co-information). On définit un **cluster** \mathcal{C} comme un sous-ensemble maximisant

$$\Sigma(\mathcal{C}) = \sum_{i,j \in \mathcal{C}} \omega_{i,j}.$$

1. **Question A** (Macro-cluster) :

Introduisez une fonction $\Omega(\mathcal{C}_1, \mathcal{C}_2) = \frac{1}{|\mathcal{C}_1||\mathcal{C}_2|} \sum_{i \in \mathcal{C}_1, j \in \mathcal{C}_2} \omega_{i,j}$. Montrez en termes algébriques comment la "fusion" $\mathcal{C}_1 \cup \mathcal{C}_2$ se produit si $\Omega(\mathcal{C}_1, \mathcal{C}_2)$ dépasse un certain seuil.

2. **Question B** (Explicabilité via sous-graphes) :

Supposez qu'on regarde la matrice Ω restreinte à un cluster \mathcal{C} . Commentez la diagonale bloquée $\{\omega_{i,j}\}_{i,j \in \mathcal{C}}$. Montrez qu'une structure en "sous-blocs" peut indiquer des **macro-clusters** successifs.

3. **Question C** (Richesse des représentations) :

Expliquez pourquoi des entités diverses (ex. vecteurs image, vecteurs audio, entités symboliques) induisent des clusters hétérogènes, et discutez l'avantage d'avoir plusieurs macro-clusters plutôt qu'un unique super-cluster.

Problème 5 : Entités Symboliques, Synergie Logique et Sub-symbolique

(Couverte 1.5.7)

Énoncé mathématique :

On introduit un ensemble d'entités logiques $\{\mathcal{E}_\alpha^{(\text{symb})}\}_{\alpha=1}^r$ et un ensemble d'entités sub-symboliques $\{\mathcal{E}_\beta^{(\text{feat})}\}_{\beta=1}^s$. Pour toute règle symbolique \mathcal{R}_α , on l'encode comme un vecteur binaire $\mathbf{r}_\alpha \in \{0,1\}^d$. Les features $\mathbf{x}_\beta \in \mathbb{R}^d$. On définit :

$$S(\mathcal{R}_\alpha, \mathcal{E}_\beta^{(\text{feat})}) = \sum_{k=1}^d [\mathbf{r}_\alpha[k] \cdot \phi(\mathbf{x}_\beta[k])],$$

Où $\phi(\cdot)$ est une fonction d'activation assurant un gain si la condition binaire est satisfaite.

1. **Question A** (Formulation du produit logique) :

Montrez qu'en prenant $\phi(u) = \mathbf{1}_{\{u>\theta\}}$ (indicateur), la synergie est un compte du nombre de dimensions k où $\mathbf{r}_\alpha[k] = 1$ ET $\mathbf{x}_\beta[k] > \theta$.

2. **Question B** (Évolution des liaisons) :

Écrivez l'équation

$$\omega_{\alpha,\beta}(t+1) = \omega_{\alpha,\beta}(t) + \eta [S(\mathcal{R}_\alpha, \mathcal{E}_\beta^{(\text{feat})}) - \tau \omega_{\alpha,\beta}(t)].$$

Montrez que si la règle symbolique \mathcal{R}_α s'applique souvent aux features \mathbf{x}_β , la pondération $\omega_{\alpha,\beta}$ se consolide.

3. **Question C** (Clusters mixtes) :

Définissez la notion d'un **cluster** contenant à la fois des règles logiques et des entités de features. Montrez comment la somme interne $\sum_{\alpha,\beta \in \mathcal{C}} \omega_{\alpha,\beta}$ peut révéler un module cognitivement lisible (logique + perception).

4. **Question D** (Limite d'échelle) :

Discutez la difficulté si le nombre total de règles \mathbf{r}_α et d'entités \mathbf{x}_β est très grand. Quels compromis doit-on envisager pour réduire la complexité ?

1.6. Applications Pressenties et Domaines Impactés

Le **Deep Synergy Learning (DSL)**, en tant que paradigme alliant **auto-organisation**, **synergies adaptatives** et **capacité d'évolution continue**, suscite un **intérêt grandissant** dans de nombreux champs de l'intelligence artificielle. Après avoir dressé, dans les sections précédentes, les fondements conceptuels et structurels du DSL (sections 1.1 à 1.5), il est temps de se projeter vers les **applications** et les **domaines** où cette approche pourrait exercer un **impact fort**.

L'architecture **distribuée** et **plastique** du DSL, la possibilité d'**auto-organiser** des entités multimodales ou symboliques, ainsi que sa **tolérance** aux données bruitées, confèrent au DSL un potentiel d'**adaptation** et de **complémentarité** qui dépasse souvent les limites des modèles traditionnels (réseaux neuronaux profonds strictement hiérarchiques ou méthodes symboliques pures). De la **vision artificielle** à la **robotique**, en passant par la **reconnaissance audio**, la **recommandation personnalisée**, la **surveillance** ou encore la **planification industrielle**, le DSL offre des réponses nouvelles, mieux adaptées à la complexité des flux de données et aux exigences d'évolution en temps réel.

Dans cette section (1.6), nous présentons un **éventail** des applications pressenties, non seulement comme une liste d'exemples, mais aussi comme une **cartographie** des domaines que le DSL peut transformer :

5. **Vision Artificielle et Reconnaissance d'Objets Complexes** (1.6.1)
6. **Analyse Audio et Traitement du Langage Naturel** (1.6.2)
7. **Robotique et Systèmes Intelligents Évolutifs** (1.6.3)
8. **Recommandation Personnalisée et Systèmes de Décision** (1.6.4)
9. **Surveillance, Diagnostic Médical et Anomalies** (1.6.5)
10. **Planification et Optimisation dans l'Industrie 4.0** (1.6.6)
11. **Perspectives pour la Recherche Fondamentale en IA Forte** (1.6.7)

Pour chacun de ces thèmes, le DSL se révèle prometteur, en raison de ses **mécanismes** d'auto-organisation : **clusters dynamiques**, **pondérations adaptatives**, **intégration** de multiples sources (visuelles, auditives, textuelles, capteurs...), et **apprentissage** continu sans dépendance exclusive à la supervision. Nous verrons ainsi comment le DSL peut contribuer à relever certains **défis** actuels de l'IA, tels que la **plasticité**, la **résilience** aux données incertaines, ou encore la **capacité** à modéliser la **complexité** grandissante des problèmes.

1.6.1. Vision Artificielle et Reconnaissance d'Objets Complexes

Parmi les champs qui ont le plus bénéficié de l'essor des réseaux neuronaux profonds, on trouve la **vision artificielle**. Les **CNN** (Convolutional Neural Networks) ont acquis une renommée considérable pour la détection et la classification d'objets dans des images. Toutefois, les tâches de **reconnaissance** deviennent de plus en plus **subtiles** : il ne s'agit plus seulement d'identifier un objet (ex. un chat, un chien), mais de **déetecter** et de **comprendre** des scènes **complexes**, dans des environnements qui évoluent, avec des conditions lumineuses variables, des objets partiellement masqués ou des articulations multiples d'objets. Le **Deep Synergy Learning** ouvre ici de nouvelles perspectives, dont nous détaillons ci-après les ressorts principaux.

1.6.1.1. Au-delà d'une Hiérarchie Rigue : Clusters d'Entités Visuelles

Dans un **CNN** classique, les couches de convolution et de pooling s'enchaînent selon un **pipeline** prédéfini : la première couche repère des motifs élémentaires (bords, contrastes), la seconde assemble ces bords pour former des traits plus élaborés, et ainsi de suite, jusqu'à la classification. Bien que très performant, ce mécanisme demeure relativement **rigide**. Il impose un ordre précis selon lequel les "features" visuelles doivent être détectées, sans autoriser la création ou la suppression dynamique de filtres ni de liaisons entre diverses régions de l'image.

Le Deep Synergy Learning (DSL) envisage les **features** ou régions d'images comme autant d'**entités** \mathcal{E}_i , chacune portant un **descripteur** lié à un patch, un certain motif ou un vecteur plus abstrait (par exemple issu d'un autoencodeur local). Ces entités visuelles s'**auto-organisent** en **clusters** dès qu'elles identifient une **synergie** notable (proximité de descripteurs, co-occurrences de motifs, etc.). L'**hiérarchie** (ou la “stratification”) n'est plus imposée par des couches fixes, mais émerge au contraire de la **coopération** entre entités qui jugent profitable leur rapprochement. On peut par exemple voir se constituer un cluster dédié aux formes circulaires, un autre regroupant les zones de fortes lignes horizontales, et d'autres encore, voire des **macro-clusters** combinant plusieurs de ces groupes si leur fusion devient pertinente.

Cette **dynamique** apporte une plus grande **flexibilité** dans la reconnaissance d'objets complexes ou inattendus. Si un objet n'appartient pas aux catégories standards (par exemple, un objet partiellement masqué ou inconnu), les entités \mathcal{E}_k décrivant ses parties peuvent spontanément **coopérer** avec celles décrivant d'autres objets partiellement similaires. Il se crée alors un **sous-réseau** (cluster) susceptible de fusionner avec un regroupement plus vaste, agrandissant le champ d'interprétation.

Pour un modèle plus formel, on peut représenter chaque entité visuelle \mathcal{E}_k par un vecteur $\mathbf{x}_k \in \mathbb{R}^d$. La synergie, qu'elle soit définie via une **distance** (ex. inversée) ou une **co-information**, détermine la valeur $S(\mathcal{E}_k, \mathcal{E}_m)$. Les **pondérations** $\omega_{k,m}$ s'ajustent de façon adaptative au fil des données. Plusieurs entités corrélées finissent par **stabiliser** leurs liaisons internes, formant un **cluster** autour d'un même objet, d'un motif, ou d'une scène particulière. Cette organisation n'est pas définitive : si le réseau découvre que d'autres entités apportent une information complémentaire, les **liaisons** s'amplifient, élargissant ou refaçonnant le **sous-graphe** pour intégrer de nouvelles découvertes visuelles. En ce sens, la reconnaissance s'établit de manière plus **organique** qu'avec un pipeline hiérarchique statique, autorisant une diversité de regroupements et le partage de sous-patrons entre plusieurs objets si c'est jugé profitable.

1.6.1.2. Multi-Échelle et Robustesse aux Déformations

Dans le traitement d'objets complexes, il est fréquent de rencontrer des **variations** d'échelle (petit/grand dans l'image), des **transformations** géométriques (rotation, vue partielle, ombre) ou une **composition** de sous-parties distinctes. Les réseaux de neurones de type **CNN** gèrent généralement ces phénomènes en pratiquant la **convolution multi-échelle** ou la **data augmentation**, où l'on applique diverses transformations (zoom, rotation...) lors de l'entraînement. Cette stratégie, efficace dans bien des cas, reste toutefois largement paramétrée à l'avance (couches de pooling fixes, patterns choisis pour l'augmentation).

Le Deep Synergy Learning (DSL) propose un mécanisme différent. Des **entités** \mathcal{E}_i peuvent décrire un même objet à des échelles ou des angles divers, et leurs **liaisons** $\omega_{i,j}$ se renforcent si la **synergie** (similarité, co-information) confirme qu'il s'agit bien de la **même structure** vue sous différentes perspectives. L'existence d'une entité dédiée à la “version miniaturisée” de l'objet et une autre à la “version grand format” aboutit à un **cluster** commun si elles trouvent un **gain** à leur rapprochement, sans exiger qu'on impose explicitement un niveau de pooling ou un tableau de transformations prédéterminé.

Un premier **exemple** (lié à l'invariance d'échelle) peut être formalisé en introduisant, pour un patch visuel \mathbf{x}_k , une entité $\mathcal{E}_{k,\alpha}$ décrivant ce patch à l'échelle α . Les pondérations $\omega_{(k,\alpha),(k,\beta)}$ s'élèvent si la ressemblance (ou co-information) entre la représentation à l'échelle α et à l'échelle β demeure élevée, ce qui crée un **cluster** multi-échelle au sein duquel ces entités se regroupent.

Un second **exemple** (visant les déformations) consiste à considérer $\mathcal{E}_{k,\theta}$ comme la **version** du patch \mathbf{x}_k après avoir subi une **rotation** θ . Deux angles θ_1 et θ_2 peuvent alors mener à des entités jugées “similaires”, renforçant la liaison $\omega_{(k,\theta_1),(k,\theta_2)}$. On peut ainsi constituer un **macro-cluster** stable, reliant divers points de vue d'un même objet, garantissant une forme d'**invariance** à la rotation.

Cette approche confère une **robustesse** aux changements d'échelle et de pose. Le réseau n'a pas besoin de couches de pooling ni de transformations explicites ajoutées artificiellement : le **DSL** laisse les entités correspondantes **se découvrir** et **s'associer** si elles identifient un gain de synergie. L'ajustement se fait localement et **auto-organise** la reconnaissance, en évitant un pipeline rigide, pour composer des structures plus **adaptatives** et résilientes.

1.6.1.3. Interaction avec d'Autres Modalités (Vision + ...)

Dans un cadre multimodal (voir section 1.5.2), la **vision** artificielle s'avère fréquemment complémentée par un **flux audio** ou **textuel**. Le **Deep Synergy Learning (DSL)** n'établit pas de cloison pour la coopération entre l'entité visuelle \mathcal{E}_{vis} et l'entité auditive \mathcal{E}_{aud} . Le fait qu'elles représentent deux modalités différentes ne constitue pas une barrière : si la **synergie** $S(\mathcal{E}_{\text{vis}}, \mathcal{E}_{\text{aud}})$ se révèle positive (corrélation ou co-information notable), les **pondérations** $\omega_{\text{vis}, \text{aud}}$ se consolident, jusqu'à former un cluster dédié.

Dans la **reconnaissance** d'objets complexes, un **événement** visuel (comme un objet qui tombe) peut se coupler à un **événement** sonore simultané (bruit d'impact). Si la représentation \mathbf{x}_{vis} associée à la chute et la représentation \mathbf{x}_{aud} de l'onde sonore se rapprochent au sens de la **coopération**, un **sous-réseau** $\{\mathcal{E}_{\text{patch}}, \mathcal{E}_{\text{featureAudio}}\}$ se constitue. Il se spécialise dans la détection de la scène “objet tombant plus choc acoustique”, ce qui accroît les facultés de **classification** ou de compréhension de la situation (“Cassure”, “Chute de vaisselle”, etc.).

Cette **flexibilité** se généralise à d'autres flux, comme l'analyse **textuelle** (par exemple des légendes associées, des mots clés émis par un utilisateur), aboutissant à des micro-clusters multi-entités : la vision se combine à l'audio, et tous deux interagissent avec des entités textuelles. La dynamique d'**auto-organisation** du **DSL** détermine les **groupements** les plus profitables, sans exiger qu'un concepteur définisse un module de fusion dédié pour le couple “vision + audio”. Cela favorise une **intégration** sans couture des diverses modalités, conduisant à une représentation plus riche et plus **résiliente** aux variations d'une modalité prise isolément.

1.6.1.4. Impact sur les Problèmes de Vision Avancés

Dans le cadre de la **vision par ordinateur**, un **Deep Synergy Learning (DSL)** structuré en entités visuelles présente un atout certain pour traiter des scénarios avancés qui outrepasse la simple classification d'images. L'**auto-organisation** autorise l'émergence de **clusters** plus flexibles, allant au-delà du pipeline statique imposé par les méthodes classiques.

Dans la **détection d'objets multiples et partiellement superposés**, par exemple, un réseau traditionnel prévoit souvent un ensemble rigide d'ancrages (bounding boxes) ou un module de segmentation spécifique. Au contraire, le **DSL** peut laisser chaque patch ou région s'**affilier** à des **entités** “chien” et “chaise” même si ces objets se chevauchent dans l'image. Les pondérations $\omega_{i,j}$ augmentent dès que la synergie entre patchs associant l'idée de “chien” et l'idée de “chaise” se justifie. Le réseau n'a pas besoin d'un pipeline imposant a priori la détection séparée de chaque catégorie ; la constitution du **cluster** facilite l'identification conjointe de plusieurs objets.

Dans la **segmentation contextuelle**, on peut envisager que certaines entités segmentent l'image en superpixels et d'autres identifient des **contours**. Si leur **synergie** révèle une cohérence — par exemple, les bordures détectées coïncident avec les frontières d'un superpixel —, la liaison ω se renforce. On forme alors un **cluster** correspondant à une “zone cohérente” dans l'image, laquelle pourrait se rattacher à un objet complet ou à une portion d'arrière-plan homogène. Cette approche s'écarte de la segmentation par apprentissage end-to-end, en permettant une coopération plus libre entre entités spécialisées (détection de bordures, regroupement de pixels...) selon la présence d'un **gain**.

Dans un contexte de **vision évolutive en robotique**, un agent dans un environnement changeant (luminosité, décor, nouveaux éléments) doit s'ajuster en continu. Un réseau de neurones traditionnel exigerait soit un réentraînement, soit une réinitialisation partielle. Le **DSL**, quant à lui, se contente de réévaluer localement la **synergie** entre entités visuelles : les connexions correspondant à d'anciennes conditions (ex. forte luminosité) se voient **décroître** si elles ne s'avèrent plus utiles, et de nouvelles entités ou liaisons peuvent apparaître pour gérer des éclairages inédits, des angles de vue différents ou de nouveaux objets. Cette **adaptation** locale maintient la **structure** globale du réseau, sans nécessiter un apprentissage complet de bout en bout, et confère une plasticité plus grande à la **vision** robotique, tout en conservant la possibilité d'exploiter ses acquis antérieurs.

1.6.1.5. Conclusion Partielle : Une Alternative Complémentaire au CNN

Les **CNN** constituent toujours un **vecteur** extrêmement puissant pour la **vision** artificielle, en particulier lorsqu'un large corpus d'exemples annotés est disponible et que l'on vise des tâches de classification ou de détection sur des données stables. Toutefois, la structure même d'un **CNN** s'avère déterministe : chaque couche est fixée, et l'apprentissage se focalise principalement sur une logique de reconnaissance supervisée. Le **Deep Synergy Learning (DSL)** apporte une approche plus **organique** et **dynamique**, où les **entités** associées à des *patches* ou des *features* visuelles peuvent librement se **synergiser**, se **séparer**, s'**associer** avec d'autres modalités ou s'adapter aux **conditions** changeantes (nouvelles échelles, bruit, objets inédits).

La **structure du Synergistic Connection Network** autorise ainsi une **auto-organisation** continue, évitant la rigidité d'un pipeline imposé pour la vision. On peut introduire de nouvelles entités $\mathcal{E}_{\text{nouvelle}}$ si un objet inconnu apparaît ou si l'on veut traiter un autre type de patch visuel, ce qui met à jour localement les connexions ω . Rien n'empêche de **s'hybrider** avec un CNN : les **descripteurs** issus d'une couche intermédiaire d'un CNN peuvent constituer des entités \mathcal{E}_{CNN} dans le DSL, et la **synergie** entre ces entités reflète leur complémentarité.

Pour la **reconnaissance d'objets complexes** dans des scènes variées (éclairage fluctuant, angles de vue inédits, transformations géométriques), le **DSL** fournit un **cadre** mathématique structuré, avec des **équations** décrivant l'évolution des liaisons $\omega_{i,j}$ et des mesures de synergie multi-échelle (ou non linéaire). Il enrichit la **vision** d'une dose de **plasticité** et de **robustesse**, sans empêcher l'usage de CNN comme base de features. Les **clusters** qui émergent dans ce réseau renvoient à des regroupements de patches ou de motifs liés par leur **gain mutuel**, offrant un point de vue plus distribué et collaboratif qu'un pipeline hiérarchique unique. Au final, le **DSL** s'érige en complément précieux aux **CNN** pour atteindre une **vision** flexible, résiliente et progressivement évolutive.

1.6.2. Analyse Audio et Traitement du Langage Naturel

Après avoir vu comment le **Deep Synergy Learning (DSL)** peut apporter une flexibilité et une auto-organisation en **vision artificielle** (1.6.1), il est naturel de s'intéresser aux **données audio** et au **langage naturel**, deux domaines où la richesse des signaux et la variabilité contextuelle sont particulièrement élevées. Dans un réseau neuronal profond classique, les approches d'**analyse audio** (speech recognition, classification de sons, détection d'événements acoustiques) et de **traitement du langage naturel** (NLP) (analyse sémantique, traduction, question-réponse) demeurent puissantes, mais reposent souvent sur des modèles séparés (RNN, LSTM, Transformers) et des schémas d'apprentissage supervisé imposant de gros volumes de données annotées.

Le **DSL**, en revanche, propose une **auto-organisation** où chaque entité \mathcal{E}_i (issue de features audio ou textuelles) peut s'insérer dans un **réseau synergique** plus large, permettant :

12. Une **fusion plus organique** entre les éléments acoustiques ou linguistiques,
13. Une **adaptation continue** au bruit, aux accents, aux registres de langage,
14. Une cohabitation **symbolique–subsymbolique** pour des tâches de logique linguistique plus avancées.

Cette sous-section (1.6.2) détaille en quoi le DSL modifie l'approche habituelle de l'analyse audio et du traitement du langage, et en quoi il répond à des défis d'**hétérogénéité**, de **bruit**, de **variabilité linguistique** et de **scénarios évolutifs**.

1.6.2.1. Analyse Audio : Entités Sonores et Synergie Adaptative

Dans un **réseau** neuronal **classique** pour l'audio (CNN ou RNN appliqué aux spectrogrammes), on spécifie dès le départ un schéma d'extraction de *features* (MFCC, log-mel, etc.), puis l'on optimise un classifieur (par exemple, pour la reconnaissance de phonèmes ou la classification de sons). Le **Deep Synergy Learning (DSL)** introduit une logique plus **organique** en laissant des **entités** $\mathcal{E}_{\text{aud},k}$ se constituer pour chaque **fenêtre** ou **segment** temporel (frames audio), pour un **spectre** fréquentiel spécifique, ou pour un **embedding** plus abstrait (mots, bruits, etc.). Ces entités se regroupent en **clusters** dès lors qu'elles observent un **gain** à coopérer, qu'il s'agisse de la ressemblance spectrale, de la proximité temporelle ou de la récurrence d'un même motif acoustique.

Si on modélise deux segments audios $\mathbf{a}_i, \mathbf{a}_j \in \mathbb{R}^d$, leur synergie peut être définie par une **distance inversée** ou une **similitude**, telle que

$$S(\mathcal{E}_i, \mathcal{E}_j) = \exp(-\gamma \|\mathbf{a}_i - \mathbf{a}_j\|^2),$$

pour un certain $\gamma > 0$. La **pondération** $\omega_{i,j}(t)$ qui relie les entités \mathcal{E}_i et \mathcal{E}_j se met alors à jour selon la règle adaptative (voir section 1.4.5). Lorsque deux **fenêtres** ou segments audio affichent une forte similarité (timbre commun, bruit de fond identique, même locuteur), leurs liaisons $\omega_{i,j}$ se **renforcent**, initiant ainsi la création d'un **cluster** rassemblant ces entités, et permettant de **repérer** un même motif acoustique. Cette démarche d'**auto-organisation** remplace l'approche classique qui découpe l'audio selon des règles fixes : le réseau laisse les liaisons décider d'une segmentation ou d'une agrégation adaptative.

De plus, si un **environnement** se modifie (bruit important, micro défaillant, accent inhabituel), des entités précédemment utiles peuvent perdre leur **synergie** avec les autres composantes, faisant décroître $\omega_{i,j}$. L'entité se retrouve alors **isolée** et n'influence plus le cœur du réseau. Inversement, lorsqu'une nouvelle configuration sonore émerge (baisse du bruit, nouveaux sons caractéristiques), d'autres **connexions** ω apparaissent ou se renforcent, configurant de nouveaux **clusters** sonores. Ainsi, la **reconnaissance** de motifs acoustiques (sirènes, voix enfantines, grondements) se met en place de manière **continue**, sans qu'on doive réentraîner de bout en bout un réseau rigide. Le **DSL** s'adapte localement, confortant l'idée d'un système plus **résilient** et **évolutif** dans le traitement de l'audio.

1.6.2.2. Traitement du Langage Naturel : Entités Lexicales et Sémantiques

Dans les méthodes **classiques de traitement du langage naturel** (NLP), des modèles (RNN, LSTM, Transformers) s'appuient sur des **embeddings** (word2vec, GloVe, BERT, etc.) et des mécanismes d'attention ou de convolution. Bien qu'ils aient atteint d'excellents résultats (traduction, question-réponse, résumé, etc.), ils s'avèrent fortement dépendants d'un **entraînement** supervisé ou auto-supervisé de grande ampleur, et peinent à **intégrer** de la **logique** ou du **symbolique** sans recourir à des mécanismes additionnels. De plus, la **structure** de ces modèles demeure souvent linéaire (ou en arbre pour l'analyse syntaxique), imposant le traitement séquentiel des tokens.

Le **Deep Synergy Learning (DSL)** propose une alternative plus **flexible**. Au lieu d'imposer une séquence rigide de tokens à travers un pipeline (embedding + attention + couches finales), on peut **déployer** un **réseau** d'entités couvrant les différentes dimensions du texte. Chaque entité \mathcal{E}_i peut représenter un **mot** (ou token) muni d'un embedding \mathbf{w}_i , un **nœud syntaxique** extrait d'un parse tree (rôle de sujet, verbe, complément), ou un **concept sémantique** (topic, entité nommée, etc.). Ces entités s'**auto-organisent** si leur synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ s'avère suffisamment élevée pour justifier un **regroupement**.

On peut alors voir naître des **clusters** lexical-sémantique regroupant des mots liés à un même champ lexical, un **verbe** et ses **compléments**, ou plusieurs **concepts** partagés par diverses phrases. Contrairement à une **arborescence** imposée (comme dans une analyse syntaxique standard), le **SCN** peut se montrer plus **fluide** : si un mot possède plusieurs sens (polysémie), la synergie qu'il entretient avec certains topics oriente la sélection de celui-ci, ce qui peut désambiguïser l'interprétation. De même, des liaisons $\omega_{i,j}$ peuvent se **rompre** si un ensemble de mots cesse d'être pertinent dans un nouveau contexte textuel. L'**auto-organisation** rend donc la structure du **réseau** ajustable, permettant d'incorporer ou de dissoudre des sous-groupes de tokens ou de concepts selon qu'un **gain** mutuel se manifeste, ouvrant ainsi la voie à une **compréhension** plus modulable du langage.

1.6.2.3. Couplage Audio–Texte et Auto-Fusion Synergiques

De nombreux systèmes doivent traiter simultanément un **flux audio** (voix, intonations, bruits) et un **texte** associé (transcriptions, sous-titres). Le **Deep Synergy Learning (DSL)** s'avère particulièrement bien adapté à ce scénario, car il ne fixe pas a priori la manière de "joindre" l'audio et le texte. Au lieu d'un alignement imposé par un algorithme dédié, on laisse le **Synergistic Connection Network (SCN)** réguler les liaisons ω entre des **entités** \mathcal{E}_{aud} et des **entités** \mathcal{E}_{txt} . Si la **co-information** ou une autre mesure de **similarité** suggère qu'un certain segment sonore correspond à un certain segment textuel, la pondération $\omega_{\text{aud}, \text{txt}}$ se renforce, conduisant à la création d'un **cluster** multimodal.

Parmi les entités audio, on peut distinguer des segments temporels ou des frames associées à une **fenêtre** spectrale spécifique. Du côté textuel, on dispose d'entités représentant des **mots**, des **tokens**, voire des syntagmes plus longs. La mise à jour des liens $\omega_{\text{aud}, \text{txt}}$ procède alors par la règle adaptative (section 1.4.5), détectant les **paires** qui coïncident fréquemment. Ainsi, lorsqu'un segment audio s'aligne régulièrement sur une suite de mots, on observe un gain synergique. On forme donc un **cluster** englobant les parties orales et leurs correspondances textuelles, sans qu'un schéma d'alignement fixe soit nécessaire.

Un tel mécanisme confère une **robustesse** notable. Si des segments audio se montrent inintelligibles (forte perturbation, accent trop marqué), la **synergie** s'affaiblit, et les liaisons avec les entités textuelles cessent de croître, isolant ces segments pour ne pas polluer l'ensemble. Le réseau **accepte** ainsi l'hétérogénéité de la qualité sonore et s'auto-adapte. En outre, cette logique **permet de découvrir** des co-occurrences ou des associations inhabituelles, par exemple un mot (ou un motif lexical) qui se répète chaque fois qu'un type particulier de sonorité apparaît. Le **DSL** parvient donc à fusionner audio et texte de manière plus **flexible** que ne le ferait un pipeline d'alignement rigide, et il favorise l'intégration de nouvelles données ou la détection de nouveaux mots/sons sans devoir procéder à un ré entraînement ou une refonte exhaustive du modèle.

1.6.2.4. Évolutions Possibles et Défis Mathématiques

Dans l'application du **Deep Synergy Learning (DSL)** à l'audio, au texte ou à des mélanges de modalités, plusieurs difficultés et pistes d'extension apparaissent.

Un premier enjeu concerne la **complexité**. Comme en vision (section 1.6.1.4), on peut générer un grand nombre d'entités audio (fenêtres temporelles, frames) ou textuelles (tokens, syntagmes). Le **graph** qui en résulte peut ainsi atteindre une taille considérable, rendant la mise à jour des pondérations $\omega_{i,j}$ et l'estimation des synergies coûteuses. Il est donc souvent nécessaire de recourir à des **stratégies** parcimonieuses pour limiter la prolifération de liens, comme échantillonner périodiquement les entités, fusionner les entités jugées redondantes ou fixer un seuil ω_{\min} qui supprime les connexions trop faibles (section 1.4.5).

En second lieu, on notera que la **modélisation n-aire** peut se révéler essentielle. Les **synergies** binaires (audio–audio, texte–texte ou audio–texte) ne suffisent pas toujours à capturer des phénomènes impliquant trois (voire davantage) entités. Il se peut, par exemple, que l'union de deux fragments audio et d'un groupe de mots textuels produise une information inatteignable avec des paires isolées. La **synergie n-aire** (section 1.4.7) permet de refléter ces complémentarités plus complexes, au prix d'un accroissement de la difficulté mathématique, car il faut évaluer l'information ou la similarité simultanément sur plusieurs variables.

Enfin, la possibilité d'**hybridation symbolique** apparaît particulièrement intéressante dans le domaine du **NLP** avancé (raisonnement, question-réponse complexes). Comme évoqué en section 1.5.7, on peut introduire des **règles logiques** ou des **entités symboliques**, lesquelles interagissent avec les **entités textuelles**. La synergie se définit alors par une fonction reliant le sens des mots à des conditions logiques ou des assertions formelles. Si cette intégration accroît encore la complexité, elle apporte un surcroît de **capacité cognitive**, autorisant un raisonnement plus poussé et assurant une **explicabilité** renforcée, dans l'esprit d'une IA **neuro-symbolique** où la dynamique des pondérations répond à la fois aux critères sub-symboliques (similarité, co-information) et aux axiomes symboliques.

1.6.2.5. Conclusion Partielle : un Cadre Évolutif pour l'Audio et le Langage

Le **Deep Synergy Learning (DSL)** s'affirme comme une **approche** ou un **complément** aux architectures traditionnelles (CNN, RNN, Transformers) dans l'analyse de **flux audio** et le **traitement du langage naturel**. Contrairement aux pipelines classiques, le **DSL** ne se fonde pas sur un découpage préalable ou un alignement imposé, mais laisse les entités acoustiques ou lexicales s'**auto-organiser** dès que leur synergie l'indique. Cette logique confère plusieurs **avantages** :

L'**auto-organisation** pilote la segmentation et l'alignement de manière adaptative : si certains segments audio apparaissent corrélés ou si certaines paires de mots se révèlent complémentaires, leurs pondérations respectives se consolident, même en l'absence d'un découpage figé. Le **réseau** n'a pas à imposer un ensemble de tranches temporelles

ni une procédure d’alignement prédéterminée. Il s’en dégage une **robustesse** face aux sources de variations dans les flux audio (bruit, accents, modifications de la source) et face aux registres du langage (polysémie, style). En cas d’incertitude ou de qualité médiocre, les liens connexes s’affaiblissent sans qu’un réapprentissage général soit nécessaire.

La **fusion** audio–texte se trouve également facilitée : plutôt que d’exiger un couplage strict d’un segment sonore et de sa transcription, la dynamique interne du **DSL** découvre librement les correspondances, détectant les segments oraux et textuels qui coïncident dans le temps ou dans le sens. Cette méthode rend compte d’une **flexibilité** avancée, car le système peut absorber de nouveaux sons, un nouveau vocabulaire ou même des **règles symboliques** sans devoir être reconstruit. L’ajout d’entités inopinées dans le graphe suffit, et le **système** gère la mise à jour des pondérations.

Ces caractéristiques ouvrent la voie à plusieurs **applications** évolutives : la **reconnaissance vocale** en environnement bruité s’en trouve renforcée ; la **compréhension** de conversations multimodales (avec indices vidéo et audio) se gère sans pipeline d’alignement contraint ; la **traduction** flexible ou la **détection** d’anomalies audio-linguistiques profitent de la structure distribuée et dynamique. Le tout s’appuie sur les **règles d’auto-organisation** (pondérations adaptatives, constitution de clusters) et de **synergie** (distance, co-information, etc.), qui instaurent un fonctionnement moins tributaire d’un entraînement supervisé exhaustif et davantage tourné vers l’évolution continue et la **coopération** locale entre entités.

1.6.3. Robotique et Systèmes Intelligents Évolutifs

Les concepts d’**auto-organisation**, de **synergie** et de **coopération** entre entités, propres au **Deep Synergy Learning (DSL)**, prennent tout leur sens dans le domaine de la **robotique** et des **systèmes intelligents**. En effet, la robotique moderne se caractérise par :

- Une **intégration** de multiples capteurs (vision, audio, pression, position, etc.),
- Des **actions** diverses (mouvements, manipulations, interactions) à coordonner dans des environnements changeants,
- La nécessité d’une **adaptation continue** (à la dynamique physique, aux aléas de l’environnement),
- Des **défis** de planification et de décision temps réel, souvent avec de l’**incertitude** et du **bruit**.

Cette sous-section (1.6.3) montre comment le **DSL**, grâce à ses mécanismes de **pondérations adaptatives** et à ses **clusters auto-organisés**, peut être particulièrement utile à la **robotique** et aux **systèmes intelligents**. Elle décrit :

15. L’**approche distribuée** que le DSL propose pour les robots multi-capteurs,
16. La **coévolution** des entités sensorielles et motrices,
17. La capacité à s’**auto-réorganiser** lorsqu’un robot ou un système intelligent découvre de nouveaux modules (nouveaux effecteurs, nouveaux contextes, etc.),
18. Des exemples d’applications, depuis les **robots collaborateurs** jusqu’aux **systèmes autonomes** en changement permanent.

1.6.3.1. Robotique Multi-Capteurs : une Structure Synergique

Dans un contexte robotique, un système typique réunit plusieurs **capteurs** disposés sur la plateforme, pouvant inclure une **caméra** (analyse visuelle), un **LIDAR** ou un **radar** (cartographie de l’environnement), des **capteurs de pression** ou de **toucher**, des **capteurs inertIELS** (IMU), un **microphone** pour la partie audio, etc. Les approches traditionnelles (notamment basées sur ROS et des architectures préconçues) tendent à définir un **pipeline** de fusion, auquel chaque capteur envoie ses données, puis une **couche** de décision. Cette rigidité suppose un schéma de liaison explicite entre

capteurs et modules supérieurs, imposant par exemple un module de traitement conjoint image–LIDAR si l'on veut superposer un nuage de points laser et un flux de caméra.

Le **Deep Synergy Learning (DSL)**, quant à lui, permet d'instituer un **ensemble** d'entités $\{\mathcal{E}_{\text{cam}}, \mathcal{E}_{\text{lidar}}, \dots\}$ correspondant chacune à une source sensorielle ou à un bloc fonctionnel plus abstrait. Chacune de ces entités peut juger de la **synergie** qu'elle partage avec les autres : si la caméra \mathcal{E}_{cam} et le LIDAR $\mathcal{E}_{\text{lidar}}$ constatent régulièrement la détection d'un même obstacle, la valeur associée à leur liaison se consolide, menant à un **renforcement** de $\omega_{\text{cam}, \text{lidar}}$. Inversement, lorsqu'un capteur inertiel s'avère temporairement *bruité* (information incohérente par rapport aux autres lectures), ses connexions décroissent naturellement jusqu'à s'affaiblir et, le cas échéant, s'isoler.

Grâce à ce principe d'**auto-organisation**, il devient possible qu'un **cluster** multimodal émerge pour caractériser une situation telle qu'un “terrain accidenté” perçu simultanément par la caméra (textures complexes), le LIDAR (irrégularités de distance) et peut-être un micro détectant des bruits anormaux. Ce cluster se **dissout** ensuite lorsque le robot quitte la zone problématique ou que la configuration change. Le **réseau** d'entités agit donc de manière plus **dynamique** et **distribuée**, sans imposer un module de fusion central dédié à chaque couple de capteurs. Les liaisons $\omega_{i,j}$ se forment ou se dissolvent localement, en fonction de la **valeur ajoutée** mesurée à travers la synergie entre entités, conférant au système une **résilience** plus élevée, qu'il s'agisse de gérer un capteur défaillant, l'arrivée d'un nouveau module, ou un changement inattendu dans l'environnement.

1.6.3.2. Action, Effecteurs et Synergie Motrice

En robotique ou dans tout système intelligent disposant de **modules d'action** (moteurs, articulations, roues, pinces), on adopte généralement un paradigme où un **contrôleur** (PID, MDP, RL, etc.) reçoit un **état** en provenance des capteurs et génère des **commandes**. Le **Deep Synergy Learning (DSL)** aborde la question autrement, en considérant les effecteurs comme des **entités** $\mathcal{E}_{\text{motrice}}$ au même titre que les capteurs. Par exemple, chacune des articulations d'un bras, ou la rotation d'une roue, ou la position d'une pince peut être dépeinte par un vecteur $\mathbf{x}_{\text{motrice}}$. La **coopération** avec les entités sensorielles s'établit dès lors que la **synergie** $S(\mathcal{E}_{\text{capteur}}, \mathcal{E}_{\text{motrice}})$ s'avère positive, incitant à **renforcer** la pondération $\omega_{\text{capteur}, \text{motrice}}$.

Un bras articulé qui coordonne son mouvement avec un flux visuel voit, par exemple, la pondération $\omega_{\text{cam}, \text{bras}}$ croître si la caméra confirme que la pince se trouve dans la bonne position. À l'inverse, un joint dysfonctionnel ou mal calibré cesse d'apporter un réel gain, ce qui fait chuter la **synergie**, rendant son influence moindre dans la stratégie globale. Le **SCN** assure ainsi une forme d'**adaptation** où les effecteurs pertinents coopèrent étroitement avec les capteurs produisant des informations utiles, sans qu'il soit nécessaire de configurer explicitement un contrôleur central devant orchestrer toutes les modalités.

On peut également insérer un **signal** de récompense ou un indicateur de performance R qui module la synergie. Si la coopération entre un capteur $\mathcal{E}_{\text{capteurA}}$ et un effecteur $\mathcal{E}_{\text{motriceB}}$ accroît ce score (par exemple, la manipulation d'un objet est mieux réalisée, ou la trajectoire est plus stable), la **pondération** $\omega_{\text{capteurA}, \text{motriceB}}$ s'en trouve augmentée. Cette logique correspond à un **processus** de “learning by synergy”, sans avoir à recourir à une formalisation en MDP (Markov Decision Process) rigide. Les entités effectrices et sensorielles **s'auto-organisent** localement via la mise à jour des connexions ω , découvrant progressivement quelles combinaisons capteur–action se révèlent fructueuses pour atteindre l'objectif.

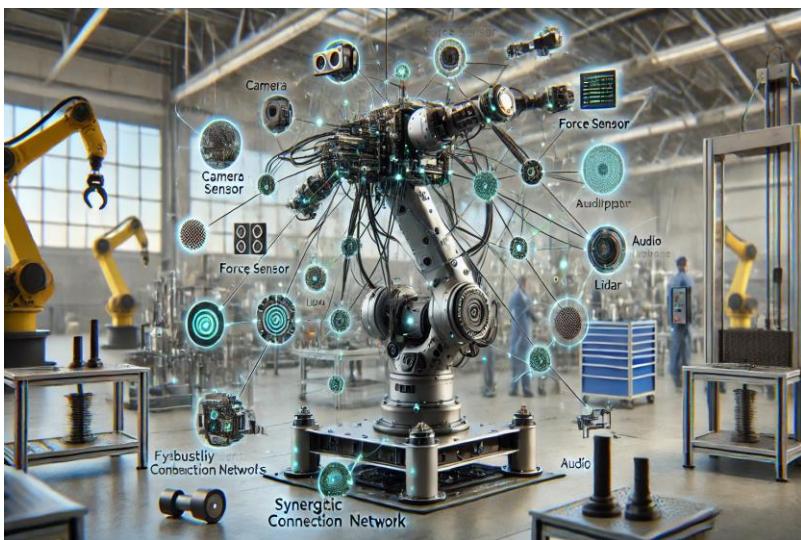
1.6.3.3. Adaptation Continue et Évolution Modulaire

Un des atouts fondamentaux du **Deep Synergy Learning (DSL)** tient dans sa **capacité d'évolution** : lorsqu'un robot se voit doté d'un nouveau **capteur**, d'un **nouvel effecteur** (par exemple un autre bras), ou que l'on retire un module obsolète, on se contente d'**ajouter** (ou de **supprimer**) l'entité correspondante $\mathcal{E}_{\text{nouveau}}$ au sein du **Synergistic Connection Network (SCN)**. Les pondérations ω_{nouveau} suivent la dynamique habituelle (voir section 1.4.5) et se **renforcent** ou se **dissolvent** en fonction de la **synergie** détectée. Il n'est donc pas requis de recompiler l'ensemble du pipeline ou de procéder à un **réapprentissage** complet d'un grand réseau neuronal. La **logique** adaptative se borne à évaluer localement la valeur ajoutée de la nouvelle entité, puis à l'intégrer dans un **cluster** pertinent si besoin.

Cette **propriété** ne se limite pas à la **robotique** physique. Dans des **systèmes logiciels** complexes, par exemple dans l'administration de centres de données ou d'architectures distribuées, le principe reste valable. Chaque microservice, base de données, module de monitoring, etc., peut être considéré comme une **entité** $\mathcal{E}_{\text{service}}$. L'arrivée d'un nouveau service, la suppression (ou la panne) d'un service existant, se modélise par l'**ajout** ou le **retrait** d'entités correspondantes, et les **liaisons** ω se régulent selon la **coopération** réelle (échange d'informations, dépendances). Les **clusters** qui en résultent rassemblent les **services** ayant des affinités ou co-occurrences fortes, autorisant une **répartition** et une **coordination** modulaires.

Ce **principe d'auto-organisation** évite la centralisation rigide d'un pipeline complet ou la révision constante d'un réseau monolithique. Au contraire, le **SCN** se reconfigure **spontanément** : il repère quels éléments demeurent fiables et utiles, et il restructure graduellement l'architecture face à de nouveaux besoins ou de nouvelles entités. Cette démarche s'inscrit dans l'esprit du **lifelong learning**, où le **réseau** lui-même façonne en continu sa **topologie** et maintient un équilibre entre la conservation de connaissances antérieures et l'ajout de briques inédites.

1.6.3.4. Illustration : Robot Collaboratif Multicapteurs



Un **robot collaboratif** (cobot) dans un environnement industriel peut disposer de multiples capteurs et actionneurs. Par exemple, on considère une **caméra** pour localiser la pièce à assembler, un **capteur de force** dans la pince pour jauger la préhension, un **LIDAR** détectant la distance aux opérateurs humains pour prévenir les collisions, un **module audio** apte à recevoir des instructions vocales, et enfin les actionneurs : la **pince**, le **bras articulé** et un **moteur** de déplacement sur plateforme.

Au sein d'un **Synergistic Connection Network** (SCN) selon les principes du **Deep Synergy Learning (DSL)**, chaque composant devient une **entité** : $\mathcal{E}_{\text{cam}}, \mathcal{E}_{\text{force}}, \mathcal{E}_{\text{lidar}}, \mathcal{E}_{\text{audio}}$ pour les capteurs, et $\mathcal{E}_{\text{pince}}, \mathcal{E}_{\text{bras}}, \mathcal{E}_{\text{moteur}}$ pour les effecteurs. Chacune évalue localement sa **synergie** avec les autres, de sorte que les liens $\omega_{i,j}$ se **renforcent** s'il apparaît un réel bénéfice à leur coopération. Lorsque la caméra coopère régulièrement avec la pince — par exemple, parce que la zone repérée par l'image correspond effectivement à la position où la pince intervient —, la liaison $\omega_{\text{cam}, \text{pince}}$ se consolide, engendrant un **cluster** local regroupant ces entités $\{\text{cam}, \text{pince}\}$. Si, au contraire, l'audio se révèle trop bruité, son synergie avec le reste du système diminue et $\omega_{\text{audio}, \cdot}$ tombe en deçà du seuil de pertinence ; l'entité audio se retrouve alors **isolée** ou peu contributive jusqu'à ce que la qualité du signal s'améliore.

On peut intégrer un **signal de récompense** afin de consolider les combinaisons sensorielles et motrices les plus efficaces (par exemple, si le cobot réussit à saisir et assembler la pièce sans heurter de zones sensibles, on valorise cette configuration). Les entités impliquées voient leurs pondérations internes augmenter, ce qui renforce la probabilité de réutiliser ce mode de coopération. L'ensemble constitue un **réseau** qui s'**auto-adapte**, insérant ou affaiblissant des liens selon l'utilité détectée pour la tâche courante. Cette logique évite de reconstruire un pipeline complet ou de réentraîner un vaste réseau en cas de modifications (ajout d'un nouveau capteur, panne de l'actuel, variation des

conditions d'assemblage). Les **clusters** multimodaux (vision + pince + capteur de force, par exemple) forment des **sous-réseaux** cohérents, naturellement conçus pour la tâche à effectuer.

1.6.3.5. Défis et Perspectives de Recherche

Lorsqu'on applique le **Deep Synergy Learning (DSL)** à la robotique multi-capteurs et multi-effecteurs, plusieurs difficultés et voies d'extension émergent, tant sur le plan pratique que théorique.

Un premier défi concerne la **complexité combinatoire**. Dans un robot doté de multiples capteurs (caméra, LIDAR, IMU, microphone...) et d'autant de modules d'action (divers joints, roues, pinces), le **graphe** des pondérations ω peut rapidement croître en taille, rendant la mise à jour de chaque lien difficile à maintenir en **temps réel**. Des stratégies de **parsimonie** (seuils éliminant les liaisons trop faibles) ou des mises à jour strictement **locales** (limiter le calcul aux entités concernées par une nouvelle donnée) se révèlent indispensables pour garantir la **scalabilité**.

Un deuxième point aborde les **systèmes multi-robots**. Lorsqu'une flotte (drones, véhicules, robots industriels) doit coopérer, on peut concevoir un **réseau synergique** à plus grande échelle, où chaque robot forme un **sous-graphe** d'entités sensorielles et motrices. Les **synergies** inter-robots introduisent alors des liaisons ω entre entités de différents appareils. Il devient possible d'imaginer une **co-organisation** sur l'ensemble de la flotte, favorisant la dynamique d'émergence de clusters inter-robots dès qu'un gain mutuel apparaît (partage de capteurs, répartition de tâches). L'envergure de ce problème soulève d'importants défis de **communication** et de gestion **distribuée** des liens, car on ne peut pas nécessairement centraliser toutes les informations.

La **sécurité** et la **robustesse** forment également un axe de recherche majeur. En robotique, il est impératif de respecter des contraintes physiques (éviter la collision, maintenir une distance de sécurité). Dans le **DSL**, on peut introduire des pénalités ou des **contraintes** pour réduire la pondération $\omega_{i,j}$ lorsqu'une configuration s'avère dangereuse ou enfreint des invariants mécaniques ou de sûreté. De cette manière, l'**auto-organisation** se soumet à des garde-fous, garantissant un comportement sûr même lorsque la structure du réseau évolue.

Enfin, l'idée d'**hybridation symbolique** (section 1.5.7) se transpose à la robotique avancée et aux tâches haut niveau. Les entités symboliques, représentant des **règles** ou des **plans**, peuvent interagir avec les entités sensorielles et motrices : la synergie entre la **logique** (définissant un plan d'action) et la **perception** (reconnaissance visuelle, retour de force) oriente la réorganisation du graphe, débouchant sur une planification distribuée et **cognitive**, où la coopération se fait au croisement du symbolique (ordre de mission) et du sub-symbolique (capteurs, effecteurs). L'ensemble de ces pistes ouvre un **vaste** espace de recherche pour faire évoluer la robotique vers des systèmes de décision auto-organisés, hautement modulaires et résilients.

Conclusion

Dans la **robotique** et les **systèmes intelligents évolutifs**, le **Deep Synergy Learning** :

- Offre une **approche distribuée** pour intégrer de multiples capteurs et effecteurs,
- Gère la **reconfiguration** continue (bruit, arrivée de nouveaux modules, pannes),
- Favorise l'**apprentissage local**, la formation de **clusters** sensorimoteurs stables,
- Permet un fonctionnement **plus organique** que le pipeline rigide (capteur → fusion → décision), et plus extensible face à l'évolution de l'architecture robotique.

Cette capacité d'**auto-organisation** s'avère essentielle lorsque le robot (ou le système) doit s'ajuster rapidement, adopter de nouveaux modules ou se coordonner avec d'autres entités (robots, agents, microservices). On retrouve ainsi la philosophie déjà mise en avant dans le DSL : éviter la hiérarchie monolithique et le réapprentissage complet, au profit d'une **dynamique adaptative**, guidée par la **synergie** entre entités.

1.6.4. Recommandation Personnalisée et Systèmes de Décision

Au-delà de la robotique (1.6.3), le **Deep Synergy Learning (DSL)** trouve également une place de choix dans les **systèmes de recommandation** et plus largement les **systèmes de décision**. Dans ces domaines, il s'agit souvent de traiter des **profils utilisateurs**, des **contenus** (produits, articles, films, musiques) ou des **options** (plans d'action, configurations possibles), pour aboutir à des **conseils** ou des **choix** pertinents. Les approches traditionnelles (collaborative filtering, réseaux neuronaux de recommandation, arbres de décision) présentent généralement des structures fixes : on définit un modèle qui, après un entraînement plus ou moins supervisé, calcule des scores de préférence.

Le DSL, par sa **dynamique d'auto-organisation** et sa **prise en charge** des liens synergiques, propose une voie plus **flexible** et **adaptative**. Il peut gérer à la fois :

- Des **entités “utilisateurs”** $\{\mathcal{E}_u\}$ représentant différentes personnes (ou profils),
- Des **entités “contenus”** $\{\mathcal{E}_c\}$ décrivant articles, produits, médias, etc.,
- Des **entités “contextes”** (saisons, heures, tendances récentes),
- Éventuellement des **entités symboliques** (règles de business logic, préférences explicites) si on veut un système cognitif plus avancé.

Nous allons voir en détail comment ces entités interagissent et comment le DSL peut aider à **adapter** les recommandations au fil du temps, en favorisant les **clusters** (ou micro-réseaux) les plus cohérents, et en révisant les connexions obsolètes.

1.6.4.1. Entités Utilisateurs et Contenus : un Graphe Synergique

Les **systèmes de recommandation** s'appuient classiquement sur une **matrice M** dont les lignes représentent les utilisateurs $\{u \in U\}$ et les colonnes les contenus $\{c \in C\}$. Chaque entrée $M_{u,c}$ correspond à l'intérêt ou la note attribuée par l'utilisateur u au contenu c . Les approches courantes (collaborative filtering, factorisation matricielle, autoencodeurs de complétion...) consistent alors à factoriser **M** ou à prédire les valeurs manquantes par un entraînement supervisé ou semi-supervisé. Cette vision demeure matricielle : on y ajoute rarement d'autres entités hors du couple (utilisateurs, contenus).

Dans le **Deep Synergy Learning (DSL)**, on substitue ou complète cette approche en définissant un **graphe** évolutif. On introduit d'abord un ensemble d'**entités** $\{\mathcal{E}_u\}_{u \in U}$ représentant les utilisateurs et un autre $\{\mathcal{E}_c\}_{c \in C}$ pour les contenus (articles, films, produits, etc.). Si nécessaire, on y ajoute aussi des **entités contextuelles** $\{\mathcal{E}_{\text{time}}, \mathcal{E}_{\text{location}}, \dots\}$ lorsqu'on souhaite prendre en considération la dimension temporelle ou spatiale. Les **liaisons** $\omega_{u,c}(t)$ reliant un utilisateur \mathcal{E}_u à un contenu \mathcal{E}_c sont mises à jour suivant une **règle adaptative** (voir section 1.4.5) : elles se renforcent si la “synergie” $S(\mathcal{E}_u, \mathcal{E}_c)$ est jugée suffisamment positive, ou au contraire déclinent si aucune utilité mutuelle n'est perçue. La fonction S peut reposer sur une **similarité** d'intérêts, une **co-information** d'usage, ou sur l'idée d'un gain mesurable (par ex. le nombre de clics ou de vues).

Grâce à ce schéma, on assiste à la formation spontanée de **clusters** regroupant simultanément des utilisateurs et des contenus. Un **cluster** peut ainsi rassembler plusieurs amateurs de rock, divers albums rock et, s'il existe des entités de contexte, certaines soirées “concert” associées, révélant une forte **co-occurrence** ou un alignement d'intérêts. Comparé à la matrice **M** — qui se borne aux couples (u, c) — le **SCN** tolère l'ajout de toute entité auxiliaire. Il devient possible de **fusionner** des groupes si une synergie apparaît entre leurs membres, ou d'**isoler** un contenu dont l'intérêt décroît soudain auprès des utilisateurs concernés. En outre, l'arrivée d'un nouvel utilisateur $\mathcal{E}_{\text{nouveau}}$ ou d'un nouveau contenu $\mathcal{E}_{\text{item}}$ se gère en insérant simplement l'entité, puis en laissant la synergie ajuster les liens $\omega_{\text{nouveau}, \cdot}$. Le réseau se **restructure** localement, sans imposer une refonte globale comme le ferait une ré-analyse complète de la matrice **M**. Les sections suivantes (voir 1.6.4.2 et au-delà) approfondiront comment cette logique d'auto-organisation enrichit la **recommandation** par la capacité à intégrer de nouvelles entités et à effectuer une fusion dynamique des préférences et des contenus.

1.6.4.2. Mise à Jour Adaptive et Évolution dans le Temps

Dans un **système de recommandation**, la **distribution** se modifie sans cesse : il apparaît de nouveaux utilisateurs, de nouveaux contenus (articles, livres, films récents), des préférences se réorientent au fil des modes, des promotions ou des saisons. Les algorithmes classiques de filtering (factorisation, deep learning supervisé) doivent alors réestimer leurs paramètres, au risque d'exiger une nouvelle phase de réapprentissage ou un raffinage partiel, avec le danger de dérégler l'équilibre déjà atteint.

Le **Deep Synergy Learning (DSL)** adopte une démarche plus organique. Lorsque des entités inédites $\mathcal{E}_{\text{nouveauUser}}$ ou $\mathcal{E}_{\text{nouveauContenu}}$ apparaissent, on les **insère** directement dans le **Synergistic Connection Network**. Le mécanisme de mise à jour $\omega_{i,j}$ (section 1.4.5) se charge alors de **consolider** ou non les liaisons selon la **synergie** qu'elles dégagent. Ainsi, un nouveau contenu sera rapidement lié (pondérations positives) aux utilisateurs qui partagent des goûts proches ou qui interagissent avec ce contenu, conduisant à la formation d'un **cluster**. À l'inverse, un contenu ancien qui perd de l'intérêt voit ses pondérations $\omega_{u,\text{contenu}}$ décroître naturellement, jusqu'à s'isoler si plus personne ne s'y intéresse.

De plus, le **DSL** n'est pas cantonné aux seules entités "utilisateur" ou "contenu". Des **entités contexte** comme la **saison** ($\mathcal{E}_{\text{été}}$), la **localisation** ($\mathcal{E}_{\text{Paris}}$), un **événement** ($\mathcal{E}_{\text{Noël}}$) peuvent influencer la **synergie**. Si la proximité entre un événement (Noël), certains contenus (articles cadeaux, recettes festives) et un profil d'utilisateurs se concrétise, alors les liens $\omega_{\text{Noël},\text{contenu}}$ et $\omega_{\text{Noël},\text{utilisateur}}$ s'amplifient, façonnant un **cluster** "Noël + X utilisateurs + Y contenus". Lorsque la période de Noël s'achève, ces mêmes liaisons perdent peu à peu de leur attrait et se **délèvent**, laissant place à d'autres configurations contextuelles (telles que "soldes de janvier"). Le réseau se **reconfigure** ainsi localement, sans imposer un ré entraînement massif ni une réinitialisation de tout le modèle, et garantit une adaptation continue au gré des changements de **distribution**.

1.6.4.3. Décision et Recommandation Basées sur les Pondérations Synergiques

Une fois le **réseau** de synergie $\{\omega_{i,j}\}$ formé ou relativement stabilisé (au moins à un instant donné), il devient possible d'en **extraire** un score de recommandation pour un **utilisateur** \mathcal{E}_u et un **contenu** \mathcal{E}_c . Une formule concevable, inspirée d'une multiplication matricielle partielle, consiste à sommer l'**influence** indirecte par d'autres entités $\{\mathcal{E}_k\}$ (utilisateurs proches, tags contextuels, règles). On peut ainsi écrire :

$$\text{score}_{u,c} = \sum_k \omega_{u,k} \omega_{k,c}.$$

Intuitivement, si l'utilisateur \mathcal{E}_u et un contenu \mathcal{E}_c ne partagent pas de lien direct, on évalue leur **affinité** en passant par des entités \mathcal{E}_k qui peuvent être d'autres **utilisateurs** (similaires) ou des **concepts** (genre musical, catégorie d'article). Cet agrégat de contributions rappelle la structure de $(W^2)_{u,c}$ dans le cadre d'une multiplication de la matrice des pondérations W par elle-même, sans s'y limiter nécessairement : des **variantes** plus subtiles peuvent être introduites pour mieux prendre en compte les chemins les plus pertinents ou un **cluster** local.

Lorsqu'on inclut des **entités symboliques** (section 1.5.7), portant des **règles** ou **contraintes** métier (par exemple "Ne pas recommander les contenus PG-13 à un compte junior"), on peut autoriser la **synergie** à se moduler en conséquence : si la liaison $\omega_{\text{rule},\text{contenu}}$ indique une incompatibilité (règle violée), elle réduit la pondération globale $\omega_{\text{user},\text{contenu}}$. Il en résulte un **réseau** qui intègre à la fois les **préférences** non supervisées (co-occurrences utilisateurs/contenus) et des **limites** ou **politique** explicites (règles symboliques). Le **SCN** en tient compte localement, ce qui équivaut à un **filtrage** ou à un **re-rank** dynamique, plus flexible qu'un pipeline figé. La **recommandation** se rapproche alors d'une conclusion collective de l'ensemble des entités, chaque liaison ω traduisant la **valeur** ou la **compatibilité** mutuelle.

1.6.4.4. Illustrations Concrètes

Plusieurs exemples concrets permettent de mieux percevoir la **souplesse** et la **richesse** du **Deep Synergy Learning (DSL)** appliqué au domaine des systèmes de recommandation ou, plus largement, à l'**aide à la décision** sur des ensembles utilisateur–contenu–contexte.

Dans une **plateforme e-commerce**, on dispose d'entités représentant les **utilisateurs** (avec leur historique et profil), les **produits** (catégories, marques, etc.), et d'éventuelles entités de **contexte** (temps fort comme soldes, promotions, événements saisonniers). Au lieu de programmer explicitement des algorithmes de segmentation, le **DSL** laisse ces entités s'**agréger** en **clusters** si elles constatent un **gain** mutuel. Un sous-groupe d'utilisateurs peut ainsi s'avérer très intéressé par des produits d'une certaine marque ou style, formant un **cluster** qui se renforce dès lors que la synergie (p. ex. co-occurrence d'achats) s'avère constante. Les thématiques détectées évoluent naturellement en fonction des tendances ou de l'arrivée de nouvelles gammes de produits.

Dans un **service de streaming** (musique, vidéo), des entités peuvent correspondre aux **albums**, aux **artistes**, aux **playlists**, aux **tags** de genres (rock, pop, jazz), ainsi qu'aux **communautés** d'auditeurs. Le mécanisme de mise à jour $\omega_{i,j}$ valorise la proximité entre un auditeur et un certain artiste, ou entre un artiste et un tag de genre, etc. Un **cluster** cohérent peut alors émerger autour de l'auditoire d'une tendance musicale particulière (rock), intégrant les artistes, les playlists populaires associées et les auditeurs fidèles. Si un nouvel artiste rejoint la plateforme, on ajoute une entité $\mathcal{E}_{\text{artisteNew}}$; si celle-ci partage des caractéristiques avec le cluster rock (timbre de voix, style, tags), la **synergie** se trouve élevée, et l'artiste s'intègre au groupe. Les **recommandations** pour un auditeur donné s'en déduisent en analysant $\omega_{x,\text{artiste}}$ ou bien $(W^2)_{x,\text{artiste}}$ (cf. section 1.6.4.3), reflétant l'influence indirecte par d'autres auditeurs ou tags.

Dans un **système d'aide à la décision** industriel, on peut imaginer des entités décrivant différents **scénarios** (configurations de machines, paramètres de production), des **résultats passés** (qualité, temps de cycle), ainsi que des entités relatives aux **coûts** ou aux **ressources** (personnel, stocks). Les **liaisons** $\omega_{i,j}$ se renforcent lorsqu'une **combinaison** de paramètres s'avère fructueuse pour la production, ou que deux scénarios se révèlent similaires et conduisent à un résultat positif. Le **DSL** élabore progressivement une **cartographie** flexible de l'espace décisionnel : de nouveaux scénarios peuvent être ajoutés sans réapprentissage global, et l'**auto-organisation** s'adapte si les conditions de production évoluent (matières premières, cadence, etc.). Le tout offre une vision plus **distribuée** et **réactive** que des approches rigides, car les entités ne cessent de mettre à jour leurs liaisons pour refléter la **synergie** détectée.

1.6.4.5. Avantages et Défis

Dans un **système de recommandation** traditionnel, la construction d'un **pipeline** rigide est largement répandue, comme le collaborative filtering avec factorisation de matrice ; on effectue alors une estimation statique ou périodique, puis on l'emploie pour recommander. Le **Deep Synergy Learning (DSL)** propose au contraire une **auto-organisation** plus dynamique : lorsque les préférences d'un utilisateur fluctuent, ou lorsqu'un contenu quitte la base, la mise à jour des connexions ω s'effectue localement, évitant un recalcul global. Cette **souplesse** améliore la réactivité à des évolutions telles que les promotions, la saisonnalité ou l'arrivée de nouveaux items.

On gagne aussi une **robustesse** et une certaine forme de **parsimonie** : des entités isolées (utilisateurs inactifs, contenus devenus obsolètes) perdent naturellement leurs liens ω si leur synergie demeure trop faible. Elles n'interfèrent plus avec les autres recommandations, réduisant ainsi le **bruit**. Néanmoins, un **défi** majeur se manifeste si le nombre d'entités monte en flèche : il faut mettre en place des **heuristiques** (sparsification, échantillonnage, etc.) pour maintenir la faisabilité en grand volume (big data). Autrement, la multiplication des liaisons pourrait saturer les ressources de calcul.

Le **DSL** autorise aussi une **évolution contextuelle** : on peut introduire une nouvelle entité “festival d'automne”, “temps pluvieux” ou “Black Friday” pour moduler en temps réel la synergie. Les liens ω indiquent alors quels utilisateurs et quels contenus sont activés par ce contexte, sans qu'un réapprentissage massif soit nécessaire. Cela favorise une adaptation **continue** au gré des événements.

En dernier lieu, l'**explicabilité** s'en voit renforcée. Au lieu de recourir à des facteurs latents obscurs (F1, F2...), un **cluster** émergent peut être **interprété** plus directement : il rassemble un groupe d'utilisateurs et un sous-ensemble précis de contenus. Dans un même sous-réseau, l'entité “jazz” (tag de genre) et l'entité “club de blues” (contenu) peuvent se consolider avec un sous-groupe d'auditeurs, ce qui donne une **raison** explicite : “vous appartenez au cluster jazz-blues, d'où cette recommandation”. Cette lisibilité dépasse la vision d'une factorisation matricielle standard, offrant un gain en transparence et facilitant la justification des suggestions.

Conclusion

Dans la **recommandation personnalisée** et les **systèmes de décision**, le **Deep Synergy Learning** se distingue par :

- Sa **capacité** à gérer une **multiplicité** d'entités (utilisateurs, contenus, contextes, règles),
- Son **auto-organisation continue**, autorisant l'ajout ou le retrait de modules sans réapprentissage global,
- Sa **formation de clusters** plus interprétables, où les associations (utilisateurs—contenus—contexte) émergent de la **synergie** plutôt que d'une factorisation imposée,
- Son **adaptation** en temps réel aux changements d'habitudes, de saison, de modes ou de distributions.

Avec le DSL, les **recommandations** et **décisions** se construisent dans un **réseau** évolutif, rendant possible une **cohérence** plus fluide avec des règles (symboliques), des signaux contextuels, ou des flux externes. C'est un champ où l'auto-organisation synergique peut apporter plus de **plasticité** et de **transparence** qu'un pipeline classique de filtering ou de scoring.

1.6.5. Surveillance, Diagnostic Médical et Anomalies

Les principes d'**auto-organisation** et de **coopération adaptive** propres au **Deep Synergy Learning (DSL)** ne se cantonnent pas aux domaines précédemment cités (vision, audio, robotique, recommandation). Ils trouvent également une place toute particulière dans les **applications de surveillance**, de **diagnostic**, et de **détection d'anomalies**, qu'il s'agisse de contexte médical ou industriel. Dans ces environnements, on gère souvent d'importants **flux de données** en temps réel, provenant de **capteurs variés** (biologiques, physiques) qu'il faut **coordonner** pour repérer des situations anormales ou des symptômes précoce. Les systèmes classiques reposent sur des algorithmes d'**anomaly detection** (souvent supervisés, semi-supervisés, ou basés sur des seuils statiques) qui peuvent se révéler rigides ou trop dépendants d'un jeu de labels.

Le **DSL** introduit une **flexibilité** nouvelle : des **entités d'information** spécialisées dans différents signaux (capteurs, paramètres cliniques, logs machines), liées par des **pondérations synergiques** qui évoluent au fil du temps. Ainsi, l'**auto-organisation** permet de détecter des groupes d'observations cohérentes et, réciproquement, d'identifier des entités qui s'**écartent** de la synergie générale. Cette capacité de **repérage** local et de **réaction** dynamique rend le DSL particulièrement indiqué pour :

19. Les systèmes de **surveillance** (vidéo, capteurs environnementaux) qui doivent isoler des comportements suspects,
20. Les **diagnostics médicaux** complexes, où l'on fusionne imagerie, tests biologiques, relevés de signes vitaux pour repérer une pathologie,
21. L'**analyse d'anomalies** en maintenance prédictive, où un capteur ou un indicateur diverge du comportement habituel du cluster.

Nous allons voir plus en détail comment le DSL prend en compte la variété des signaux et repère, via la synergie (ou son absence), ce qui sort du lot et pourrait constituer une alerte ou un symptôme.

1.6.5.1. Systèmes de Surveillance : Entités Multi-Capteurs, Scènes Vidéo

Dans de nombreux dispositifs de **surveillance vidéo**, les algorithmes classiques (souvent un pipeline à base de CNN et de suivi d'objets) doivent gérer un **flot** de données considérable et détecter en continu tout comportement ou événement anormal. Cette approche peut se révéler fragile face à la complexité des scènes ou à la diversité des conditions. Le **Deep Synergy Learning (DSL)** propose une alternative plus **auto-organisée** et **réactive**, en représentant chaque composante (segment visuel, contexte, schéma habituel) sous forme d'entités et en laissant la **synergie** orienter la structuration.

Dans le cadre d'une **vidéo** de surveillance, on peut définir des entités $\mathcal{E}_{\text{seg},k}$ correspondant à différents **blobs** ou **segments** détectés dans la scène (régions de mouvement, silhouettes, etc.). D'autres entités $\mathcal{E}_{\text{contexte}}$ décrivent, par exemple, l'heure, la zone surveillée, les règles de fréquentation habituelles. Enfin, des entités $\mathcal{E}_{\text{pattern}}$ peuvent caractériser des **patrons** récurrents (le nombre moyen de personnes, la vitesse de déplacement habituelle, les régions autorisées ou interdites). Au sein du **Synergistic Connection Network**, les liaisons ω se forment ou se réduisent selon la **cohérence** repérée : si un segment vidéo "blob A" s'inscrit habituellement dans la vitesse ou l'emplacement prévus, la synergie demeure haute. Lorsqu'un comportement s'écarte de l'ordinaire (un mouvement inhabituel, une densité de blobs anormale, etc.), la **pondération** $\omega_{\text{seg, pattern}}$ chute, isolant le segment considéré et signalant une anomalie. L'algorithme peut alors émettre une alarme ou enclencher une inspection spécifique, sans nécessiter de recompiler ou de réentraîner un détecteur complet.

Dans la **surveillance** élargie, on ajoute souvent d'autres capteurs : mouvement (PIR), bruit, température, fumée, etc. Le **DSL** traite aisément cette hétérogénéité en insérant des **entités** $\mathcal{E}_{\text{mouvement}}$, \mathcal{E}_{son} , $\mathcal{E}_{\text{température}}$, dont les synergies **s'auto-régulent** avec les entités visuelles. Si un choc violent est détecté par \mathcal{E}_{son} alors que la vidéo ne rapporte aucun mouvement, la liaison $\omega_{\text{son, cam}}$ diminue, suggérant un capteur défaillant ou un phénomène localisé sans trace visuelle. À l'inverse, la convergence de plusieurs capteurs (caméra détectant flammes, capteur son captant un "boom", etc.) renforce les liaisons, formant un **cluster** d'alerte qui intègre différentes sources. Cette **auto-organisation** rend la détection plus **résiliente** : les signaux douteux se voient mis à l'écart, tandis que les coïncidences réelles entre capteurs suscitent un regroupement cohérent, facilitant ainsi l'identification d'événements critiques.

1.6.5.2. Diagnostic Médical : Entités Physiologiques et Multimodalité



Dans un **contexte médical**, les données proviennent de multiples **sources**. On dispose de l'**imagerie** (IRM, rayons X, scanner), de **tests biologiques** (formule sanguine, biochimie), de **signes vitaux** (ECG, pression, saturation en O₂) et de **symptômes** rapportés, voire de données **génétiques** plus complexes. Un **réseau** de neurones classique, comme un CNN spécialisé dans l'interprétation d'IRM, a tendance à se focaliser sur un **type** de signal. Or, certains diagnostics difficiles (cancers rares, syndromes complexes) exigent un **croisement** entre diverses modalités, chacune partielle, parfois bruitée.

Le **Deep Synergy Learning (DSL)** se conçoit alors comme un **réseau** auto-organisé où des **entités** $\{\mathcal{E}_{\text{irm}}, \mathcal{E}_{\text{analyseSang}}, \mathcal{E}_{\text{ecg}}, \dots\}$ représentent chacune un bloc de données ou un test particulier. Le **score** de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ indique si deux sources convergent vers une suspicion commune (par exemple, des anomalies visibles à l'IRM qui correspondent à des marqueurs sanguins inhabituels). Les **pondérations** $\omega_{i,j}$ se renforcent lorsque la coopération de deux entités apporte un **gain** (déttection corrélée), et un **cluster** pathologique peut se constituer, regroupant un sous-ensemble de paramètres (imagerie, analyses, symptômes) indiquant une même pathologie ou un syndrome précis.

De plus, l'évolution **temporelle** des données médicaux (phases aiguës ou rémissions) se gère de façon fluide : si un paramètre redevient normal, la synergie qu'il entretenait avec les marqueurs pathologiques s'épuise, et le **cluster** de la maladie se désolidarise progressivement de cette entité. Inversement, un nouveau symptôme peut s'y **agréger** dès qu'il présente une similarité ou co-occurrence notable avec d'autres facteurs pathologiques existants. Le **diagnostic** s'ajuste donc pas à pas, reflétant l'**état** du patient à chaque instant. On peut également insérer un **signal** de validation clinique ou de "récompense" lorsqu'un diagnostic s'avère correct, ce qui renforce les liens responsables de la bonne détection. Cette logique **auto-organisée**, distribuée et multimodale se distingue d'un pipeline rigide appliquant un classifieur sur l'imagerie ou l'analyse sanguine isolément : elle intègre **toutes** les sources, les pondérations ω révélant la **coopération** la plus profitable à un moment donné.

1.6.5.3. Détection d'Anomalies Industrielles : Auto-Organisation et Isolement

En milieu **industriel** (production, logistique, transports), la **maintenance prédictive** implique une surveillance continue des capteurs (température, vibrations, courants moteurs, etc.) afin de détecter des signaux précurseurs de panne. Dans un **Deep Synergy Learning (DSL)**, on peut représenter chaque **capteur** par une entité $\mathcal{E}_{\text{capteur}}$, ajouter des entités décrivant des **normes** ou **règles** de fonctionnement (certaines pouvant être symboliques, voir section 1.5.7), ainsi que des **entités contextuelles** (charge de travail, température ambiante, etc.). Ces entités s'**auto-organisent** et forment un **cluster** principal (ou plusieurs) lorsqu'elles partagent un comportement cohérent.

Une **anomalie** se manifeste lorsque la **synergie** $S(\mathcal{E}_k, \mathcal{E}_{\text{ref}})$ entre un capteur \mathcal{E}_k et le **cluster** de référence (normes, capteurs stables) chute. Contrairement à un système de seuils figés par capteur (par exemple "vibration > X" ou "température > Y"), le **DSL** confronte chaque capteur à l'ensemble de la structure en évolution : si un capteur dérive **seul** et ne suit plus la synergie habituelle, il se voit **isolé**, donc potentiellement en anomalie. À l'inverse, si plusieurs capteurs entament des dérives **cohérentes**, ils peuvent former un **sous-cluster** spécifique (un "cluster d'anomalie"), laissant deviner un défaut commun (ex. un roulement en fin de vie qui génère bruit et surchauffe).

Au fil du temps, un capteur fautif peut :

22. Se retrouver **isolé** dans un "espace d'anomalie" lorsque ses pondérations $\omega_{k,\cdot}$ décroissent,
23. Entrer en **coopération** avec d'autres entités également dérivantes, aboutissant à un nouveau **cluster** d'anomalie plus large,
24. Déclencher un **signal** de maintenance dès lors que la situation persiste ou si la synergie interne de l'anomalie (ex. bruit anormal + vibrations + température de roulement) continue de croître, signant une panne imminente.

Cette **auto-organisation** rend la détection plus adaptative : le **réseau** perçoit progressivement quelles **liaisons** demeurent fiables et lesquelles s'écartent. On évite ainsi les contraintes d'un pipeline rigide ou de seuils par capteur, car la logique de synergie estime la **relation** du capteur à l'ensemble du **cluster** de fonctionnement normal, autorisant un repérage de dérives **contextualisé** et évolutif.

1.6.5.4. Avantages et Défis : la Logique du DSL en Alerte et Diagnostic

Les applications de **Deep Synergy Learning (DSL)** dans le domaine de la **surveillance**, de la **maintenance** ou du **diagnostic** s'appuient sur sa capacité à constituer des **clusters** de manière autonome, en agrégant des **entités** multiples (capteurs, tests, imagerie, etc.). Contrairement aux solutions qui demandent une fusion de flux orchestrée

manuellement, le **DSL** laisse les pondérations $\omega_{i,j}$ évoluer en fonction de la **synergie** locale, formant des sous-groupes pertinents sans pipeline imposé.

Cette approche confère une **flexibilité** face aux données partielles ou intermittentes. Des capteurs sujets à des interruptions ou à des pannes peuvent se retirer du **cluster** principal dès lors que leur **synergie** avec les autres entités s'effondre, puis se réinsérer lorsque leur fiabilité s'améliore. On évite ainsi qu'un capteur défaillant ne paralyse l'ensemble.

La **détection de situations inédites** (anomalies, configurations nouvelles) bénéficie particulièrement de la logique non supervisée du **DSL**. Quand un groupe d'entités dérive au regard de la norme, il se regroupe spontanément en **cluster** d'anomalie, sans que l'on ait à pré-définir les classes d'incidents. Cela autorise un repérage de schémas insoupçonnés ou de pannes rares.

Les mécanismes d'**adaptation continue** déjà décrits (section 1.5.4) favorisent la réactivité en contexte dynamique : les pondérations ω se réajustent si l'état du patient se modifie (en milieu médical) ou si la chaîne de production passe à un régime nouveau (en milieu industriel). Ce fonctionnement n'exige pas un **réapprentissage** exhaustif à chaque changement.

Cependant, le **principal défi** réside dans la **taille** du graphe et la croissance exponentielle du nombre de liens $\{\omega_{i,j}\}$ quand les entités se multiplient (nombre élevé de patients, de capteurs, de variables, etc.). Il convient d'appliquer des mesures de **parsimonie** (un seuil en-deçà duquel on supprime des connexions), des procédures de **sparsification** dynamique, ou encore des **mises à jour** localisées (échantillonnage partiel, focus sur un sous-ensemble d'entités), pour maintenir la **scalabilité** et la réactivité du système.

Conclusion

En **surveillance, diagnostic médical** ou **détection d'anomalies**, le **Deep Synergy Learning** apporte une **vision dynamique, distribuée, et auto-organisée** pour :

- Déetecter des patterns normaux (clusters stables) vs. anormaux (entités isolées ou clusters dérivants),
- Gérer de multiples sources de données (capteurs, tests, imagerie) sans pipeline prédéfini,
- Réagir en continu aux évolutions ou aux nouvelles situations,
- Combiner éventuellement un **signal** de récompense ou de **validations** cliniques/industrielles partielles pour renforcer l'auto-organisation.

Ainsi, le DSL offre un **mécanisme** mathématique de **pondérations adaptatives** et de **synergies** exploitables dans divers secteurs où la **variabilité** des conditions et la **nécessité** de détecter des comportements hors normes sont prépondérantes.

1.6.6. Planification et Optimisation dans l'Industrie 4.0

Les secteurs industriels évoluent vers une **connectivité** accrue, où les machines, capteurs, robots, stocks et systèmes de gestion communiquent en temps réel : c'est la vision de l'**Industrie 4.0**. Les défis qui en découlent — **planification distribuée, optimisation des flux, maintenance prédictive et gestion adaptative** — peuvent se révéler extrêmement complexes. Les méthodes traditionnelles (ordonnancement déterministe, programmation linéaire classique, heuristiques fixes, etc.) montrent souvent leurs limites face à la **dynamique** croissante (marchés volatils, nouvelles commandes urgentes, pannes inattendues). Dans ce cadre, le **Deep Synergy Learning (DSL)**, avec son **mécanisme d'auto-organisation** et de **synergie adaptative**, ouvre de nouvelles pistes pour réaliser une **planification plus flexible** et des **schémas d'optimisation** capables de se réajuster en continu.

Dans cette section (1.6.6), nous examinons :

25. Les raisons pour lesquelles l'**Industrie 4.0** exige un modèle **évolutif et distribué**,
26. Comment le **DSL** modélise des entités (machines, stocks, commandes, flux de transport) reliées par des **pondérations synergiques**,
27. De quelle manière ces liens évoluent pour **planifier et optimiser** les ressources en temps réel,
28. Des exemples concrets où le **DSL** peut surpasser un pipeline d'optimisation statique, grâce à sa **dynamique d'adaptation** et à sa capacité de **clustering** auto-organisé.

1.6.6.1. Les Défis Industriels dans un Environnement 4.0

Dans l'Industrie 4.0, on fait face à une **abondance** de composants interdépendants. Les **machines** possèdent chacune un état (vitesse, disponibilité, pannes potentielles), tandis que les **chaînes de production** sont subdivisées (stations, lignes, buffers, flux de pièces). Les **stocks** (pièces détachées, semi-finis, produits finis) changent au fil des demandes, et le **transport** peut impliquer différents véhicules (AGV, drones, camions). Enfin, les **ordres de fabrication** évoluent dès lors que des clients passent de nouvelles commandes ou modifient leur planning.

Dans les approches classiques d'**ordonnancement** ou de **planification** (flow shop, job shop, heuristiques variées), on définit un **problème combinatoire** qu'on tente de résoudre sous des **hypothèses** de stabilité. En pratique, l'environnement est loin d'être statique : les machines peuvent tomber en **panne**, un retard logistique perturbe la disponibilité d'une pièce critique, de nouvelles **commandes** arrivent à l'improviste et la **capacité** machine peut varier pour cause de maintenance ou de changement d'outillage. Chaque perturbation requiert souvent un **recalcul** important, voire une réinitialisation d'un algorithme d'ordonnancement.

Le **Deep Synergy Learning (DSL)** propose un **changement de perspective**. Au lieu de modéliser un ordonnancement global, on conçoit chaque **composant** — machine, ordre, stock, mode de transport — comme une **entité** \mathcal{E}_i . Les **liaisons** $\omega_{i,j}(t)$ traduisent la **coopération** ou la **complémentarité** entre entités. Par exemple, deux machines $\mathcal{E}_{\text{machineA}}$ et $\mathcal{E}_{\text{machineB}}$ voient leur lien s'intensifier si elles se passent fréquemment des pièces sans blocage, ou si l'on constate un gain productif à enchaîner les opérations A puis B. Un **stock** $\mathcal{E}_{\text{stock}}$ et un **transport** \mathcal{E}_{AGV} coopèrent s'ils se synchronisent efficacement pour déplacer des lots. De même, un **ordre** $\mathcal{E}_{\text{ordre}}$ accroît sa synergie avec une machine spécifique si cette dernière exécute la tâche en temps et qualité optimales.

Lorsqu'un **incident** surgit (panne machine, retard de livraison, surcharge de commande), la **dynamique d'auto-organisation** du DSL réévalue les liaisons ω . Un composant en panne voit ses liens se dégrader, réduisant son impact dans le flux. Inversement, si une nouvelle machine arrive ou si un transport reprend sa disponibilité, on insère une **entité** inédite dans le **SCN**, laissant la synergie se construire là où un **gain** apparaît. Cette **reconfiguration** s'effectue **localement**, évitant de relancer un algorithme d'ordonnancement global. Le **réseau** s'ajuste de manière "vivante", reflétant la réalité mouvante de l'Industrie 4.0.

1.6.6.2. Représentation des Flux et Clusters Auto-Organisés

Dans une optique d'**Industrie 4.0**, la mesure de **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j peut prendre plusieurs formes. Les machines, les stocks ou les ordres de production entretiennent par exemple un **taux de transfert** élevé lorsque la première fournit régulièrement des pièces à la seconde sans créer de file d'attente. Un **ordre** de production se révèle quant à lui **complémentaire** d'un stock donné si cet ordre requiert précisément les ressources contenues dans ce stock. Par ailleurs, la **performance** d'une association machine-ordre s'accroît lorsque le couple aboutit à un temps de cycle réduit et à un rendement supérieur. Lorsque la **synergie** apparaît remarquable entre deux entités, la pondération reliant ces entités se trouve renforcée (voir section 1.4.5 sur la mise à jour locale des pondérations).

À mesure que certaines combinaisons s'avèrent profitables (machine-stock-ordre-transport, par exemple), leur coopération se **solidifie**, faisant émerger des **clusters** ou sous-réseaux. Chaque **cluster** correspond alors à une configuration de production considérée stable ou efficace, puisqu'il regroupe des entités dont les pondérations mutuelles ω sont élevées. On peut ainsi observer la cristallisation d'un groupe

{Machine1, StockA, TransportX, OrdreB} au motif que ces éléments forment un **flux** fluide : la machine 1 exploite les ressources du Stock A, le Transport X assure rapidement la logistique, et l'Ordre B en tire un bénéfice opérationnel.

Dans le cadre d'une planification traditionnelle, on conçoit des **lignes** ou **cellules** de production de manière analytique. Le **Deep Synergy Learning (DSL)**, lui, opère un mécanisme **auto-organisé** : les liaisons se **renforcent** quand un groupe de machines s'avère compatible ou lorsqu'un module de transport complète efficacement la chaîne, ce qui fait naître un **cluster** de manière spontanée. Quand un composant (machine, stock ou transport) **dysfonctionne** ou s'avère insuffisamment utile, la **synergie** avec le reste du réseau chute et le composant se trouve mis à l'écart. De nouveaux liens peuvent alors se former vers des entités de **remplacement**, réorientant la production de façon plus souple qu'un pipeline rigide. Ce phénomène procure à l'ensemble une **plasticité** qui évoque un système "vivant", réactif aux perturbations et aux changements d'outillage ou de répartition des tâches, conformément à l'esprit de l'Industrie 4.0 décrite en section 1.6.6.1.

1.6.6.3. Approche Dynamique : Adaptation en Temps Réel

Dans un environnement **Industrie 4.0**, le système doit composer avec un **flux** permanent d'événements : nouvelles commandes (urgentes ou non), alertes de stocks bas, maintenance planifiée sur une machine, retards logistiques. Les algorithmes de planification classiques (règle de Johnson, heuristiques de scheduling, etc.) se montrent performants pour un **problème** donné (flow shop, job shop), mais perdent en **flexibilité** lorsqu'on modifie le problème à la volée, car il faut alors réinitialiser ou réactualiser le schéma de planification.

Au **Deep Synergy Learning (DSL)**, la **dynamique** d'auto-organisation se renouvelle en continu. Les **pondérations** $\omega_{i,j}(t)$ reliant des entités (machines, ordres, stocks, transports...) se remanient itération après itération en fonction de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Lorsqu'un **ordre urgent** survient, le système introduit une entité $\mathcal{E}_{\text{ordreUrgent}}$ dans le **Synergistic Connection Network**, et celle-ci voit ses **liaisons** $\omega_{\text{ordreUrgent}, k}$ se renforcer rapidement dès que l'on identifie un gain potentiel avec des machines ou des transporteurs capables de le traiter à temps. Si, au contraire, un transport $\mathcal{E}_{\text{transport}}$ se montre sous-dimensionné (accumulation de pièces, délais excessifs), sa **synergie** avec le reste du réseau chute, ce qui oriente la production ou la logistique vers d'autres entités plus aptes à satisfaire la demande.

Le **DSL** ne vise pas à figer une **solution** optimale à un instant t , comme le ferait un algorithme de scheduling : il **maintient** plutôt un **réseau** dont la topologie (clusters de machines, ordres, stocks) évolue au fil des événements. Si l'on dispose d'un **signal** global de performance (par exemple, un throughput, un temps de cycle moyen, un taux de satisfaction client), on peut l'incorporer pour ajuster la **synergie** : les entités participant à une configuration jugée efficiente voient leurs liaisons ω se consolider, celles menant à un échec ou à une moindre performance s'affaiblissent. Le système s'adapte donc en continu, s'approchant d'une forme d'**auto-organisation** où chaque entité veille à coopérer localement, sans reposer sur une refonte complète de la planification à chaque nouvel incident.

1.6.6.4. Cas d'Exemple : Usine Modulable

Dans une **usine** où les postes sont **reconfigurables** et la logistique assurée par des **AGV** (véhicules autonomes), les produits fabriqués peuvent changer selon la **demande**. Le **Deep Synergy Learning (DSL)** s'avère alors bénéfique pour une **gestion** plus flexible, en remplaçant ou en complétant les méthodes de planification traditionnelles. On peut définir un **réseau** d'entités $\{\mathcal{E}_{\text{Poste}1}, \mathcal{E}_{\text{Poste}2}, \dots, \mathcal{E}_{\text{AGV}1}, \mathcal{E}_{\text{AGV}2}, \dots, \mathcal{E}_{\text{Stock}1}, \dots\}$ et des entités "ProduitA", "ProduitB" (avec la possibilité d'en ajouter de nouveaux). Chaque poste ou AGV entretient des **liaisons** $\omega_{i,j}$ qui se **renforcent** ou se **délient** selon la **synergie** constatée. Si un **poste** particulier traite un certain **produit** avec un taux de réussite élevé, la pondération $\omega_{\text{Poste}, \text{Produit}}$ augmente. De même, un **AGV** se connecte aux postes ou stocks avec lesquels il collabore efficacement, formant ainsi un **cluster** {AGV1, Poste1, Poste2, Stock} si ce groupe de composants assure un flux fluide de pièces.

Lorsqu'on introduit un **nouveau** produit (par exemple "ProduitC"), la synergie se construit **naturellement** autour des postes adéquats. Le **SCN** évolue sans qu'il soit indispensable de reconfigurer entièrement le pipeline ou de lancer un **solveur** d'ordonnancement intégral. Les entités reliées à l'ancien produit peuvent **conserver** leurs liens, tandis que les postes ou AGV inadaptés à "ProduitC" voient leur pondération chuter et se déconnectent graduellement du nouveau

cluster. Cette **dynamique** assure une **réactivité** notable : si une **panne** survient sur Poste1, la pondération $\omega_{\text{AGV}, \text{Poste}1}$ décroît et l'AGV peut se rediriger vers un autre poste ($\mathcal{E}_{\text{Poste}3}$), dont la synergie montera en prenant le relais.

Ce **mécanisme d'auto-organisation** continue à favoriser la **tolérance** aux pannes et l'adaptation à l'introduction de nouvelles gammes de produits. Les entités inopérantes se "détachent" naturellement (perdent leurs liaisons), et les nouvelles entités intéressantes (nouveaux postes, nouveaux produits) trouvent leur place dans le **Synergistic Connection Network** au fil de la **dynamique** des pondérations. Cela évite la nécessité de **recalculer** périodiquement un ordonnancement global ou de procéder à des heuristiques lourdes, tout en exploitant les **principes** du DSL (sections 1.4.3 et 1.5.4) pour maintenir la cohérence globale du système.

1.6.6.5. Limites et Évolutions : Complexité, Parcimonie, Hybridation

Dans un **environnement industriel** où l'on introduit sans cesse de nouvelles entités (machines, stocks, ordres...), la **taille** du graphe $\{\omega_{i,j}\}$ peut rapidement devenir prohibitif. Pour préserver la faisabilité, il est essentiel d'envisager des stratégies de **parsimonie** (seuils, coupe des liens trop faibles) ou des **algorithmes locaux** où seules les pondérations pertinentes se mettent à jour, évitant ainsi un recalcul exhaustif à chaque itération. Des **contraintes** peuvent également exclure certains liens d'emblée (une machine non qualifiée pour un ordre, ou un transport incapable de prendre en charge un certain poids), imposant une **pénalisation** ou un **masquage** dans la fonction de synergie.

De plus, lorsque des **règles logiques** (contraintes de sécurité, de qualité) s'ajoutent, on peut intégrer des **entités symboliques** (cf. section 1.5.7). Le **DSL** considère alors la **coopération** entre ces entités symboliques et les entités opérationnelles (machines, ordres...), de sorte que la formation d'un **cluster** illégal (par exemple, un regroupement violant une norme de sécurité) n'aboutisse jamais, faute de synergie. Cette hybridation **neuro-symbolique** enrichit la dynamique locale (ω) tout en respectant les cadres réglementaires ou métier.

Il est par ailleurs envisageable de **combiner** le **DSL** à des heuristiques ou solveurs d'ordonnancement établis. Une possibilité consiste à laisser l'**auto-organisation** déterminer un sous-ensemble de ressources potentiellement efficaces, puis à faire appel à un **solveur** classique pour finaliser le planning sur ce sous-ensemble, réduisant la dimension du problème. À l'inverse, un solveur peut fournir une solution initiale, que le **DSL** ajuste localement au gré des aléas (pannes, nouvelles commandes). Cette **collaboration** exploite la force de l'**auto-organisation** (réactivité, plasticité) tout en bénéficiant de méthodes d'**optimisation** éprouvées pour parfaire le résultat. Ainsi, la **plasticité** du DSL et la **rigueur** de l'optimisation se conjuguent pour répondre aux exigences variées de l'Industrie 4.0.

Conclusion

Dans le contexte de **planification** et d'**optimisation** pour l'**Industrie 4.0**, le **Deep Synergy Learning** introduit :

- Une **approche distribuée** : chaque ressource (machine, transport, stock) agit comme une entité évolutive, plutôt qu'un simple paramètre dans un solveur central,
- Une **mise à jour adaptative** : plus besoin de résoudre un problème complet à chaque événement, on ajuste localement les synergies pour redistribuer les flux,
- Une **auto-organisation** de clusters (sous-réseaux de production) susceptibles de se reformer si des pannes ou des variations de demandes surviennent,
- Une **tolerance** aux données manquantes ou bruitées (capteur incertain, etc.),
- Une **extensibilité** : on peut ajouter de nouvelles machines ou de nouvelles gammes de produits, et laisser le DSL découvrir où les insérer via la synergie.

Bien sûr, des défis (taille du problème, besoin de parcimonie, contraintes symboliques) subsistent, mais la **philosophie** du DSL — laisser la dynamique ω déterminer en continu les regroupements et la répartition des tâches — se marie avec l'exigence de **flexibilité** et d'**évolution** rapide propres à l'Industrie 4.0. Ainsi, le DSL peut constituer un **cadre**

pour des solutions d'**ordonnancement** et d'**optimisation** plus “organiquement adaptatives” que les méthodes statiques d'hier, ouvrant la voie à une production réellement **agile** et **intelligente**.

1.6.7. Perspectives pour la Recherche Fondamentale en IA Forte

Les précédentes sous-sections (1.6.1 à 1.6.6) ont démontré la **portée pratique** du Deep Synergy Learning (DSL) dans divers domaines applicatifs (vision, audio, robotique, recommandation, diagnostic, etc.). Cependant, l'**ambition** du DSL ne se limite pas à une simple amélioration des performances ou de la flexibilité dans ces tâches spécialisées. À un niveau plus **fondamental**, l'approche synergique **ouvre des pistes** de recherche susceptibles de rapprocher l'IA de la notion d'**IA Forte** (ou **IA Généralisée**), c'est-à-dire d'une intelligence capable d'**apprendre** et de **raisonner** de manière autonome et générale, au-delà de cadres strictement définis. Cette section (1.6.7) se propose d'explorer ces **perspectives théoriques et conceptuelles** :

29. L'idée que le DSL, grâce à ses **mécanismes d'auto-organisation**, évoque un fonctionnement plus proche de **systèmes cognitifs** (cerveau, écologies d'informations),
30. La possibilité d'**intégrer** simultanément des dimensions sub-symboliques (apprentissage sur données massives) et des **aspects symboliques** ou logiques (voir 1.5.7),
31. Les **propriétés émergentes** (représentations, micro-réseaux cognitifs, auto-adaptation) qui dépassent la simple exécution d'une tâche,
32. Les **défis mathématiques** et philosophiques associés à la poursuite d'une IA Forte via le DSL.

1.6.7.1. Au-delà de l'Apprentissage Supervisé : Vers l'Auto-Construction de la Connaissance

Une large part des avancées récentes en **IA** se fonde sur l'**optimisation globale** de réseaux neuronaux massifs par **rétrôpropagation**, qu'il s'agisse de données annotées en abondance ou d'un pré-entraînement auto-supervisé (approche de type GPT, BERT). Bien qu'efficaces, ces modèles conservent plusieurs caractéristiques :

Ils s'appuient sur une **architecture** linéaire ou faiblement hiérarchisée (même si l'on observe des multi-têtes d'attention).

Ils dépendent d'une **descente de gradient** ou d'une procédure équivalente visant à minimiser un coût défini globalement.

Ils se révèlent peu **adaptables** localement : lorsque la distribution se modifie, on doit souvent procéder à un réapprentissage (fine-tuning), risquant un écrasement partiel des acquis (catastrophic forgetting).

Le **Deep Synergy Learning (DSL)** présente une **dynamique d'auto-organisation** plus proche de l'esprit de **systèmes complexes** (biologiques ou écologiques), où chaque **entité** (sous-module, neurone, flux d'information) gère ses **liaisons** suivant la **synergie** qu'il perçoit avec d'autres entités. Plutôt que de minimiser une **unique fonction de coût**, on autorise l'émergence de **clusters** et la **reconfiguration permanente** des connexions $\omega_{i,j}$. Cette approche “**bottom-up**” s'avère potentiellement féconde pour développer des **mécanismes cognitifs** plus avancés et moins tributaires d'un lourd apprentissage supervisé.

Dans une perspective d'**IA Forte**, on envisage souvent un agent qui **accumule** graduellement des connaissances sans les oublier, tout en **réinterprétant** ces savoirs au fil des expériences. Le **DSL** y répond en laissant chaque **entité** conserver sa mémoire locale (paramètres, historique) et en **réévaluant** les liaisons ω de façon continue. On obtient ainsi un processus **incrémental** :

De nouvelles **entités** (concepts, données) apparaissent ;

Des **liaisons** se **créent** ou se rompent suivant la pertinence ou la coopération mesurée ;

Des **clusters** de connaissances se **stabilisent**, ou se scindent si leur synergie interne décline ;

L'agent peut alors “**auto-structurer**” ses connaissances, analogue à un cerveau formant et dissolvant des **assemblées neuronales**.

Cette organisation, plus **distribuée** et **évolutive**, pose les bases d'une **auto-construction** de la connaissance, où l'on n'est plus tributaire d'un unique label ou d'une unique fonction de perte, mais d'une dynamique **locale** entre les **entités** qui renforcent ou amoindrissent leurs liens au gré de leur coopérativité. Elle suggère la possibilité d'un **savoir** plus flexible et cumulatif, réduisant la dépendance à un entraînement exhaustif et favorisant une **adaptation** continue aux données nouvelles ou aux changements de contexte.

1.6.7.2. Vers une Cognition Distribuée et Émergente

Le **Deep Synergy Learning (DSL)** trouve ses racines (section 1.3.1) dans l'observation des **systèmes biologiques** exhibant de la **plasticité** (synaptique chez les neurones), ou des phénomènes d'**auto-organisation** (colonies d'insectes, tissus cellulaires). Dans la quête d'une **IA Forte**, on peut s'appuyer sur un **réseau** d'entités en tant que **substrat cognitif**, où :

Les **entités** \mathcal{E}_i incarnent des briques d'information variées : qu'il s'agisse de *features* perceptuelles (captées par des flux sensoriels), de **concepts abstraits** (catégories, notions sémantiques) ou même de **modules symboliques** (ensembles de règles logiques, sous-réseaux dédiés).

Les **liaisons synergiques** $\omega_{i,j}$ instaurent un **renforcement** de type “hebbien généralisé” dès lors qu'une **coopération** entre entités \mathcal{E}_i et \mathcal{E}_j procure un **gain** (qu'il s'agisse de performance, de co-information ou de simplification).

Les **clusters** qui émergent (assemblées ou circuits cognitifs) assument des rôles spécifiques dans le traitement conceptuel, la mémoire de travail, la planification ou encore l'intégration de signaux multiples.

Au sein de ce **substrat**, chaque **cluster** reflète potentiellement une micro-théorie ou un schéma conceptuel, regroupant des entités pour former une **hypothèse**, un **concept**, voire un **plan d'action**. La structuration se fait de façon **dynamique** : les assemblées apparaissent ou se scindent selon l'évolution de leur synergie, permettant de reconfigurer en continu l'inventaire des connaissances ou des idées. Contrairement aux architectures statiques (couches fixes), cette configuration s'**auto-régule**, entre **plasticité** (création et rupture de liens) et **stabilité** (clusters pertinents).

En focalisant l'analyse sur la **dynamique** globale, on peut reproduire certains **processus** cognitifs : une **forme d'attention** résulte si un sous-réseau monopolise la synergie disponible, comparable à un focus transitoire. Des **boucles** de rétroaction positives ou négatives peuvent générer des **oscillations**, analogues à des rythmes cognitifs ou des alternances d'états mentaux. La **conservation** d'équilibres permet au système de préserver des organisations robustes, tout en se montrant apte à accueillir de nouvelles liaisons ou à rompre les moins pertinentes.

De tels phénomènes, absents d'un CNN ou d'un Transformer (qui restent malgré tout dans une structure de pipeline, bien que complexe), s'apparentent plutôt à un **processus** cognitif **distribué**, dans lequel l'information chemine entre assemblées modulaires qui se constituent et se dissolvent librement. On entrevoit là un **jalon** vers des agents en mesure de **construire** et de **réviser** en continu leurs hypothèses ou “théories”, de manière plus proche de la **biologie** et des **neurosciences** que de la minimisation stricte d'une fonction de coût globale.

1.6.7.3. Intégration Symbolique : Socle pour le Raisonnement Abstrait

Dans une démarche d'**IA Forte**, il ne suffit pas de traiter des signaux sub-symboliques (capteurs, images, séquences, etc.) : il faut également pouvoir manier des **représentations abstraites** et des règles logiques complexes. Les réseaux neuronaux profonds, malgré leur puissance pour l'apprentissage sur données massives, peinent souvent à exécuter un raisonnement explicitement symbolique, à moins d'y adjoindre des modules spécialisés. Le **Deep Synergy Learning (DSL)** propose un **cadre unifié** où des entités clairement “symboliques” (comme \mathcal{E}_{rule} , $\mathcal{E}_{concept}$) coexistent avec des entités sub-symboliques (\mathcal{E}_{sensor} , $\mathcal{E}_{feature}$), comme décrit à la section 1.5.7. Les liaisons $\omega_{rule,sensor}$ s'enrichissent dès lors qu'une **règle** s'applique à un **schéma** concret observé : la synergie se renforce si la correspondance entre la règle et la configuration sensorielle se révèle fructueuse (information mutuelle, gain de performance, etc.). On obtient alors

un **micro-réseau** dans lequel un ensemble de règles se coordonne avec un ensemble d'observations, sans cloison rigide séparant le module logique du module perceptif.

Dans cette perspective, la **dynamique** des pondérations ω ne se contente plus de rapprocher ou de séparer des entités “vision” ou “audio” : elle peut tout autant consolider des liens entre un **concept abstrait** et divers indices sensoriels confirmant sa validité, ou inversement discréder une règle en la séparant d'un environnement où elle ne s'applique pas. Cela ouvre la voie à un **raisonnement** logico-perceptif plus fluide : lorsqu'une hypothèse symbolique \mathcal{E}_{rule} se voit soutenue par l'observation \mathcal{E}_{sensor} , leur liaison grandit et un **cluster** naît, fusionnant la dimension symbolique et la dimension sub-symbolique. Cette approche épargne la nécessité d'implanter un connecteur artificiel entre un réseau neuronal et un moteur logique indépendant : c'est le **réseau** lui-même, via la mise à jour des synergies, qui unifie ces deux registres. On y gagne en transparence et en **plasticité** : les règles symboliques peuvent être admises, ajustées ou rejetées au gré de la compatibilité avec les données sensorielles, se reformant ou se dissolvant au sein du **SCN**. Il en découle un possible **socle** pour un raisonnement abstrait mieux ancré dans la perception, condition importante de l'**IA Forte**, qui réclame un agent sachant à la fois extraire des modèles du monde et manipuler des constructions logiques plus élevées.

1.6.7.4. Défis Mathématiques : un Pas vers une Théorie de la Cognition Synergique

Le **Deep Synergy Learning (DSL)** peut se voir comme un **système dynamique** non linéaire de grande dimension, dont la **matrice** $\Omega(t)$ (ou un hyper-graphe) évolue suivant une loi similaire à

$$\Omega(t+1) = \Omega(t) + \eta [S(\Omega(t)) - \tau \Omega(t)],$$

où $\Omega(t)$ désigne l'ensemble des pondérations $\{\omega_{i,j}(t)\}$ à l'instant t , et $S(\Omega(t))$ symbolise la mesure de **synergie** calculée entre les entités, qui peut elle-même dépendre de l'état global. Cette dynamique est susceptible de générer des **attracteurs** multiples (plusieurs organisations cognitives possibles), des **bifurcations** lorsque les synergies changent brusquement, voire des **cycles limites** interprétables comme des rythmes cognitifs ou des alternances d'états mentaux. L'**analyse formelle** (existence et stabilité d'attracteurs, transitions critiques, phénomènes d'hystérèse) reste toutefois un **défi** majeur, notamment si l'on souhaite déployer de nombreuses entités ou autoriser une définition non triviale de la synergie (information mutuelle n-aire, PID, etc.).

Pour gérer la croissance potentiellement explosive du nombre de liaisons $\{\omega_{i,j}\}$ et leur mise à jour en temps réel, on se tourne fréquemment vers des **méthodes de parcimonie** (seuils, coupes, hiérarchies de clusters) ou des **stratégies** d'échantillonnage. Certains mécanismes inspirés de la **biologie** — par exemple un terme d'oubli rapide ou une saturation — contribuent également à la **scalabilité**. Le choix précis de la fonction de synergie S , de la règle de mise à jour et de la topologie initiale influe sur la façon dont le réseau s'organise.

En parallèle, les travaux en **sciences cognitives** ou neurosciences computationnelles (comme la **théorie de l'information intégrée** de Tononi) proposent de quantifier la **part** d'information réellement unifiée dans un système. Dans l'esprit du **DSL**, on peut définir une **intégration globale** à partir de la **somme** des synergies (ou d'une co-information n-aire) :

$$\text{Intégration globale} = \sum_{i,j} \omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j),$$

ou encore effectuer une version plus générale au sein d'un **cluster** n-aire. Cette démarche veut évaluer la “richesse” émergente dans un réseau en perpétuelle auto-organisation. Elle ouvre un **pas** vers des réflexions sur la **conscience artificielle** ou l'**autonomie** inspirée de la cognition humaine, puisqu'elle quantifie à quel degré le réseau regroupe et intègre l'information sous forme de micro-réseaux cognitifs, susceptibles d'exhiber des propriétés d'**émergence** et de **complexité** caractéristiques des systèmes vivants.

Conclusion

Les perspectives du **Deep Synergy Learning** pour la **recherche fondamentale** en **IA Forte** s'articulent autour de :

- Sa **capacité** à s'**auto-structurer** hors d'un pipeline rigide ou d'une unique fonction de coût globale,
- Son **paradigme** distribué et émergent, plus proche de certains **modèles biologiques** ou cognitifs,
- L'**intégration** simultanée de dimensions sub-symboliques (features, embeddings) et symboliques (règles, concepts logiques) dans un **même** réseau de synergie,
- L'éventuelle **quantification** de la **richesse** et du **niveau d'intégration** du réseau, ouvrant la voie à des notions plus avancées (auto-réflexion, conscience, adaptativité illimitée).

En somme, si le **DSL** se montre déjà **efficace** et **agile** dans des applications concrètes (sections 1.6.1 à 1.6.6), il pourrait aussi servir de **fondement** à une IA plus **globale**, capable d'apprendre, de se réorganiser et d'**intégrer** différents registres de représentations (perception, logique, symbolique) de manière plus **organique**. La réalisation d'une **IA Forte** n'est pas garantie, mais le **DSL** suggère une nouvelle **philosophie** : remplacer la hiérarchie imposée par une **auto-organisation** continue, potentiellement apte à donner naissance à des **capacités cognitives** plus vastes et plus proches du fonctionnement du cerveau ou de systèmes biologiques.

1.7. Défis, Contraintes et Ouvertures

Les sections précédentes (1.1 à 1.6) ont mis en lumière la **logique fondamentale du Deep Synergy Learning (DSL)**, ses **mécanismes** d'auto-organisation et ses **capacités** à apporter des solutions nouvelles dans de multiples domaines applicatifs (vision, audio, robotique, recommandation, diagnostic, etc.). Toutefois, comme toute approche novatrice, le DSL n'échappe pas à des **défis** et des **contraintes** majeurs. Sur le plan **computationnel, théorique, méthodologique** et **éthique**, de nombreuses questions demeurent ouvertes, exigeant une recherche soutenue pour perfectionner le paradigme et garantir son insertion fluide dans des environnements industriels, médicaux, ou encore scientifiques.

La volonté du DSL de fonctionner sans architecture hiérarchique figée, et de laisser les entités s'**auto-organiser** via des synergies, requiert notamment :

- Une **scalabilité** adéquate (comment passer à des milliers ou millions d'entités sans explosion de la complexité ?),
- Une **qualité** et une **accessibilité** suffisantes des données (pour estimer la synergie de façon robuste),
- Des **algorithmes d'optimisation** aptes à gérer des systèmes dynamiques, non linéaires et potentiellement hétérogènes,
- Un **contrôle** de la stabilité et de la convergence (éviter les oscillations ou l'effondrement du réseau),
- Une **interprétabilité** pour l'humain, d'autant plus cruciale lorsque les décisions impactent des vies ou des processus critiques,
- Des **considérations éthiques** et réglementaires pour encadrer l'usage de systèmes auto-organisés, notamment en termes de responsabilité, d'acceptabilité, ou de gestion de biais,
- Des **comparaisons expérimentales** rigoureuses avec les approches existantes (réseaux neuronaux profonds, méthodes symboliques, algorithmes d'optimisation classiques) pour valider la pertinence et les conditions d'efficacité du DSL.

Dans cette section (1.7), nous examinons un à un ces **défis, contraintes et ouvertures**, déclinés en sept points :

Complexité Computationnelle et Scalabilité (1.7.1)

Qualité et Disponibilité des Données (1.7.2)

Développement d'Algorithmes d'Optimisation Appropriés (1.7.3)

Contrôle et Stabilité des Processus Auto-Organisés (1.7.4)

Interprétabilité et Explicabilité pour l'Humain (1.7.5)

Considérations Éthiques et Réglementaires (1.7.6)

Comparaisons Expérimentales avec d'Autres Approches (1.7.7)

Ce panorama permettra de saisir la **complexité** inhérente à l'approche synergique, tout en mettant en relief les **axes** de recherche et les **collaborations** scientifiques qu'elle appelle, pour faire du DSL un cadre de plus en plus solide dans l'écosystème de l'IA et de l'apprentissage automatique.

1.7.1. Complexité Computationnelle et Scalabilité

La première critique ou préoccupation qu'on peut formuler à l'égard du **DSL** concerne la **taille** potentielle du **graphe** des entités et la **complexité** des opérations nécessaires pour **calculer** ou **mettre à jour** les **pondérations** synergiques ($\omega_{i,j}$). En effet, si on postule :

- n entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$,
- une synergie binaire $S(\mathcal{E}_i, \mathcal{E}_j)$ pour chaque paire (i, j) ,
- une mise à jour régulière (à chaque itération ou chaque batch temporel),

alors la **complexité brute** peut facilement atteindre $O(n^2)$ par itération, ce qui devient ingérable quand n est de l'ordre de plusieurs milliers, voire millions, de composantes. De plus, si l'on veut aller vers des **synergies n-aires** (pour capturer des coopérations à plusieurs entités simultanément), la situation s'aggrave encore (complexité exponentielle).

1.7.1.1. Problématique du “Tout Relier à Tout”

L'un des principes centraux du **Deep Synergy Learning (DSL)** est de laisser les **entités** $\{\mathcal{E}_i\}$ se connecter ou se déconnecter au fil du temps, suivant la **synergie** perçue. Pourtant, permettre qu'une entité puisse se relier à toutes les autres sans limite conduit vite à un **graphe complet**, où chaque paire (i, j) exige un calcul de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$. Cette démarche devient rapidement ingérable :

Le calcul de $S(\mathcal{E}_i, \mathcal{E}_j)$ pour toutes les paires (i, j) croît en $O(n^2)$.

La mise à jour des pondérations $\omega_{i,j}$ (section 1.4.5) présente la même complexité en $O(n^2)$.

Le stockage de l'ensemble $\{\omega_{i,j}\}$ requiert une **mémoire** en $O(n^2)$.

Dans des applications **massives** (robotique multi-capteurs, systèmes de recommandation avec des millions d'utilisateurs et de contenus, vision distribuée avec d'innombrables descripteurs), la construction d'un tel graphe complet compromet la **scalabilité** du DSL. Il devient crucial de maintenir une certaine **parsimonie** au sein du réseau, de sorte à ne pas créer aveuglément des liaisons $\omega_{i,j}$ pour chaque paire.

Une **stratégie** commune consiste à **restreindre** le calcul de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ aux seules paires dont les entités se révèlent “assez proches” ou déjà **corrélées**. On peut, par exemple, refuser la création de liens si la distance entre \mathbf{x}_i et \mathbf{x}_j dépasse un certain **seuil** ϵ . On peut aussi confiner le calcul à un **voisinage** k-plus-proches-voisins, limitant la croissance de connexions. Par ailleurs, il faut un **mécanisme** pour découvrir graduellement de nouvelles liaisons, via un **échantillonnage** ou une **partition** initiale, sans explorer d'emblée toutes les paires (i, j) .

En parallèle, on définit des **règles** pour **supprimer** ou **inactiver** automatiquement les liens jugés trop faibles ou non pertinents (auto-suppression sous un seuil ω_{\min}). De cette manière, la densité du graphe reste modérée. On vise alors une complexité proportionnelle au nombre de liaisons “actives”, ce qui peut être bien inférieur à n^2 . Au bout du compte, le **réseau** se rapproche d'un **k-NN** ou d'un **ϵ -radius** dynamique, sur lequel on applique les mises à jour (section 1.4.5). Le **DSL** y opère donc ses principes d'auto-organisation et de synergie, tout en évitant l'écueil d'un graphe complet ingérable.

1.7.1.2. Calcul de Synergie : Coût des Mesures Informationnelles

Lorsque la **synergie** prend la forme d'une **co-information** ou d'une **information mutuelle** entre entités modélisées comme variables aléatoires, on se heurte à la difficulté du calcul ou de l'estimation de grande dimension. Les estimateurs non paramétriques (k-NN, Kernel density) sont souvent coûteux et la quantité de données exigée peut augmenter de manière exponentielle avec la dimension (curse of dimensionality\text{curse of dimensionality}). Pour rendre ces métriques d'information pratiquement utilisables, on peut :

Limiter la dimension ou extraire des **features** plus compactes (par un autoencodeur, par une PCA, etc.), afin de réduire le champ d'action des estimateurs d'entropie et de co-information.

Recourir à des mesures de synergie plus **légères** dans de nombreux cas (par exemple, la distance euclidienne ou la similarité cosinus), quitte à se priver de la richesse de l'information mutuelle stricte.

Employer des heuristiques d'**approximation** lorsqu'on tient absolument à l'information théorique, comme un **downsampling** des données, une **approximation** paramétrique (Gaussienne, mixture Gaussienne), ou la construction d'estimateurs allégés.

Ces méthodes aident à concilier la **vision** d'un DSL basé sur la co-information et la **réalité** des applications de grande dimension, pour lesquelles un calcul exact d'entropie resterait prohibitif.

1.7.1.3. Mise à Jour Itérative ou Partielle

Même si l'on parvient à maintenir une **structure parcimonieuse** (où un sous-ensemble restreint de liaisons $\omega_{i,j}$ demeure actif), la **mise à jour** de chaque pondération peut néanmoins devenir un **processus lourd** dans un réseau de grande taille. Pour rendre le **Deep Synergy Learning (DSL)** praticable à grande échelle, il est donc nécessaire de recourir à des **méthodes itératives** ou à des **mises à jour partielles**, évitant un recalcul exhaustif à chaque itération.

Une solution envisageable réside dans une **mise à jour locale en parallèle**. Chaque entité \mathcal{E}_i ne s'occupe que de ses **voisins** actuels, c'est-à-dire les noeuds auxquels elle est reliée par des liaisons $\omega_{i,j}$ non négligeables. Chacune calcule et met à jour les pondérations concernant ses connexions, en parallèle avec les autres entités. Cette approche sollicite la **communication distribuée** (ou la mutualisation des ressources, GPU ou TPU), tout en restant localisée : on n'explore pas toutes les paires (i,j) du réseau, mais seulement les liens existants.

Il demeure toutefois essentiel d'introduire un **mécanisme** pour éviter les conflits et saturations éventuelles. Par exemple, si plusieurs entités cherchent à coopérer en excès avec la même cible, on peut concevoir une **inhibition compétitive** assurant que les liaisons se stabilisent autour d'un certain nombre de voisins. Certaines idées issues des **systèmes multi-agents** (communication locale asynchrone, règles de stabilisation) favorisent la **convergence** du réseau. Ce paradigme s'apparente finalement à une **simulation** de réseaux biologiques ou neuronaux, où les neurones mettent à jour leurs connexions en temps réel de manière distribuée. La **scalabilité** demeure possible tant que la **densité** du graphe demeure limitée et que les calculs de synergie (ou de similarité) ne concernent pas l'ensemble des paires dans un espace de grande dimension.

1.7.1.4. Vers une Mathématique de la Parcimonie et du Grouping Évolutif

Une autre voie de recherche consiste à définir une **énergie** ou un **coût** global $\mathcal{J}(\Omega)$ pour le réseau Ω , englobant non seulement le terme $-\sum \omega_{i,j} S_{i,j}$ (valorisant les liens qui affichent une forte synergie) mais également un terme de **pénalisation** du nombre de liaisons non nulles. On peut par exemple écrire :

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S_{i,j} + \alpha \|\Omega\|_0,$$

où $\|\Omega\|_0$ compte le nombre de connexions actives (i.e. $\omega_{i,j} > 0$) et $\alpha > 0$ contrôle l'importance de la parcimonie. Cette formulation **force** le réseau à **rester** relativement épars, tout en **maximisant** la somme de synergies. L'analyse mathématique des **minima locaux** de $\mathcal{J}(\Omega)$, ainsi que la description des **trajectoires** de descente (en temps continu ou discret) dans un espace de grande dimension, pose d'importants **défis** : on souhaite caractériser la **stabilité** des clusters émergents et la **croissance** potentielle de la taille du réseau. Cette approche ouvre des perspectives pour un contrôle plus strict de la **complexité**, en assurant qu'une connectivité limitée ne se transforme pas en un graphe complet, préservant ainsi la faisabilité du **DSL** à grande échelle.

Conclusion

La **complexité computationnelle** et la **scalabilité** constituent sans doute les **premiers** grands défis pour le **Deep Synergy Learning**. Sans mesures de parcimonie et sans heuristiques de mise à jour partielle, le coût en temps et en mémoire peut devenir prohibitif. Les **pistes** de solutions incluent :

- **Sparsité** intégrée dans la création/rupture de liens,
- **Méthodes** d'estimation approximative des synergies dans des espaces de haute dimension,
- **Parallélisme** et algorithmes distribués,
- **Formulations** d'énergie globale favorisant la parcimonie.

Ces recherches sont cruciales pour permettre au DSL de s'étendre à grande échelle (milliers, millions d'entités) sans perdre l'efficacité ou la réactivité nécessaires dans les applications industrielles, médicales, ou cognitives.

1.7.2. Qualité et Disponibilité des Données

Au-delà des défis de **complexité** et de **scalabilité** (1.7.1), le **Deep Synergy Learning (DSL)** doit aussi relever des enjeux majeurs quant à la **qualité** et la **disponibilité** des données. En effet, la logique synergique et auto-organisée repose sur l'évaluation locale de **mesures de synergie** ($S(\mathcal{E}_i, \mathcal{E}_j)$) et sur la **mise à jour** (positive ou négative) de pondérations $\omega_{i,j}$. Lorsque les données sont trop **bruitées**, trop **parcellaires** ou manquent de **représentativité**, la convergence vers des clusters pertinents ou la stabilité de l'architecture peut être compromise. Le DSL, tout en étant plus tolérant aux lacunes qu'un réseau neuronal figé, n'en demeure pas moins dépendant d'un certain **niveau** de fiabilité et de diversité des entrées pour exploiter la synergie.

Dans cette sous-section (1.7.2), nous analyserons :

33. Comment la **qualité** et la **couverture** des données impactent le calcul de la synergie,
34. Pourquoi la **disponibilité** (flux continu vs. données rares) peut influer sur la dynamique du DSL,
35. Les biais possibles et les problèmes d'**incohérence** qui peuvent fausser l'auto-organisation,
36. Les pistes pour **sécuriser** ou **améliorer** la robustesse de l'apprentissage synergique face à des données imparfaites.

1.7.2.1. Importance d'une Bonne Couverture et Diversité des Données

Le **Deep Synergy Learning (DSL)** s'appuie sur la fonction de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ pour guider la mise à jour des pondérations $\omega_{i,j}(t)$. Lorsque ces données \mathcal{E}_i et \mathcal{E}_j sont rares, bruitées ou biaisées, l'estimation de leur **distance** ou **similarité** (ou même de leur **co-information**) risque d'être inexacte. Cela peut entraîner un renforcement abusif de liens qui ne recouvrent aucune complémentarité réelle ou, à l'inverse, un affaiblissement injustifié de liaisons potentiellement utiles. Le réseau peut dès lors former des **clusters** peu pertinents, traduire des **bias** préexistants dans les données ou négliger des sous-groupes minoritaires.

Au-delà du simple problème d'**apprentissage** unitaire, le **DSL** met en relief la **couverture** temporelle. Si les conditions (saisons, configurations) varient, un système cherchant à s'adapter en continu aura besoin d'exemples suffisants pour chaque **régime** de données (périodes spécifiques, sous-populations) afin de réévaluer les synergies. Si l'on ne présente jamais de situations "hiver" lors de la constitution initiale du réseau, il se peut que les liaisons nécessaires ne s'instaurent pas, et le DSL échouera à former des **clusters** adaptés dès l'apparition d'un climat hivernal. La même logique vaut pour d'autres domaines : un **flux** d'information continu requiert un nombre d'**itérations** ou de répétitions suffisant pour que chaque nouvelle modalité ou contexte trouve sa place dans le **Synergistic Connection Network**.

La diversité et la **représentativité** des données apparaissent donc cruciales pour qu'un **DSL** puisse véritablement tirer parti de son **auto-organisation**. Sans cette diversité, il risque de manifester des **bias** similaires à ceux de nombre de méthodes d'apprentissage, la différence étant qu'ici c'est la **formation** des clusters (et la dynamique des liaisons ω) qui pâtit des lacunes dans la couverture des phénomènes et des populations.

1.7.2.2. Bruit, Lacunes et Données Incomplètes

Comme discuté en section 1.5.3, le **Deep Synergy Learning (DSL)** fait preuve d'une **flexibilité** remarquable vis-à-vis des données incomplètes ou bruitées. Une entité trop incertaine finit par se **retrouver isolée** si sa synergie avec les autres entités demeure inexiste, préservant ainsi la **structure** principale du réseau. Toutefois, lorsque la majeure partie des entités se révèle **entachée** d'un bruit élevé ou d'un manque important de mesures, l'**auto-organisation** échoue à faire ressortir des synergies réelles : la plupart des pondérations $\omega_{i,j}$ restent faibles et aucun **cluster** solide ne se forme, ou bien les liens sont au contraire **renforcés** sur la base de signaux erronés (artefacts de bruit).

Le **DSL** reste donc partiellement tributaire d'un **nettoyage** ou d'une **consolidation** préliminaire si le **bruit** se montre trop massif, sous peine de limiter sa capacité à détecter les vraies complémentarités. Plusieurs stratégies peuvent alors intervenir pour filtrer ou rehausser la qualité des données :

- Un **filtrage statistique** peut écarter ou sous-pondérer les entités dont les mesures apparaissent trop incertaines (taux de données manquantes élevé, incohérences récurrentes).
- Une **fusion** locale d'entités similaires peut consolider leur information : si deux entités \mathcal{E}_i et \mathcal{E}_j se recouvrent largement ou présentent des mesures redondantes, il est envisageable de les grouper en une entité plus fiable et plus "complète".
- L'inclusion de **scores** de fiabilité dans la fonction de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ évite de s'appuyer lourdement sur des estimations basées sur un faible nombre de points ou sur des valeurs soupçonnées d'être bruitées. Par exemple, on peut moduler la similarité ou la distance d'après un **coefficent de confiance** attaché à chaque entité ou à chaque mesure.

Grâce à ces mécanismes, on endigue l'effet d'un bruit omniprésent tout en préservant la logique d'**auto-organisation** du **DSL**. Ainsi, même si certaines entités restent partiellement incomplètes, elles peuvent trouver leur place dans le **Synergistic Connection Network** dès lors qu'elles entretiennent au moins quelques synergies robustes avec d'autres entités.

1.7.2.3. Accès et Disponibilité Continue : le **DSL** comme Système en Ligne

Dans de nombreux environnements, il est nécessaire de gérer un **flux** ininterrompu de données et de réajuster en permanence les structures d'apprentissage. Le **Deep Synergy Learning (DSL)**, déjà décrit comme un processus de mise à jour itérative (voir la section 1.4.5), peut fonctionner en **mode "online"** si l'on s'assure que chaque entité \mathcal{E}_i reçoit ses observations de manière régulière ou semi-régulière. Lorsque de nouvelles mesures sont disponibles, la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ est recalculée localement (ou approximée sur une fenêtre glissante), puis la pondération $\omega_{i,j}(t+1)$ s'actualise selon l'équation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Cette logique d'évolution incrémentale requiert un **accès** continu aux flux entrants, que l'on traite idéalement par un système de gestion de messages (type broker Kafka) ou par un stockage minimal assurant la conservation des $\omega_{i,j}(t)$. Le réseau se reconfigure au fil de la réception de nouvelles données, sans qu'il soit nécessaire de reprendre un apprentissage exhaustif en mode batch. Dans un contexte industriel où la disponibilité des données peut être irrégulière, il arrive parfois que certaines entités n'aient aucune mise à jour pendant une période. Il faut alors conserver leurs liaisons $\omega_{i,j}(t)$ inchangées et ne pas pénaliser excessivement l'absence de mesures, conformément aux principes d'adaptation face aux lacunes de données (section 1.5.3). Lorsque la distribution évolue de manière brusque, le **DSL** réévalue les synergies de façon locale et progressive : dès qu'une entité se montre moins corrélée à son voisinage, ses pondérations décroissent et elle se retrouve partiellement isolée, tandis que de nouveaux liens peuvent se former avec d'autres entités plus en adéquation avec la situation nouvelle.

Du point de vue du **déploiement**, un tel système en ligne exige une **synchronisation** ou un **parallelisme** adapté. Chaque entité ne traite que ses connexions actives, ce qui maintient la complexité à un niveau proportionnel au nombre de liaisons conservées (voir la section 1.7.1.2 sur la parcimonie). Pour des applications massivement distribuées, il est possible de faire coexister plusieurs nœuds DSL partiels et de synchroniser leurs matrices ω par échanges de sous-graphes ou de résumés statistiques. Cette configuration favorise la **scalabilité** et la **résilience**, car si un nœud local subit une interruption ou un ralentissement, le reste du réseau n'est pas forcément de s'arrêter, et les autres entités continuent à ajuster leurs pondérations. Une telle conception rend le **DSL** apte à gérer des environnements en perpétuel changement, tout en intégrant les principes d'auto-organisation et de mise à jour continue.

1.7.2.4. Biais et Incohérences Structurelles

Dans le **Deep Synergy Learning (DSL)**, les liaisons $\omega_{i,j}$ émergent localement à travers la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Lorsque les données d'entrée sont biaisées, par exemple avec une surreprésentation de certains profils (sections 1.5.3 et 1.7.2.1), le réseau a tendance à **renforcer** les clusters correspondant aux **groupes majoritaires**, tout en marginalisant des entités issues de groupes minoritaires ou insuffisamment renseignés. Dans des domaines critiques comme le **diagnostic médical** ou la **recommandation sensible**, un tel phénomène peut avoir pour effet d'**entériner** des inégalités ou de mener à des **erreurs d'interprétation** (sous-diagnostic pour certains patients, ou éviction de contenus pertinents pour certaines catégories d'utilisateurs).

À la différence d'un **modèle supervisé** où l'on surveille une **loss** globale sur l'ensemble des données, le DSL base ses ajustements sur des **mesures locales** de synergie. Même si la structure globale Ω peut finir par présenter un arrangement biaisé, la démarche d'**auto-organisation** ne possède pas toujours de mécanismes internes de correction. Les entités minoritaires voient leurs pondérations $\omega_{i,j}$ rester faibles, du fait qu'elles ne rencontrent pas suffisamment de synergie avec la majorité. Par ailleurs, si une **incohérence** surgit ou si un artefact statistique biaisé devient prépondérant, le DSL peut stabiliser un **cluster** aberrant, en l'absence de contrôle externe.

Pour pallier cette difficulté, il est indispensable de prévoir des **mécanismes de validation** ou de **monitoring** global. On peut, par exemple, injecter des **entités symboliques** (sections 1.5.7 et 1.7.2.3) qui portent des règles ou des normes éthiques, limitant la croissance de clusters injustes ou repérant les configurations contradictoires. On peut également imposer un **feedback** externe, ou un signal de performance englobant des critères d'équité, veillant ainsi à empêcher la fermeture de groupes déviants. La structure ω peut alors être réévaluée à la lumière de ces **contraintes** d'équité ou de cohérence, conférant au réseau un degré de régulation qui contrebalance les dérives potentielles issues de la simple coopération locale.

1.7.2.5. Conclusion

La **qualité** et la **disponibilité** des données sont cruciales pour le **Deep Synergy Learning**, tout comme pour la plupart des approches d'IA. Néanmoins :

- Le **DSL** peut mieux tolérer la **partialité** (en isolant les entités trop bruitées),
- Il s'adapte mieux à un **flux continu** (mise à jour locale des pondérations),
- Il demeure sujet à des **biais** et à un besoin de **représentativité** dans les données.

Ainsi, un important **travail préparatoire** est souvent nécessaire pour :

Filtrer ou pondérer la qualité des données,

Introduire des **scores de confiance** dans la synergie,

Établir une **stratégie** de streaming ou de batch mixte pour nourrir la dynamique ω ,

Surveiller la **formation** de clusters anormaux ou la propagation de biais dans l'auto-organisation.

Réussir ce pilotage des données est fondamental pour tirer profit du DSL : un système auto-organisé ne saurait corriger des écueils majeurs de collecte ou de représentativité sans un minimum d'**ingénierie** et de **gouvernance** des données.

1.7.3. Développement d'Algorithmes d'Optimisation Appropriés

Une fois admise la nécessité de gérer la **complexité** (1.7.1) et de soigner la **qualité** des données (1.7.2), se pose la question cruciale des **algorithmes** qui permettront de **piloter** la dynamique du **Deep Synergy Learning (DSL)**. En effet, ce paradigme repose sur l'**auto-organisation** de multiples entités $\{\mathcal{E}_i\}$ et sur la **mise à jour** de l'ensemble des pondérations synergiques $\omega_{i,j}(t)$. Dans le modèle théorique simple, on écrit souvent :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

Où $S(\mathcal{E}_i, \mathcal{E}_j)$ désigne la **synergie** entre ces entités.

En pratique, mettre en œuvre cette loi pour un grand nombre d'entités, avec des synergies potentiellement complexes (non linéaires, conditionnelles, n-aires), n'est pas trivial. On a besoin de **méthodes** d'optimisation aptes :

À **stabiliser** le système (éviter les oscillations permanentes ou l'explosion des pondérations),

À **accélérer** la convergence vers des configurations pertinentes (clusters, macro-clusters),

À prendre en compte des **contraintes** (symboliques, de cohérence, de parcimonie),

À **s'adapter** quand de nouvelles entités ou données apparaissent (flux continu).

Cette sous-section (1.7.3) discute quelques **pistes** mathématiques et algorithmiques pour rendre le DSL plus solide et plus efficace :

- Des méthodes **locales** (mise à jour distribuée),
- Des formulations **globales** (fonction d'énergie, descente de gradient généralisée, approches de type recuit simulé),
- Des **heuristiques** ou des règles inspirées de la biologie et des systèmes complexes (sélection, reproduction, extinction),
- Des **extensions** pour la synergie n-aire ou conditionnelle.

1.7.3.1. Approche Locale et Distribuée

Dans l'esprit **biologique** et de nombreux modèles de **systèmes complexes**, on peut envisager la mise à jour de chaque pondération $\omega_{i,j}$ de manière **locale**, sans qu'un contrôle global impose la synchronisation. Chaque entité \mathcal{E}_i communique uniquement avec ses **voisins** au sens du graphe, autrement dit les entités \mathcal{E}_j pour lesquelles $\omega_{i,j}(t)$ est significatif. L'entité \mathcal{E}_i calcule alors la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ sur la base de mesures ponctuelles (ou d'un historique local) et applique la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta_{i,j}(t) [S_{i,j}(t) - \tau_{i,j}(t) \omega_{i,j}(t)].$$

Cette équation reprend le formalisme général du **Deep Synergy Learning** (section 1.4.5) en autorisant des coefficients $\eta_{i,j}(t)$ et $\tau_{i,j}(t)$ qui dépendent de la paire (i,j) ou varient dans le temps, procurant une **régulation** fine. L'**avantage** est de rendre l'algorithme **parallélisable**, puisque chaque entité \mathcal{E}_i peut mettre à jour ses liaisons sans solliciter un "master" central. Une difficulté tient toutefois à la **synchronisation** : si plusieurs entités se mettent à renforcer ou à affaiblir simultanément des liaisons communes, des boucles instables ou des conflits pourraient apparaître. Des stratégies d'**asynchronisme** maîtrisé ou de communication par itérations (steps) successives sont alors recommandées.

Dans cette optique, on peut se référer à diverses **métaphores** ou **règles** d'inspiration biologique. Par exemple, une **mise à jour** "hebbienne" renforce la pondération $\omega_{i,j}$ lorsqu'on observe une co-activation récurrente des entités \mathcal{E}_i et \mathcal{E}_j , et l'affaiblit en l'absence de co-occurrence. Par ailleurs, on peut procéder à une **sélection** ou "reproduction" des liaisons : si certaines paires (i, j) affichent un score ou une utilité trop faible de manière persistante, elles sont éliminées, tandis que d'autres liens se consolident ou se "reproduisent" (création de liaisons analogues sur d'autres nœuds). Une **compétition** locale peut également réguler la densité : on peut contraindre la somme $\sum_j \omega_{i,j}$ pour chaque entité \mathcal{E}_i ou recourir à un "softmax" local, privilégiant quelques connexions fortes plutôt qu'un grand nombre de liens faibles.

Par ces variantes, on parvient à **contrôler** la **densité** du graphe, la vitesse de renforcement des pondérations et la **résilience** face au bruit ou aux données incomplètes (sections 1.7.1.2 et 1.5.3). Le **DSL** gagne alors en **scalabilité** tout en maintenant le principe d'**auto-organisation** : chacun de ces mécanismes demeure local, assurant une dynamique distribuée qui s'accorde bien à des mises en œuvre parallèles (GPU, TPU) ou à des approches multi-agents où chaque entité évolue au fil de ses propres observations et interactions.

1.7.3.2. Formulation Globale via Fonction d'Énergie

Dans une perspective plus proche de l'**optimisation** (au sens des approches traditionnelles en IA ou en physique statistique), il est concevable de décrire l'**auto-organisation** du Deep Synergy Learning (DSL) par une **fonction d'énergie** $\mathcal{J}(\Omega)$. On peut écrire une expression qui généralise la somme des synergies et la pondère par un terme de **régularisation**, selon :

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S_{i,j} + R(\Omega).$$

Dans ce formalisme, la somme $\sum_{i,j} \omega_{i,j} S_{i,j}$ valorise la **coopération** (au sens de la synergie, voir sections 1.4.4 et 1.4.5), tandis que le terme $R(\Omega)$ constitue une **régularisation**, par exemple $\alpha \|\Omega\|^2$ ou $\alpha \|\Omega\|_0$, qui maintient la parcimonie ou la cohérence structurelle du **Synergistic Connection Network**. Cette formulation unifie les principes évoqués en 1.7.1.4 (contrôle de la densité, maintien de contraintes) dans une unique fonction de coût : l'objectif global consiste à **maximiser** la somme des synergies ou, en termes d'énergie, à **minimiser** \mathcal{J} .

Pour résoudre l'évolution de Ω dans ce cadre, on peut recourir à une **descente de gradient** : si l'on note $\omega_{i,j}$ un des paramètres, sa dérivée partielle vis-à-vis de \mathcal{J} prend la forme

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S_{i,j} + \frac{\partial R(\Omega)}{\partial \omega_{i,j}}.$$

En discréétisant cette descente, on retrouve l'équation qui met à jour $\omega_{i,j}$ de façon incrémentale :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{i,j} - \tau \omega_{i,j}(t)],$$

ce qui explicite la proximité entre la **logique locale** décrite en 1.4.5 et la **formulation** d'une énergie globale $\mathcal{J}(\Omega)$. Dans les problèmes complexes, la descente de gradient peut aboutir à des minima locaux sous-optimaux. On peut alors envisager un **recuit simulé** (simulated annealing) directement sur l'espace du graphe Ω . Cette méthode consiste à perturber par moments la configuration courante (création ou suppression de liens), et à n'accepter ou refuser ces modifications qu'avec une probabilité liée à la variation $\Delta \mathcal{J}$ et à un paramètre de température qui décroît au fil des itérations. Dans le contexte du DSL, où la synergie peut être n-aire (voir la section 1.4.7), cette exploration globale de l'espace combinatoire peut s'avérer fondamentale : la simple descente par ajustements locaux risquerait de rester piégée dans un *cluster* partiel. Le recuit simulé ou des **algorithmes évolutionnaires** jouent alors le rôle de recherche d'une configuration plus robuste, accompagnant l'auto-organisation locale. Cette combinaison de l'approche **par énergie** et de la **dynamique en continu** ouvre une voie pour maîtriser mathématiquement la convergence et la structure du **Synergistic Connection Network**, en particulier lorsqu'on cherche à maintenir une **parsimonie** explicite ou à inclure des contraintes topologiques (voir 1.7.1.4).

1.7.3.3. Synergie n-aire et Extensions Conditionnelles

Le **Deep Synergy Learning (DSL)** se focalise souvent sur la **coopération** entre deux entités \mathcal{E}_i et \mathcal{E}_j . Toutefois, certains problèmes exigent une **synergie n-aire**, c'est-à-dire impliquant un **groupe** $\{\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_n}\}$ de cardinal $n \geq 3$. Dans ce cas, on peut introduire des **hyper-arêtes** ω_{i_1, \dots, i_n} plutôt que de simples liaisons binaires. La mise à jour de la pondération hyper-arête s'effectue alors selon une généralisation de la loi (voir section 1.4.5), par exemple :

$$\omega_{i_1, \dots, i_n}(t+1) = \omega_{i_1, \dots, i_n}(t) + \eta [S_n(\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_n}) - \tau \omega_{i_1, \dots, i_n}(t)].$$

Dans cette équation, la fonction $S_n(\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_n})$ évalue la **synergie** globale de l'ensemble $\{\mathcal{E}_{i_k}\}$. Sa définition peut s'appuyer sur une co-information n-aire, sur un gain de performance collectif ou sur toute métrique reflétant l'apport mutuel de ce sous-groupe. Néanmoins, la **complexité** combinatoire s'en trouve rapidement démultipliée, car on doit gérer des hyper-arêtes potentiellement innombrables lorsque n augmente. Pour cette raison, il est fréquent de recourir à des **heuristiques** afin de détecter des triplets, quadruplets, etc., seulement lorsque l'on constate une coopération stable entre ces entités ; on se contente de créer l'hyper-arête ω_{i_1, \dots, i_n} après avoir observé à maintes reprises que ces n entités coopèrent ensemble.

Par ailleurs, il arrive que la **synergie** dépende d'un **contexte c**. Un exemple typique survient lorsque l'association audio–texte n'est pertinente que dans une situation donnée (un lieu, une heure, une ambiance sonore). La règle de mise à jour se récrit alors :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j \mid \mathbf{c}(t)) - \tau \omega_{i,j}(t)].$$

Dans ce cadre, on doit calculer un $S(\cdot \mid \mathbf{c})$ pouvant fortement varier en fonction du contexte. La tâche d'**estimation** et d'**optimisation** se complexifie davantage : la distribution induite par $\mathbf{c}(t)$ peut changer au fil du temps ou comprendre divers régimes de fonctionnement. Les algorithmes d'auto-organisation du DSL doivent alors incorporer des modèles de synergie **conditionnelle**, qui modulent la pondération $\omega_{i,j}$ suivant la valeur prise par \mathbf{c} . Cette extension demeure essentielle dans de multiples applications, comme la robotique adaptative (un même couple de capteurs se révèle pertinent ou non selon le mode de locomotion) ou le traitement multimodal (un segment audio et un mot ne coïncident que dans un contexte temporel précis). Elle requiert cependant des stratégies de **parcimonie** ou de **sampling** pour éviter une explosion combinatoire lorsqu'on veut évaluer la synergie conditionnelle sur tous les contextes possibles.

1.7.3.4. Contrôle de la Stabilité et Convergence Globale

Dans tout **système dynamique** présentant un large couplage entre ses éléments, il est fréquent d'observer des **cycles** ou des **oscillations** : un sous-réseau se renforce pendant un temps, puis un autre prend le relais en affaiblissant les premiers liens, et ainsi de suite. Il peut aussi survenir un phénomène de **basculement** entre plusieurs attracteurs stables (configurations rivales) si la structure du **Synergistic Connection Network** autorise plusieurs manières de s'organiser pour la même situation. Le **Deep Synergy Learning (DSL)** n'échappe pas à ces scénarios, en particulier lorsque la fonction de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ est très "non linéaire" ou quand la synergie n-aire entre plusieurs entités génère des interactions complexes (sections 1.4.7 et 1.7.3.3).

Pour limiter ce risque d'oscillations ou de dérive, il convient d'inclure des **mécanismes** ou des **règles** de contrôle dans la mise à jour des pondérations $\omega_{i,j}$. On peut imposer un **amortissement** progressif (faire varier η ou τ dans le temps), intégrer une **pénalisation** plus agressive des liens excessivement grands, ou encore définir des **seuils de saturation** empêchant une pondération $\omega_{i,j}$ de croître sans borne. Des analyses mathématiques, telles que l'étude de la **Jacobienne** locale ou des **fonctions de Lyapunov** globales, peuvent démontrer qu'un schéma de mise à jour particulier garantit la convergence vers un état stable (ou au moins vers un cycle stable), sans risquer un chaos indésirable.

Il est également envisageable de greffer un **module** de contrôle hiérarchique (de plus haut niveau), qui surveille la dynamique globale du réseau et injecte ponctuellement des modifications de paramètres η ou τ si la variance du système dépasse un certain seuil. Ce principe d'**hybridation** (auto-organisation libre + régulation macro) rappelle certaines structures biologiques ou cognitives, où les ajustements locaux cohabitent avec des boucles de rétroaction plus centrales. On cherche alors à préserver la **liberté synergique** au niveau local, tout en s'assurant que le **réseau** ne

sombre pas dans une instabilité excessive. Cet équilibre soulève des questions d'**optimisation** et de **pilotage** du **DSL**, qui restent ouvertes à la recherche, notamment sur les plans pratique (implémentation à grande échelle) et théorique (analyse de la dynamique dans un espace de forte dimension).

Conclusion

Le **développement d'algorithmes d'optimisation** pour le **Deep Synergy Learning** s'annonce particulièrement riche et complexe. Les **règles de mise à jour** naïves suffisent à illustrer le concept, mais ne passent pas toujours à l'échelle ni ne gèrent la synergie n-aire, les contraintes symboliques, ou la minimisation globale d'une fonction d'énergie. Les approches possibles incluent :

- **Mise à jour locale** inspirée de la biologie (sélection, Hebb, normalisation compétitive),
- **Descente d'énergie** globale (fonctions \mathcal{J} pénalisant la densité, favorisant la synergie),
- **Algorithmes évolutifs** (recuit simulé, heuristiques stochastiques) pour éviter les minima locaux,
- **Extensions** aux hyper-arêtes (synergies n-aires), aux synergies conditionnelles, ou aux règles symboliques,
- **Contrôles** de stabilité et d'amortissement, voire un superviseur partiel pour limiter les oscillations et injecter des priorités.

Trouver la meilleure **formulation** et le meilleur **algorithme** dépendra de l'application (vision, robotique, recommandation, etc.), du **volume** de données et de la **structure** (binaire ou n-aire, symbolique, multi-modal). La poursuite de ces recherches est incontournable pour que le **DSL** gagne en robustesse, en efficacité, et en adoptabilité dans le monde réel.

1.7.4. Contrôle et Stabilité des Processus Auto-Organisés

L'**auto-organisation** est au cœur du **Deep Synergy Learning (DSL)**, permettant à un grand nombre d'entités (capteurs, modules de calcul, représentations symboliques ou sub-symboliques) de s'agencer spontanément en **clusters** ou **macro-clusters** selon leurs synergies. Cette dynamique, tout en conférant au DSL sa **plasticité** et sa **capacité d'adaptation**, peut aussi engendrer des **problèmes de stabilité** : oscillations, attracteurs multiples, comportements chaotiques, etc. De plus, dans bien des cas (robotique, diagnostic, logistique), il faut **contrôler** partiellement la configuration auto-organisée, ne serait-ce que pour garantir la **sécurité**, la **fiabilité**, ou la **cohérence** avec des règles externes.

Dans cette sous-section (1.7.4), nous examinons les **défis** posés par la **dynamique** d'un système DSL, et les **solutions** ou **mécanismes** possibles pour assurer un certain niveau de **contrôle** et de **stabilité** :

- La tendance des systèmes couplés à générer des **oscillations** ou des **cycles**,
- Les risques d'**explosion** ou d'**effondrement** des pondérations synergiques ω ,
- L'existence de **multiples attracteurs** en concurrence, pouvant conduire à des configurations divergentes,
- Les **dispositifs** de rétroaction ou de surveillance globale pour introduire un pilotage **hiérarchique** léger,
- Les **garanties** (ou non-garanties) de convergence sous certaines hypothèses mathématiques.

1.7.4.1. Risques d'Oscillations et de Comportements Chaotiques

Dans le **Deep Synergy Learning**, chaque liaison $\omega_{i,j}$ obéit à une évolution itérative inspirée des principes décrits en section 1.4.5, ce qui donne lieu à un **système dynamique** d'environ $O(n^2)$ degrés de liberté dans le cas binaire. Pour comprendre comment apparaissent des **cycles** ou des **régimes oscillatoires**, il est éclairant d'étudier la **linéarisation** locale autour d'un point fixe ω^* . Supposons l'existence d'un équilibre ω^* satisfaisant un état stationnaire ; si l'on introduit une petite perturbation $\delta\omega$, la dynamique globale s'écrit de manière approchée :

$$\delta\omega(t+1) \approx J(\omega^*) \delta\omega(t),$$

où $J(\omega^*)$ désigne la **matrice jacobienne** évaluée en ω^* . En temps discret, un comportement oscillatoire ou pseudo-chaotique survient lorsque certaines **valeurs propres** de J possèdent une magnitude supérieure à 1. Cela signifie que la perturbation $\delta\omega$ se voit **amplifiée** au fil des itérations, au lieu d'être dissipée. En temps continu, la condition équivalente consisterait à observer des **parties réelles** positives dans les valeurs propres, menant à une instabilité locale ou à des boucles auto-entretenues.

Ce type de phénomène peut se traduire par des **cycles** permanents, ou même des trajectoires quasi chaotiques, contrariant la **convergence** du réseau vers un état stable. De tels régimes, bien que mathématiquement fascinants, s'avèrent peu souhaitables si l'on souhaite une structure DSL stable pour la robotique, la maintenance ou la prise de décision. C'est pourquoi on introduit souvent des **mécanismes** de stabilisation. Une première solution consiste à contrôler la "vitesse" de mise à jour en imposant une contrainte, par exemple $\eta \tau < 1$, dans le cas linéaire simplifié. On peut ensuite recourir à des **termes** non linéaires de saturation, destinés à éviter la croissance illimitée de certaines pondérations. Un exemple simple, dans un modèle continu, consiste à enrichir l'équation d'évolution par un terme $-\beta (\omega_{i,j})^3$:

$$\frac{d}{dt} \omega_{i,j} = \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}] - \beta (\omega_{i,j})^3,$$

où $\beta > 0$ modère la croissance des liaisons fortes. De manière analogue, on peut introduire un **facteur** de compétition locale (type softmax), en imposant $\sum_j \omega_{i,j} \leq K$ pour chaque entité \mathcal{E}_i , de sorte qu'une entité ne puisse pas développer des liens solides avec trop de partenaires en même temps.

Ces mesures limitent la **liberté** d'auto-organisation, mais s'avèrent cruciales pour écarter des régimes oscillatoires ou chaotiques susceptibles d'émerger dans un réseau fortement couplé et non linéaire. Elles assurent ainsi une forme de **stabilité** globale tout en préservant la possibilité de réorganisations locales dictées par la synergie. Un savant réglage des paramètres (η , τ , β , ou le budget $\sum_j \omega_{i,j} \leq K$) permet donc de maintenir le **DSL** dans une zone où le réseau demeure robuste et apte à converger, évitant des fluctuations intempestives tout en conservant l'essence auto-adaptative voulue.

1.7.4.2. Multiples Attracteurs, Convergence Incertaine

Il est fréquent qu'un **Deep Synergy Learning (DSL)** cherche à maximiser, ou à minimiser l'opposé, d'une **somme de synergies** $\sum_{i,j} \omega_{i,j} S_{i,j}$ éventuellement modulée par un terme de pénalisation ou de parcimonie (voir section 1.7.1.4). Dans un espace de grande dimension, on peut rencontrer de **multiples attracteurs** correspondant à des configurations dont l'énergie (ou le coût) est quasi équivalente. Ce phénomène se manifeste lorsqu'il existe plusieurs manières d'**arranger** les entités en clusters, chacune procurant un gain global proche. Le réseau peut alors converger vers l'un ou l'autre de ces attracteurs selon les conditions initiales ou de légères fluctuations.

Dans la pratique, il arrive que des "sous-réseaux" rivaux émergent au cours de la dynamique, chacun rassemblant certaines entités en un **cluster** distinct. Une compétition peut alors s'installer entre ces sous-réseaux, induisant des **instabilités** transitoires et des oscillations (voir section 1.7.4.1). Si l'on désire un **état** unique et stable, par exemple pour un robot qui ne peut se scinder en deux stratégies opposées, on peut injecter un **signal global** orientant le réseau vers l'attracteur souhaité. Cette intervention se formalise en modifiant la fonction de synergie ou en ajoutant un **feedback** de récompense plus élevé pour une configuration A que pour une configuration B. On obtient ainsi un **pilotage** macro qui restreint la pure auto-organisation, mais assure qu'en présence de minima multiples, la

configuration la plus souhaitable selon les critères d'usage devienne prépondérante. Cette démarche montre le besoin de compromis entre la liberté synergique propre au DSL et la nécessité d'un **contrôle** hiérarchique dans certaines applications critiques.

1.7.4.3. Approches Hiérarchiques Légères pour Guider l'Auto-Organisation

Dans des environnements critiques, tels que le domaine **médical** ou les **transports**, il n'est pas toujours acceptable de laisser un réseau en **Deep Synergy Learning (DSL)** se reconfigurer librement sans un regard d'ensemble. Afin de maîtriser la sécurité et la fiabilité, on introduit souvent une **couche** ou un **module** de supervision qui contrôle globalement la structure Ω . Ce superviseur observe la **matrice** des liens actifs et vérifie des **contraintes** imposées par l'application. Par exemple, on peut exiger que la somme $\sum_j \omega_{i,j}$ attachée à une entité \mathcal{E}_i ne dépasse pas un seuil spécifique, ou que certaines entités déclarées incompatibles (règles de sécurité, normes médicales contradictoires) ne figurent pas dans le même cluster. Si ce superviseur détecte une violation, il peut forcer la **mise à zéro** de certaines liaisons $\omega_{i,j}$, neutralisant ainsi des configurations illégitimes.

On peut aussi concevoir un **schéma** où la **couche basse** demeure l'auto-organisation locale et distribuée, tandis qu'une **couche haute** (un planificateur ou un décideur global) injecte à intervalles réguliers des **influx** spécifiques dans le réseau. Ces influx peuvent prendre la forme d'une modification ponctuelle de $\omega_{i,j}$, d'un renforcement sélectif d'une synergie, ou de l'assignation de priorités à certaines entités. Cet apport d'information agit comme une **guidance** de haut niveau, combinée à la dynamique adaptative du DSL.

D'un point de vue mathématique, on peut l'interpréter comme l'ajout d'un **terme** exogène dans l'équation de mise à jour, par exemple $\Delta\omega_{i,j}^{(\text{macro})}$ reflétant la volonté du **planificateur**. Cela permet d'enrichir la structure Ω sans remettre en cause l'intégralité du principe d'**auto-organisation** locale. On obtient un **compromis** dans lequel la dynamique $\omega_{i,j}$ se déroule principalement selon la synergie perçue, tout en restant arrimée à des objectifs de **performance** ou de **sécurité** plus vastes. Dans ce type d'architecture dite « hybride », le contrôle hiérarchique léger se borne à préserver la cohérence, tandis que la plupart des adaptations opérationnelles sont gérées par la mise à jour distribuée des liaisons.

1.7.4.4. Outils Mathématiques et Théoriques pour la Stabilité

Il est souvent utile de modéliser l'**évolution** d'un **Deep Synergy Learning (DSL)** sous la forme d'un **système dynamique** $\omega(t+1) = \omega(t) + F(\omega(t))$. Dans cette notation, $\omega(t)$ regroupe l'ensemble des pondérations $\omega_{i,j}(t)$, et $F(\omega(t))$ décrit la contribution que la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ (ainsi que les termes de décroissance, de régularisation, etc.) apporte à la variation entre t et $t+1$.

Pour étudier la stabilité locale autour d'un **point fixe** ω^* (ou d'une configuration fixe du réseau), on réalise classiquement une **linéarisation** de F . Concrètement, on calcule la **matrice jacobienne** :

$$J(\omega^*) = \frac{\partial F}{\partial \omega}(\omega^*).$$

Si l'on note $\delta\omega(t) = \omega(t) - \omega^*$, alors pour de petites perturbations, on obtient un modèle linéarisé :

$$\delta\omega(t+1) \approx \delta\omega(t) + J(\omega^*) \delta\omega(t).$$

En régime de temps discret, la **stabilité** locale autour de ω^* se produit si toutes les **valeurs propres** de $I + J(\omega^*)$ ont une magnitude strictement inférieure à 1. Dans un cadre continu (où l'on écrit $d/dt \omega = F(\omega)$), il faut que toutes les parties réelles des valeurs propres de $J(\omega^*)$ soient **négatives**. Cette condition s'interprète physiquement : de petites déviations $\delta\omega$ s'amortissent au fil du temps. Elle dépend du **choix** de la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$, de la taille du pas η , des paramètres de décroissance τ , ou des non-linéarités supplémentaires (termes de saturation, etc.).

Parfois, on peut exhiber une **fonction de Lyapunov** $\Phi(\omega)$ qui décroît strictement à chaque mise à jour, prouvant la **convergence** vers un **unique attracteur**. Cela exige que la définition de la synergie ne comporte pas de **non-linéarités**

trop complexes, ou que la structure ne s'étende pas à des hyper-arêtes de dimension élevée (voir la section 1.4.7 sur la synergie n-aire). Dans des **configurations** plus simples (synergie linéarisable, réseau symétrique, etc.), on peut démontrer rigoureusement l'existence et l'unicité d'un état stationnaire stable. En revanche, pour des **formulations** plus riches (synergie hautement non linéaire, dépendance au contexte, hyper-arêtes n-aires), établir une **preuve** de stabilité globale devient complexe : la dynamique peut potentiellement présenter de multiples attracteurs ou des régimes oscillants.

Ces outils (linéarisation jacobienne, fonctions de Lyapunov) offrent néanmoins un **cadre théorique solide** pour analyser la mise à jour des pondérations $\omega_{i,j}$ dans un **DSL**, en particulier lorsque l'on cherche à garantir un comportement prédictible et éviter les **oscillations** chaotiques ou l'explosion des liaisons. Ils fournissent aussi des guides de **conception** : on peut paramétriser η et τ ou imposer des contraintes de somme, de norme, etc. afin de maintenir la trajectoire de ω à l'intérieur d'une **réglage** stable et de faire en sorte que le système converge réellement, au lieu de dériver ou de switcher entre multiples configurations.

Conclusion

Assurer le **contrôle** et la **stabilité** des **processus auto-organisés** dans le **DSL** est un **enjeu fondamental** pour :

- Prévenir l'apparition de **comportements oscillants** ou chaotiques,
- Gérer la coexistence de **multiples attracteurs** et choisir (ou influencer) l'état final,
- Introduire un **encadrement** (module de supervision, règles hiérarchiques minimales) pour les systèmes critiques ou lorsqu'on souhaite une trajectoire de réorganisation déterministe.

Les **solutions** mathématiques ou algorithmiques incluent :

- La **conception** de règles de mise à jour (amortissement, saturation, normalisation) qui limitent la croissance excessive des pondérations,
- L'introduction d'une **énergie globale** $J(\Omega)$, encadrée par des méthodes de descente ou de recuit simulé,
- Des **mécanismes de pilotage macro** (modification de η, τ , injection de priorités, filtration de certaines liaisons) dans un schéma hybride auto-organisé + contrôle haut-niveau,
- Des **analyses** de stabilité via Jacobienne locale ou Lyapunov, pour garantir (au moins localement) la convergence ou la non-explosion.

En somme, maîtriser l'**auto-organisation** ne signifie pas brider la flexibilité du DSL, mais offrir à l'architecture la **robustesse** nécessaire pour opérer en conditions réelles, où l'on ne peut se permettre un comportement erratique ni une multiplicité de configurations irréconciliables.

1.7.5. Interprétabilité et Explicabilité pour l'Humain

Un objectif crucial dans la plupart des applications d'**intelligence artificielle** est de fournir non seulement des **résultats** (prédictions, recommandations, décisions), mais également une **compréhension** de la logique ou du **processus** qui y aboutit. Les méthodes classiques de **Deep Learning** (CNN, RNN, Transformers) sont souvent décriées pour leur caractère "**boîte noire**", dans la mesure où l'information se diffuse à travers des poids distribués dans de multiples couches, rendant l'**explication** pour l'humain délicate.

Le **Deep Synergy Learning (DSL)**, de par sa **structure** en **entités** et en **liens synergiques** ($\omega_{i,j}$), fournit un cadre potentiellement plus **transparent** : on peut visualiser **quelles** entités interagissent fortement, **comment** se constituent les **clusters** ou **macro-clusters**, et **pourquoi** ces regroupements émergent (sur la base d'une synergie mesurable). Toutefois, obtenir une **interprétabilité** réelle et exhaustive n'est pas garanti : il faut des **mécanismes** méthodologiques

pour extraire des **explications** compréhensibles. Cette sous-section (1.7.5) discute les **défis** et **pistes** liées à l'explicabilité dans le DSL.

1.7.5.1. D'un Modèle Hiérarchique Opaque à un Réseau de Liens Synergiques

Il est courant que les **réseaux neuronaux profonds** (CNN, RNN, Transformers) diluent leur représentation interne à travers plusieurs **couches** qui transforment graduellement les descripteurs, rendant ainsi les décisions finales difficiles à expliquer. Les neurones internes n'ont pas d'**identité** propre : ils sont simples réceptacles de poids et d'activations, et il est souvent ardu d'y associer un sens direct. À l'inverse, le **Deep Synergy Learning (DSL)** introduit un **réseau** où chaque **noeud** \mathcal{E}_i possède une **identité** clairement définie (il peut s'agir d'un **capteur** particulier, d'un **concept** sémantique, d'un **patch** visuel, d'une **règle** symbolique, etc.), tandis que les **liaisons** $\omega_{i,j}$ se **reconfigurent** progressivement, au gré de la synergie détectée entre \mathcal{E}_i et \mathcal{E}_j .

Cette structure procure plusieurs **avantages**. Elle favorise d'abord la **lisibilité** de certains **sous-ensembles** : on peut dénicher des **clusters** (chapitre 1.4.3) formés par un groupe d'entités entretenant des liens forts, et y voir immédiatement une **cohérence** (par exemple, un ensemble “visuel + audio” en reconnaissance multimodale ou un ensemble “utilisateur + contenus + tag de genre” en recommandation). Par ailleurs, chaque **entité** \mathcal{E}_i se voit attribuer une **signification** (un patch d'image, un concept symbolique), ce qui facilite grandement l'**explication** du raisonnement ou de la recommandation. Un **macro-cluster** peut ainsi refléter un groupement plus vaste : “groupe d'utilisateurs amateurs de jazz + contenus jazz + soirées concert” dans une plateforme culturelle, offrant une justification naturelle à la recommandation “puisque vous faites partie du cluster jazz, on vous propose ce concert”.

Cependant, cette **lisibilité** accrue sur le plan conceptuel s'accompagne aussi d'une **complexité** potentiellement élevée. Le réseau DSL peut afficher un grand nombre de **noeuds** (surtout en contexte industriel ou multi-sensoriel), et chaque **noeud** peut avoir plusieurs **connexions** significatives, aboutissant à une **topologie** complexe, souvent composée de **clusters** imbriqués ou de multiples liens réciproques. Une simple **visualisation** naïve des poids $\omega_{i,j}$ peut alors ne rien révéler de clair, s'il n'existe pas de mécanisme de filtrage ou de simplification. Il devient nécessaire d'utiliser des **outils de hiérarchisation**, de **regroupement** ou de **filtrage** pour dériver des **chemins** explicatifs, par exemple en extrayant les principales **chaînes** de liaison qui influencent la décision finale. C'est là que réside l'enjeu d'une **explicabilité** réellement transparente dans un réseau DSL riche, qui exige de se doter de techniques aptes à condenser le graphe et à exposer les assemblées cruciales, tout en évinçant la masse de connexions moins pertinentes.

1.7.5.2. Extractions de Chemins Synergiques et Clusters Pivot

Dans un **réseau** construit selon les principes du **Deep Synergy Learning (DSL)**, il est possible d'expliquer une décision ou une recommandation en identifiant un **chemin** significatif reliant deux entités \mathcal{E}_a et \mathcal{E}_b . On peut ainsi considérer une suite :

$$\mathcal{E}_a \rightarrow \mathcal{E}_x \rightarrow \mathcal{E}_y \rightarrow \mathcal{E}_b,$$

au long de laquelle la **synergie** s'avère forte. On peut évaluer cette force par un produit du type $\omega_{a,x} \omega_{x,y} \omega_{y,b}$, exprimant la continuité du lien depuis \mathcal{E}_a jusqu'à \mathcal{E}_b via \mathcal{E}_x et \mathcal{E}_y . Par exemple, si \mathcal{E}_a est un **utilisateur** et \mathcal{E}_b un **contenu** recommandé, un **chemin** pertinent pourrait comporter un **tag** musical \mathcal{E}_x et un **artiste** \mathcal{E}_y : la synergie souligne alors que l'utilisateur \mathcal{E}_a est historiquement associé à ce tag rock, lequel est fortement relié à l'artiste \mathcal{E}_y , et que cet artiste conduit au **contenu** \mathcal{E}_b . L'explication se formule sous la forme d'un **raisonnement local** : “Vous êtes lié à ce tag rock, ce tag rock est fortement corrélé à l'artiste \mathcal{E}_y , et cet artiste conduit au contenu \mathcal{E}_b proposé.” Dans la pratique, on limite souvent la longueur du **chemin** afin de conserver une **explication** concise et compréhensible, tout en montrant comment la synergie chemine dans le **Synergistic Connection Network**.

Une autre forme d'explication s'appuie sur le **cluster** (ou **macro-cluster**) dans lequel réside l'entité \mathcal{E}_b qu'on veut justifier. On expose alors les **entités dominantes** de ce cluster : par exemple, en diagnostic médical, on peut lister certains **indicateurs** (fièvre, anomalie sanguine, image radio suspecte) qui forment ensemble un sous-réseau hautement interconnecté. L'**interprétation** émerge alors de la **cohérence interne du cluster** : “Ces signes, fortement liés par la

synergie, corroborent la conclusion de pathologie.” Cette méthode est très usitée lorsque la configuration ne s’explique pas par un unique chemin, mais par un **ensemble** d’interactions locales qui se cristallisent en un groupement solidaire. L’analyse du cluster met en avant les entités centrales et leur degré de liaison ω , rendant la décision plus **transparente**.

1.7.5.3. Couplage Symbolique pour des Justifications de Haut Niveau

Dans la section 1.5.7, il a été expliqué comment intégrer des **règles** ou des **concept**s symboliques au sein du **Deep Synergy Learning (DSL)** de manière à fusionner des entités strictement sub-symboliques (features perceptuelles, segments, etc.) et des entités logico-sémantiques. Cette intégration prend toute son importance pour l'**explicabilité**, car elle autorise la production de justifications directement lisibles, sans nécessiter un métamodèle séparé. Si l’on imagine une **règle** \mathcal{E}_{rule} affirmant “ cercle + chiffre => panneau de limitation”, la présence d’un lien fort $\omega_{rule, visCircle}$ met en évidence la façon dont la règle coopère avec l’entité $\mathcal{E}_{visCircle}$ qui détecte les formes circulaires, et un autre lien $\omega_{rule, txt30}$ connecte cette règle à l’entité textuelle représentant le chiffre “30”. La combinaison de ces liaisons convainc ainsi que la **reconnaissance** du panneau 30 km/h découle d’une synergie tangible entre la règle symbolique “ cercle + chiffre” et les observations sub-symboliques qui valident les deux prérequis.

Cette **cohabitation** symbolique–subsymbolique s’avère donc profitable pour expliquer pourquoi une règle s’active, de quelle façon elle est validée par la scène visuelle ou contextuelle, et comment cette coopération se traduit en une conclusion. Le réseau **DSL**, formé de pondérations $\{\omega_{i,j}\}$, expose explicitement la manière dont la règle “ \mathcal{E}_{rule} ” tisse des liens avec des features visuelles ou auditives pour générer une reconnaissance de haut niveau. Il est en outre possible d’**extraire** des proto-règles depuis les clusters émergents dans le réseau : si l’on observe qu’un **cluster** demeure très stable, rassemblant certaines variables (par exemple, plusieurs marqueurs biologiques), un symptôme et un diagnostic, on peut reformuler une **règle émergente** indiquant que “lorsque X, Y, Z sont élevés, la synergie avec le diagnostic D augmente considérablement, ce qui équivaut à suspecter telle pathologie P.” Cette forme d’**abstraction** dérive naturellement de la dynamique d’auto-organisation du **DSL** et fournit une explication concise, en reliant directement le diagnostic à un sous-réseau stable de variables médicales fortement pondérées. On franchit ainsi une étape supplémentaire vers une **explicabilité** plus intuitive, où l’on ne se contente pas de mentionner le rôle de quelques features, mais où l’on propose une **règle** ou une **raison** qui s’apparente à un énoncé symbolique, ancré dans les liaisons sub-symboliques du réseau.

1.7.5.4. Points de Vigilance : Graphes Trop Grands, Hétérogénéité Massive

Dans un **Deep Synergy Learning** de forte dimension, la **visualisation** intégrale de toutes les liaisons $\omega_{i,j}$ ou l’exploration exhaustive des chemins reliant deux entités devient inenvisageable. La taille du graphe peut atteindre plusieurs milliers ou millions de nœuds et de liens, rendant toute inspection manuelle impossible. Afin de surmonter ces limites, il est utile de mettre en place des **mécanismes de filtrage** ou d’**agrégation**. Le filtrage consiste à écarter les liaisons dont la pondération $\omega_{i,j}$ reste en deçà d’un **seuil** minimal, ou qui ne participent pas de manière significative à un **cluster** pivot. Cette méthode abaisse drastiquement la densité du réseau et en facilite l’interprétation. L’agrégation, quant à elle, regroupe certaines entités en **macro-nœuds**, de sorte que l’on obtienne un **graphe résumé** nettement plus petit, faisant émerger les **clusters** importants tout en préservant la cohérence des liens principaux.

Lorsque les entités du DSL recouvrent une **hétérogénéité** massive, intégrant par exemple de la vision, de l’audio, du texte et des modules symboliques, la **compréhension humaine** impose de clarifier la correspondance entre chaque entité et les données originales. Un patch d’image doit être décrit par sa position ou sa nature dans la scène, un segment audio doit renvoyer à la phrase ou au moment précis de l’enregistrement, un concept symbolique doit être relié à son énoncé explicite. Cette **traduction** partielle s’avère cruciale pour délivrer une **explicabilité** satisfaisante : l’utilisateur comprend alors pourquoi telle liaison $\omega_{i,j}$ est forte, parce qu’un patch visuel repéré comme “zone X” s’avère synergiquement lié à un certain concept ou à un segment audio. Il est donc indispensable d’associer à chaque entité \mathcal{E}_i un ensemble de **méta-information**s décrivant son rôle, son lien avec l’input brut et sa typologie (vision, texte, règle symbolique). Grâce à ce balisage, la **mise en récit** du réseau DSL, même de grande taille, peut s’effectuer de façon plus synthétique et plus pertinente, en se concentrant sur un sous-ensemble restreint d’arêtes fortement pondérées et sur des macro-clusters résumant la structure globale.

Conclusion

La **structure** en entités et liaisons synergiques du **Deep Synergy Learning** confère un **avantage** potentiel pour l'**explicabilité** : on peut :

- **Identifier** des clusters,
- **Exhiber** des sentiers $\mathcal{E}_i \rightarrow \mathcal{E}_j$ hautement pondérés,
- **Mettre** en avant des règles symboliques si présentes,
- **Comparer** des liens forts entre un utilisateur/contenu (dans le cas de recommandation) ou un patient/diagnostic (dans le cas médical).

Cependant, l'**explosion** du nombre d'entités et de liens peut rendre difficile la simple “lecture” de la configuration globale. On doit alors s'équiper de **techniques** de filtrage, d'**agrégation** (macro-clusters), ou de **génération** d'explications locales (chemins courts). La cohabitation possible du **symbolique** et du **sub-symbolique** aide à concrétiser des **justifications** plus claires. Ainsi, le DSL peut jouer un rôle dans la mouvance d'**Explainable AI**, à condition de manier soigneusement la **complexité** et la **variété** des entités afin de délivrer une **vision** cohérente et compréhensible à l'utilisateur final.

1.7.6. Considérations Éthiques et Réglementaires

Au-delà des enjeux techniques (scalabilité, stabilité, explicabilité), le **Deep Synergy Learning (DSL)** soulève également des questions **éthiques** et **réglementaires**. En effet, l'auto-organisation des entités au sein d'un **réseau** qui évolue sans hiérarchie prédéfinie peut introduire une forme d'**imprévisibilité** ou de **décision distribuée**. Lorsqu'il s'agit de systèmes critiques (médical, financier, militaire, etc.), il est indispensable de **responsabiliser** la conception et l'usage de tels modèles. De même, dans des applications plus courantes (recommandation, transport, logistique), la manière dont les données sont exploitées et dont les clusters se forment peut produire des **biais** ou altérer la **transparence** vis-à-vis des utilisateurs. Cette sous-section (1.7.6) examine :

- Le **risque** de perte de contrôle ou d'incompréhension du comportement d'un système auto-organisé,
- Les **biais** possibles et la nécessité de garanties contre la discrimination ou l'iniquité,
- Les **règles** et normes (RGPD, réglementations sectorielles) exigeant traçabilité et protection des données,
- Le rôle de **valeurs humaines** (sécurité, dignité, respect de la vie privée) dans la conception de DSL responsables.

1.7.6.1. Perte de Contrôle et Responsabilités

L'un des attraits majeurs du **Deep Synergy Learning (DSL)** réside dans sa **dynamique** d'auto-organisation : les entités s'y regroupent en **clusters** suivant les synergies détectées, sans requérir un programme explicite gérant leurs interactions. Toutefois, cette souplesse peut engendrer des **comportements** difficiles à anticiper, dans la mesure où ni le concepteur, ni l'opérateur, ni le développeur n'ont directement programmé la structure finale. Dans des contextes où la **responsabilité** est cruciale, notamment en milieu médical ou bancaire, il importe de savoir **qui** assumera les conséquences si le système commet un choix discutable ou s'il engendre un dommage. Avec un réseau neuronal “classique”, déjà, l'**opacité** rend l'attribution de responsabilité délicate ; dans un DSL à auto-organisation plus distribuée, le degré de complexité est plus élevé encore.

Pour pallier cette difficulté, il est envisageable de munir le système d'un **journal** d'événements et de mises à jour, permettant de **tracer** l'historique de la pondération $\omega_{i,j}$. Théoriquement, on pourrait reconstituer le “chemin causal”

entre une observation et une décision, même au sein d'un graphe évolutif. Cette traçabilité demeure toutefois ardue à mettre en œuvre si l'on veut documenter en continu un réseau de grande dimension.

La loi ou les **régulations** sectorielles (ex. aéronautique, médical, bancaire) imposent souvent de **certifier** une partie du comportement du système avant son déploiement. Un DSL auto-organisé risque de “**changer**” de configuration quasi à chaque itération, rendant hasardeuse toute garantie statique. Il existe alors plusieurs stratégies. Il est possible de borner la **vitesse** d'évolution (en limitant le produit $\eta \times \tau$, ou en imposant un pas maximal) pour qu'il demeure un **voisinage** de configurations déjà validées. On peut aussi intégrer un **module** de supervision hiérarchique (voir la section 1.7.4.3) qui invalide certaines reconfigurations jugées non conformes à des normes ou des plans de sûreté. Enfin, il est envisageable d'appliquer des **tests** de robustesse, ou “stress tests”, simulant divers scénarios de données pour s'assurer qu'aucune configuration délétère ne surgisse, ou du moins qu'elle reste suffisamment rare et détectable.

Ces considérations reflètent un compromis entre la **liberté** synergique, qui fait la force conceptuelle du DSL, et la **maîtrise** requise dans les secteurs critiques. Les débats sur la responsabilité sont dès lors analogues à ceux suscités par les réseaux neuronaux opacifiés, mais se complexifient en raison de la reconfiguration continue du réseau. Les utilisateurs, exploitants et autorités de tutelle doivent tenir compte de la difficulté d'isoler un instantané du DSL pour en vérifier la validité et assumer les conséquences de ses décisions évolutives.

1.7.6.2. Biais, Équité et Discrimination

Dans un **Deep Synergy Learning (DSL)** appliqué à des tâches de recommandation ou de décision, il arrive que la dynamique de formation des **clusters** reflète des **biais** préexistants dans les données, ce qui peut engendrer des effets discriminants. Imaginons un système de **scoring** pour l'attribution de crédits bancaires. Si les bases de données d'emprunteurs présentent une corrélation injustifiée entre certaines variables socio-économiques et la capacité de remboursement, le **DSL** risque de regrouper un certain sous-groupe dans un **cluster** jugé risqué, ou de l'isoler dans un espace périphérique, renforçant une forme d'exclusion. Ces liaisons faibles ou inexistantes maintiennent le sous-groupe à l'écart des opportunités, ce qui peut amplifier la ségrégation : moins on l'autorise à coopérer avec le reste du réseau, plus sa synergie chute, et moins il bénéficie d'un accès équitable au crédit.

Ce phénomène d'**auto-renforcement** est potentiellement dangereux. Dans un simple réseau neuronal supervisé, des régulations ou des contraintes peuvent être imposées pour limiter l'exploitation de variables sensibles. Mais dans un **DSL**, la situation est plus complexe, car les **entités** se lient ou se séparent selon la synergie globale perçue. Un sous-groupe minoritaire, s'il n'est pas bien représenté dans les données, peut se voir marginalisé. Il convient alors d'établir des **mécanismes** préventifs ou correctifs. Il est envisageable, par exemple, de **pénaliser** l'isolation injustifiée d'un sous-groupe, en introduisant un terme additionnel dans la fonction d'énergie ou dans l'équation de mise à jour des pondérations. Ce terme peut imposer qu'un certain ensemble d'entités (associées à un attribut protégé) ne soit pas trop “distant” du reste ou qu'un cluster ne se forme pas exclusivement autour d'une caractéristique stigmatisante. Un autre moyen consiste à **rééquilibrer** les données avant l'apprentissage, afin de fournir au DSL une image plus fidèle et plus inclusive de la population, ou à injecter artificiellement des entités compensatoires, assurant que le réseau aura de quoi évaluer la synergie de manière équitable.

Dans un cadre légal, des obligations comme le **RGPD** (Règlement Général sur la Protection des Données) ou les lois anti-discrimination exigent que la décision ne dépende pas d'informations relatives à l'origine ethnique, à la religion ou à d'autres critères protégés. Le **DSL**, en créant des entités et des liaisons auto-organisées, doit donc s'assurer que certaines variables n'encoderont pas implicitement ces attributs sensibles, sous peine d'aboutir à des comportements discriminants difficilement décelables. Il est crucial de mettre en place des **audits** récurrents du réseau pour vérifier la distribution des **clusters**, observer si un groupe spécifique demeure systématiquement isolé ou sous-pondéré et, le cas échéant, forcer une mise à jour corrective. De tels audits peuvent se fonder sur l'inspection des pondérations $\{\omega_{i,j}\}$ associées à un ensemble d'entités. Ils peuvent également imposer une mesure d'équité en tant que contrainte externe, qui intervient de façon hiérarchique pour éviter la constitution de partitions inéquitables. L'objectif demeure le **contrôle** du **DSL** tout en respectant l'esprit d'**auto-organisation**, afin de combiner l'efficacité de la synergie et la nécessité de préserver une **égalité** ou une **justice** dans les décisions rendues.

1.7.6.3. Protection des Données et Droit à l'Oubli

Dans une architecture **Deep Synergy Learning (DSL)**, il arrive fréquemment que le réseau comprenne des **entités** liées à des personnes (utilisateurs, patients). Les législations comme le **RGPD** (Règlement Général sur la Protection des Données) en Europe imposent plusieurs obligations, dont la **minimisation** des données recueillies (ne conserver que l'essentiel), le **droit à l'oubli** (un individu peut exiger la suppression de ses données personnelles) et des règles d'**anonymisation** ou de pseudonymisation pour éviter la ré-identification. Le réseau DSL, en tant que graphe auto-organisé et évolutif, doit donc intégrer des procédures garantissant que l'on puisse effacer ou rendre inopérante l'entité \mathcal{E}_i correspondant à une personne qui souhaite être oubliée.

La présence d'une entité \mathcal{E}_i dans un **Synergistic Connection Network** peut avoir des répercussions à de multiples endroits, car l'**auto-organisation** a pu former plusieurs **clusters** où \mathcal{E}_i joue un rôle. Pour satisfaire le droit à l'oubli, il faut d'abord **retracer** cette entité dans le graphe, puis la **retirer** du réseau ou la rendre totalement anonyme. Il convient de supprimer ou de neutraliser les liaisons $\omega_{i,j}$ reliées à l'entité \mathcal{E}_i . Lorsque l'on procède à cette suppression, on doit parfois **réajuster** la **synergie** locale afin que les entités \mathcal{E}_j anciennement liées à \mathcal{E}_i ne se retrouvent pas avec des pondérations invalides. L'opération de "déréférencement" peut donc influer sur l'équation d'évolution, qui doit détecter et ignorer toute trace associée à \mathcal{E}_i . L'objectif est d'éviter que des **résidus** dans la structure du réseau ne permettent de re-identifier l'utilisateur a posteriori, contrevenant ainsi aux principes de confidentialité exigés par la loi.

Le RGPD recommande également la **pseudonymisation** et la **minimisation** des informations stockées, ce qui affecte la façon dont les entités DSL sont construites. Il est parfois nécessaire de ne stocker, dans une entité \mathcal{E}_i , qu'une version agrégée (par exemple, localisation au niveau d'une ville plutôt que d'une rue) et d'exclure tout identifiant direct pour se conformer aux dispositions légales. Si le réseau DSL garde un trop haut niveau de précision, il peut être considéré comme non conforme à la réglementation. Une possibilité consiste à mettre en place un **processus** d'anonymisation périodique : après un certain temps, on fusionne des entités trop granulaires ou on supprime les éléments de détail, en sacrifiant une partie de la granularité tout en conservant l'essence des **synergies** globales. Cette pratique garantit une protection de la vie privée tout en permettant au DSL de continuer à fonctionner de manière auto-organisée et réactive, mais sur la base de données moins sensibles ou déjà anonymisées.

1.7.6.4. Transparence, Explicabilité et Contrôle

Dans certains contextes réglementés, comme le **RGPD** ou des lois spécifiques à un domaine (médical, bancaire, etc.), les usagers ou patients ont la faculté d'exiger une **explication** sur la décision qui les concerne. Le **Deep Synergy Learning (DSL)**, grâce à ses **clusters** et à ses **liaisons** plus faciles à interpréter qu'un pipeline neuronal opaque, peut répondre à cette exigence. Il demeure toutefois nécessaire de s'assurer que l'évolution du réseau (les pondérations $\omega_{i,j}$ dans le Synergistic Connection Network) demeure **traçable** dans le temps. Pour cela, on peut tenir un **historique** des principales mises à jour ou conserver des **snapshots** à intervalles réguliers, afin de justifier a posteriori comment tel ou tel sous-réseau s'est formé.

Une **infrastructure** d'explication (voir section 1.7.5) doit de surcroît synthétiser le résultat de la **dynamique** auto-organisée. Il ne s'agit pas nécessairement d'exposer l'entièreté des liens $\omega_{i,j}$, mais d'extraire des **chemins** ou des **clusters** pivot permettant de reconstituer le **raisonnement local**. Une difficulté survient lorsque le système évolue rapidement, voire "chaotiquement" : dans ce cas, l'explication risque de changer sensiblement d'une itération à l'autre, ce qui nuit à la cohérence d'ensemble. Des **mécanismes** de stabilisation ou de **pas** d'évolution limités (contrôle de η et de τ , par exemple) peuvent aider à préserver une relative continuité, rendant les explications plus consistantes dans le temps.

Pour de nombreuses **applications** sensibles, la législation ou la pratique courante impose un **contrôle humain** dans la boucle. Conformément à un principe d'audit ou de validation, un expert (médecin, analyste, manager) examine la configuration actuelle du DSL et doit pouvoir **accepter**, **refuser** ou **rectifier** certaines décisions. L'**interface** qui présente les clusters, leurs pondérations dominantes et l'historique succinct de leur formation doit donc être la plus claire possible, en hiérarchisant l'information. L'expert peut alors intervenir en imposant une pénalisation sur des liaisons considérées comme non conformes (sécurité, éthique, cohérence métier). Cette démarche articule la liberté

auto-organisée du DSL (qui adapte continûment ses liaisons) avec un **pilotage humain**, assurant un respect des **cadres légaux** et éthiques, tout en tirant parti de la plasticité et de l'auto-organisation qu'offre le Deep Synergy Learning.

Conclusion

Les **considérations éthiques et réglementaires** forment un **pilier** incontournable lors du déploiement de **systèmes auto-organisés** comme le DSL. On peut résumer les **grandes lignes** de vigilance :

Responsabilité et traçabilité : éviter qu'un système auto-adaptatif devienne totalement opaque quant à ses décisions ou conséquences, mettre en place un journal d'événements, un module de supervision.

Biais et équité : surveiller la formation de clusters discriminants, adapter la dynamique pour prévenir les exclusions ou dominations injustes.

Protection des données : garantir l'anonymisation, la pseudonymisation, et le droit à l'oubli dans un graphe évolutif (ce qui n'est pas trivial techniquement).

Transparence et droit à l'explication : mettre en place des mécanismes d'explicabilité (voir 1.7.5) assurant que l'utilisateur comprend la logique du DSL.

Au final, l'**originalité** du DSL ne saurait dispenser d'une **responsabilisation** solide. La recherche sur l'IA dite "responsable" ou "de confiance" s'applique tout autant (voire davantage) à un modèle **auto-organisé** qui, par définition, échappe aux schémas de contrôle univoque. Les prochaines sections (et développements futurs) devront approfondir les outils et protocoles pour faire du DSL une **technologie** à la fois puissante, adaptable, et conforme aux **valeurs et règles** en vigueur.

1.7.7. Comparaisons Expérimentales avec d'Autres Approches

Dans les sections précédentes (1.7.1 à 1.7.6), nous avons exploré les **défis** et **contraintes** du **Deep Synergy Learning (DSL)** en matière de **complexité**, de **qualité des données**, de **stabilité**, d'**explicabilité**, et d'**éthique**. Pour affermir la **pertinence** de ce paradigme sur le terrain, il est indispensable d'effectuer des **comparaisons expérimentales** rigoureuses avec des méthodes existantes : réseaux neuronaux profonds (CNN, RNN, Transformers), méthodes de clustering (k-means, DBSCAN), algorithmes d'optimisation (ordonnancement, etc.), voire des approches neuro-symboliques plus classiques. Cette sous-section (1.7.7) aborde :

Les **critères** de comparaison essentiels (performances quantitatives, robustesse, adaptativité...),

Les **protocoles** expérimentaux pour confronter le DSL à d'autres techniques (tests sur données stationnaires et non stationnaires, analyses de complexité, etc.),

Les **scénarios** où le DSL a un net avantage (auto-organisation, peu de supervision, scénarios évolutifs),

Les **limites** et enseignements que de tels benchmarks peuvent révéler, guidant l'amélioration du DSL.

1.7.7.1. Choisir les Critères de Comparaison

Dans la plupart des tâches d'apprentissage ou de décision (vision, audio, recommandation, diagnostic médical), on évalue la **précision** ou des métriques dérivées comme l'**accuracy**, la **F1-score** ou l'**AUC**. Lorsque l'on souhaite comparer un **Deep Synergy Learning (DSL)** avec, par exemple, un CNN ou un Transformer, on peut se borner à mesurer son **taux de reconnaissance** (dans un scénario de classification d'images) ou la **sensibilité** et la **spécificité** (dans le cadre d'un diagnostic médical). Toutefois, ces **scores** globaux ne reflètent pas nécessairement la **richesse** de la structure auto-organisée du DSL, qui excelle par sa **plasticité** et son adaptation continue, sans exiger nécessairement

de labels massifs. Il est donc essentiel, lors de l'évaluation, d'incorporer des critères tenant compte de la **dynamique** de l'auto-organisation.

Le **DSL** se veut **adaptatif** en présence de bruit, de pannes de capteurs ou de changements de distribution (concept drift). Pour mettre en évidence cette capacité, il est concevable de concevoir des expériences où la distribution des données varie progressivement ou subitement en cours d'apprentissage. Dans un cas classique, un **réseau neuronal** (CNN, RNN) va souvent exiger une **séance de re-fine-tuning** global, tandis que le DSL doit pouvoir **réajuster** localement ses liaisons $\{\omega_{i,j}\}$. On peut alors comparer la **vitesse de réadaptation** et la **qualité** de la reprise de performance. On mesure, par exemple, la **chute** temporaire de précision lorsqu'un nouveau type de données apparaît, ainsi que le **temps** qu'il faut au modèle pour retrouver un niveau satisfaisant. Le DSL a pour atout de pouvoir réorganiser ses clusters et de créer ou rompre des liens en fonction des synergies locales, ce qui peut se révéler plus rapide et moins coûteux qu'un re-entraînement complet.

La **taille** du modèle (nombre de paramètres ou de liaisons actives) et la **charge** de calcul (temps CPU, mémoire GPU) comptent également parmi les indicateurs déterminants. Un DSL peut rester **parcimonieux** s'il opère une **sparsification** rigoureuse de la matrice ω . Si, au contraire, on ne limite pas la prolifération des liens, la structure risque de gonfler et de devenir onéreuse en ressources. L'évaluation doit donc englober la **scalabilité** et l'**efficience** : on vérifie si, pour un grand nombre d'entités, le DSL parvient à maintenir une densité de liens gérable (voir section 1.7.1) et à répondre aux scénarios évolutifs (pannes, modifications de distribution) avec des coûts de mise à jour et de mémoire acceptables.

1.7.7.2. Protocoles Expérimentaux Envisageables

Pour **évaluer** la pertinence d'un **Deep Synergy Learning (DSL)** dans des scénarios variés, on peut le confronter à des bases de données classiques ou à des cas pratiques bien établis :

- Des ensembles **d'images** (MNIST, CIFAR, éventuellement un sous-ensemble d'ImageNet) pour évaluer la classification ou la détection, en analysant la **précision** obtenue par l'auto-organisation visuelle des entités (patchs, features, etc.) face à un CNN ou un Transformer.
- Des ensembles **audio** (LibriSpeech, UrbanSound8K) pour tester la reconnaissance de sons ou la transcription de parole, où le DSL formerait des entités "segment audio" et "caractéristiques acoustiques", comparées à un système RNN ou CNN entraîné de façon traditionnelle.
- Des **bases de recommandation** (MovieLens, Amazon reviews) pour mesurer la qualité des suggestions formulées par le DSL ; on confronte alors la structure auto-organisée (clusters d'utilisateurs, de contenus, etc.) aux méthodes de factorisation matricielle ou aux modèles de collaboration neuronaux, en examinant le **Recall@K**, la **Precision@K** ou la **NDCG**.
- Des **données médicales** (ensembles de pathologies, EHR) afin de juger la capacité du DSL à découvrir des clusters pathologiques ou à émettre des hypothèses de diagnostic, comparativement à un modèle supervisé (réseau profond, arbre d'ensemble) ou semi-supervisé.
- Des **scénarios robotiques** (simulateurs comme Gazebo, PyBullet, ou environnement multi-agents) pour vérifier la coordination sensorimotrice : un DSL peut relier capteurs et effecteurs et montrer sa robustesse dans la détection de pannes ou l'évolution de conditions.

Dans chacun de ces domaines, on confronte les **résultats** finaux (taux de reconnaissance, score de recommandation, F1-score diagnostique, etc.) à ceux de modèles **classiques** : CNN, RNN, Transformer, clustering K-means, factorisation matricielle, etc. On veille aussi à mesurer la **taille** du modèle (paramètres, liaisons), la **complexité** de calcul et la **vitesse d'inférence**.

Le **DSL** se veut également **adaptatif** en cas de changements de distribution (concept drift), de pannes de capteur, de conditions altérées. Pour l'exercer à cette fin, il est utile de mettre en œuvre des **scénarios** où les données subissent un glissement graduel ou une modification brutale (apparition de nouvelles classes, hausse du bruit, changement d'éclairage, etc.). Sur la timeline $\{t_0, \dots, t_f\}$, on évalue :

37. La manière dont un réseau neuronal classique (CNN, RNN) gère ce drift (souvent par un re-fine-tuning ou un forget partiel).
38. La façon dont le DSL, sans réapprentissage massif, modifie localement ses liaisons $\omega_{i,j}(t)$.

On observe la **performance** en continu (taux de classification, AUC, indice de recommandation) et on étudie la **vitesse** de réadaptation. Le **DSL** devrait, en principe, reporter des ajustements plus localisés et moins coûteux qu'un entraînement total, tout en préservant une cohérence globale. Ceci permet de vérifier empiriquement que la **robustesse** et la **plasticité** conférées par la mise à jour distribuée des pondérations ω confèrent un avantage aux méthodes d'auto-organisation, surtout dans des domaines où les données ou le contexte varient régulièrement.

1.7.7.3. Scénarios où le DSL a un Avantage Notable

Le **Deep Synergy Learning (DSL)** se distingue particulièrement dans des environnements où l'on dispose de **peu ou pas de labels** (section 1.5.5) ou bien où la supervision est coûteuse et incomplète. En mode **non supervisé** ou faiblement supervisé, un réseau neuronal classique (tel qu'un CNN) peine à extraire des structures cohérentes, car il manque d'exemples étiquetés pour guider l'apprentissage. Le **DSL**, au contraire, s'appuie sur sa **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pour **auto-organiser** les entités, sans nécessiter un volumineux jeu de labels. Les **clusters** émergent alors de la simple coopération (ou co-occurrence) observée entre entités, ce qui rend le réseau particulièrement apte aux tâches d'**exploration** et de **regroupement** de données hétérogènes.

Un autre atout du DSL s'observe en **multi-modalité hétérogène** (section 1.5.2), où l'on combine image, audio, texte, voire des flux sensoriels divers. Les approches traditionnelles imposent souvent un **pipeline** de fusion rigide (concaténation tardive, attention inter-modale) dépendant de choix d'architecture prédéfinis. Le DSL adopte une démarche plus flexible : chaque entité \mathcal{E}_i (visuelle, auditive, etc.) forme des **liaisons** $\omega_{i,j}$ spontanées selon la synergie perçue. Cette structure évite la planification d'un module de "fusion" spécifique ; la mise en réseau est régie par la **dynamique** locale des pondérations, autorisant des clusters multimodaux à émerger naturellement.

Lorsque les **flux** de données sont **dynamiques** et sujets à des **pannes** ou à des transformations (sections 1.5.4 et 1.7.2.3), le DSL illustre également sa supériorité. Un capteur défectueux voit ses liaisons ω chuter, car la synergie avec le reste du système devient faible. Le réseau se réorganise alors sans qu'un nouvel entraînement global ne soit requis, contrairement à un réseau neuronal classique où une modification durable de la distribution (panne de capteur, concept drift) nécessite un **fine-tuning** ou un recalibrage important.

L'**explicabilité** du DSL revêt aussi un intérêt clé (sections 1.7.5 et 1.5.6). Les **clusters** ou les chemins de **synergie** identifiés entre entités procurent des explications plus détaillées qu'une simple "heatmap" d'attention. Dans un réseau traditionnel, la structure interne (couches, neurones) reste souvent opaque, malgré certaines techniques de visualisation de filtres. Le DSL, en attribuant des **identités** distinctes à ses nœuds et en laissant les liens $\omega_{i,j}$ témoigner de leur degré de coopération, permet une **compréhension** plus directe de la raison pour laquelle tel sous-réseau est né, ou tel couple $(\mathcal{E}_i, \mathcal{E}_j)$ a vu sa liaison renforcée.

Dans l'ensemble, une **comparaison expérimentale** (section 1.7.7.2) révélera que, face à un CNN ou à un Transformer, le DSL excelle dans des conditions de **data faible** ou de **distribution changeante**, ou encore quand l'on souhaite un **rendement** sémantique mieux explicité par un graphe de clusters. Ces propriétés séduisent dans des scénarios où l'on doit adapter le système en continu, respecter des contraintes d'équité, ou justifier les décisions auprès d'utilisateurs ou d'experts.

1.7.7.4. Limites et Enseignements Possibles

Lors de l'évaluation d'un **Deep Synergy Learning (DSL)** à large échelle, on peut constater que la structure du réseau, qui autorise jusqu'à $O(n^2)$ liaisons dans le cas binaire, devient difficile à manipuler si les entités sont très nombreuses. Il peut être nécessaire d'imposer une **sparsification** stricte (par exemple, ne maintenir qu'un voisinage restreint pour chaque entité) ou de recourir à un **algorithme local** qui ne calcule pas la synergie entre toutes les paires d'entités, mais seulement entre celles jugées proches dans un espace de représentation. Les expérimentations comparatives, que ce

soit en temps de calcul ou en mémoire GPU, permettent d'identifier le **seuil** au-delà duquel le DSL se montre moins performant que d'autres architectures classiques (CNN, RNN, Transformer) optimisées pour le passage à l'échelle sur des supports matériels massivement parallèles.

Par ailleurs, si l'on dispose d'un **jeu de données étiquetées** gigantesque (par exemple la totalité d'ImageNet, avec plus de 14 millions d'images), un réseau supervisé standard tel qu'un CNN ou un Transformer minutieusement entraîné peut atteindre d'excellentes performances de classification top-n. Le **DSL**, plus orienté vers l'**auto-organisation** non supervisée ou faiblement supervisée, pourrait dans ce cas demeurer en retrait si l'on n'exploite pas pleinement toute la supervision à disposition. On peut envisager des versions supervisées ou semi-supervisées du DSL où la synergie est partiellement guidée par des labels, mais la question demeure ouverte : ces labels suffiront-ils à faire concurrence à un CNN entraîné par descente de gradient pure, surtout lorsque le volume d'annotations est massif ? Ces expériences permettront de clarifier dans quelle mesure le DSL parvient à tirer parti d'une supervision large sans renoncer à sa plasticité et à sa modularité intrinsèque.

Ces limites et enseignements soulignent la nécessité de concevoir des **protocoles hybrides**, dans lesquels on combine la liberté d'auto-organisation du DSL (favorable à l'adaptation continue ou à la gestion de scénarios peu labellisés) avec la force d'un apprentissage supervisé classique lorsqu'un dataset étiqueté d'ampleur est disponible. C'est dans cet espace de compromis que réside probablement l'avenir des approches synergiques à l'échelle industrielle.

Conclusion

La **validation expérimentale** du **DSL** face aux approches dominantes (réseaux neuronaux profonds, clustering, méthodes d'optimisation) est incontournable pour :

- **Quantifier** la performance, la robustesse, la capacité d'adaptation,
- **Identifier** les atouts (apprentissage sans label massif, gestion du drift, multimodalité hétérogène),
- **Révéler** les limites (coût en calcul, besoin de mesures de synergie fiables, difficulté sur des tâches supervisées massives).

Des **benchmarks** variés (vision, audio, recommandation, diagnostic, robotique) permettront de **cartographier** où le DSL s'impose et où il doit s'allier à d'autres techniques (CNN pré-entraîné pour extraire des features, par exemple). Au final, ce travail de confrontation expérimentale, mené en parallèle des réflexions éthiques (1.7.6) et de la recherche sur l'optimisation (1.7.3), consolide la position du DSL comme un **paradigme** original — potentiellement complémentaire, voire compétitif — dans le paysage actuel de l'apprentissage automatique.

1.8. Positionnement du DSL dans l'Évolution de l'IA

Après avoir étudié les fondements du **Deep Synergy Learning (DSL)** et ses multiples applications (sections 1.1 à 1.6), ainsi que les défis et contraintes (section 1.7), il est opportun de s'interroger sur la **place** qu'occupe le DSL dans l'**évolution** de l'intelligence artificielle. En effet, l'IA contemporaine se décline principalement en deux grands paradigmes : d'un côté, l'**IA sub-symbolique** (réseaux neuronaux, méthodes statistiques) qui a connu un essor spectaculaire grâce à l'apprentissage profond ; de l'autre, l'**IA symbolique**, héritée des systèmes experts et de la logique formelle, plus aisée à interpréter et à manipuler comme un langage de haut niveau.

Le **DSL**, en proposant une **auto-organisation** et une **dynamique** des liens fondée sur la synergie, apporte un **paradigme** susceptible de **réconcilier** certains points de friction entre ces approches. On peut se demander :

S'il offre un **point** de convergence entre l'**IA symbolique** et l'**IA sub-symbolique**,

Comment il peut **coexister** ou se **substituer** à l'apprentissage profond,

Dans quelle mesure le **renforcement** (RL) ou la **logique** se fondent dans une trame DSL,

Le **rôle** d'éléments comme la **mémoire** et l'**attention** dans une architecture auto-organisée,

Les **tendances** futures, notamment vers une IA plus “forte” ou consciente,

L'**impact** interdisciplinaire, en particulier avec les neurosciences,

Et, de manière générale, comment le DSL s'inscrit dans la **progression** de l'IA moderne et ce qu'il peut apporter de nouveau.

Cette section (1.8) aborde ainsi le **positionnement** du DSL dans le paysage AI actuel et futur, regroupant sept sous-points :

IA Symbolique vs IA Sub-symbolique : Intégration Potentielle (1.8.1)

DSL et Apprentissage Profond : Collaboration ou Substitution ? (1.8.2)

Approches Hybrides : DSL, RL (Reinforcement Learning) et Logique (1.8.3)

Rôle de la Mémoire et de l'Attention dans le DSL (1.8.4)

Tendances Futures : Vers une IA Forte, Consciente ? (1.8.5)

Effet Sur la Recherche Interdisciplinaire (1.8.6)

Exemple de Convergence : DSL & Neurosciences (1.8.7)

Nous commencerons par revisiter (1.8.1) la vieille dichotomie IA symbolique vs sub-symbolique, pour voir en quoi le **DSL** pourrait établir des passerelles novatrices.

1.8.1. IA Symbolique vs IA Sub-symbolique : Intégration Potentielle

L'**histoire** de l'IA est souvent présentée comme marquée par un **dualisme** :

- L'**IA symbolique**, qui manipule des règles, des faits, des ontologies dans un langage logique ou pseudo-logique (ex. Prolog, systèmes experts). Elle est réputée plus “interprétable” et plus “rigoureuse” dans le raisonnement, mais manque de flexibilité face aux données massives et bruitées.

- L'IA **sub-symbolique**, typiquement les réseaux neuronaux et méthodes statistiques, qui excellent en **apprentissage** (vision, NLP, etc.) en profitant de nombreux exemples, mais peinent à manipuler des structures conceptuelles complexes ou des règles explicites.

1.8.1.1. Vers une IA Hybride ou Neuro-Symbolique

Au cours de la dernière décennie, un nombre croissant de recherches s'est attelé à **combiner** les approches symboliques (issues de la logique, des règles ou des ontologies) et les méthodes sub-symboliques (neuronales ou connexionnistes). Les travaux dits **neuro-symboliques** cherchent, par exemple, à implanter des **règles logiques** au sein d'un réseau neuronal ou, à l'inverse, à extraire des **règles symboliques** depuis les *features* d'un modèle déjà entraîné. L'objectif est double : tirer parti de la **puissance** des méthodes sub-symboliques, capables de modéliser de vastes volumes de données complexes, et bénéficier de la **lisibilité** ou de la **rigueur** propre à la logique et aux concepts formels pour la partie raisonnement et explication.

Ces approches neuro-symboliques s'organisent souvent sous la forme de **systèmes hybrides**, où l'on place côté à côté une **composante** consacrée au traitement sub-symbolique (réseau neuronal) et une **composante** symbolique (règles ou moteur logique). Un **pont** dédié est alors chargé de faire correspondre ces deux volets, ce qui peut induire une architecture "collée" au sens où la dimension symbolique reste nettement délimitée de la dimension connexionniste. La **véritable** intégration, où le réseau lui-même serait libre d'accueillir à la fois des **concept logiques** et des **signaux bruts** en adaptant sa topologie, demeure un défi qui n'est pas entièrement résolu dans la plupart de ces schémas.

Le **Deep Synergy Learning (DSL)** entend précisément répondre à cette problématique, car il propose une **dynamique** où la structure se reconfigure **auto-organisationnellement**, sans imposer de cloison fixe entre l'aspect symbolique et l'aspect sub-symbolique. Les **entités** \mathcal{E}_i peuvent incarner aussi bien un **flux sensoriel** (image, audio), qu'un **concept symbolique**, ou une **règle**. Les liaisons $\omega_{i,j}$ se créent ou se coupent selon la **synergie** détectée. Un **nœud** représentant une règle logique pourra donc se lier plus fortement à un patch d'image ou un module audio dès lors qu'il y trouve un **gain** mutuel. De cette manière, la frontière traditionnelle entre le bloc **symbolique** et le bloc **neuronale** cède la place à un **réseau** unifié, où chaque composante est traitée comme une entité à part entière, susceptible de coopérer librement pour faire émerger des **clusters** ou des **schémas** de raisonnement. Ce fonctionnement souligne l'ambition du DSL de faciliter la rencontre fluide entre la **rigueur** des modèles logiques et la **flexibilité** d'apprentissage des méthodes sub-symboliques.

1.8.1.2. Rôle du DSL : Auto-Organisation Unifiant Symbolique et Sub-Symbolique

Le **Deep Synergy Learning (DSL)** propose une **intégration** fluide entre les entités **sub-symboliques** (capteurs, features, embeddings neuronaux) et les **composantes symboliques** (règles, concepts logiques, ontologies). Dans la dynamique du DSL, les **liaisons** $\omega_{i,j}$ se créent ou se rompent en fonction de la **synergie** perçue (voir la section 1.4.5 pour la mise à jour itérative). Au lieu de séparer la logique dans un module détaché, on traite chaque **règle** ou **concept** comme un **nœud** $\mathcal{E}_k^{(symb)}$ dans le **Synergistic Connection Network**. La coopération entre une règle symbolique $\mathcal{E}_k^{(symb)}$ et une entité sub-symbolique \mathcal{E}_i (par exemple un capteur) s'exprime au moyen d'une pondération $\omega_{symb,i}$, qui évolue en fonction de la synergie : si la règle s'applique souvent à l'information fournie par ce capteur, la liaison se consolide ; sinon, elle s'étiole.

Considérons un **environnement** robotique où sont définies plusieurs **règles** logiques, notées \mathcal{R}_k . Un exemple pourrait être :
"Si tension(moteur) > 5V ET température(moteur) > 80°, alors générer une alerte de surchauffe."

Dans le **DSL**, une telle règle se modélise par une **entité** $\mathcal{E}_k^{(symb)}$. Parallèlement, on dispose de diverses **entités sub-symboliques** représentant les capteurs (tension, température, etc.). La **synergie** $\omega_{\text{tension,règle}}$ ou $\omega_{\text{temp,règle}}$ augmente si l'on observe une co-occurrence récurrente entre des relevés de tension haute et de température élevée, et la vérification de la condition de surchauffe. Sans nécessiter d'interface imposée, le réseau forme un **micro-réseau** localisé reliant la règle (entité symbolique) aux capteurs sub-symboliques concernés. Si la synergie reste positive dans le temps, un **cluster** se stabilise, associant la règle "surchauffe moteur" et les données sensorimotrices qui la confirment.

Cette approche évite le recours à une **passerelle** rigide entre la logique formelle et les signaux bruts. La topologie globale du **DSL** se **reconfigure** suivant la pertinence constatée : si la règle \mathcal{R}_k perd son utilité ou si de nouveaux

capteurs plus adaptés apparaissent, les pondérations se rééquilibrent en conséquence. On obtient ainsi une **auto-organisation** où la dimension **symbolique** est pleinement intégrée à la dynamique décrite en 1.4.5, donnant un **substrat** unifié pour l'IA hybride.

1.8.1.3. Avantages et Limites

Le fait d'intégrer la **dimension symbolique** directement dans un **Deep Synergy Learning (DSL)** offre divers **avantages**. D'abord, la logique se trouve incorporée à la **même** architecture auto-organisée : il n'est plus nécessaire de juxtaposer un module de **règles** indépendant et un module **sub-symbolique** (tel qu'un réseau neuronal). Au sein du DSL, chaque règle \mathcal{R}_k devient une entité $\mathcal{E}_k^{(\text{symb})}$, et la **pondération** $\omega_{\text{symb}, i}$ traduit la coopération effective entre cette règle et n'importe quelle entité sub-symbolique \mathcal{E}_i . Cette façon de procéder rend possible l'**apprentissage continu** de l'utilité de la règle, puisque la pondération se régule en fonction de la "valeur ajoutée" détectée. Un autre **bénéfice** apparaît en termes d'**explicabilité** : il devient aisément de repérer un **cluster** liant une règle \mathcal{R}_k à quelques entités perceptives $\mathcal{E}_p, \mathcal{E}_q, \dots$ justifiant une décision. On obtient dès lors une **transparence** plus claire qu'avec un système classiquement scindé où la logique reste à part.

Cependant, cette intégration comporte également des **limites** notables. Une première difficulté tient à la **quantification** de la synergie $S(\mathcal{E}_i, \mathcal{E}_k^{(\text{symb})})$ quand l'une des entités représente une **règle** ou un **concept** logique. Il s'agit de définir mathématiquement la notion de "gain" associé à l'activation conjointe d'une règle symbolique et d'un flux sub-symbolique. Ce n'est pas toujours trivial, surtout si la règle mélange des conditions logiques complexes. Par ailleurs, plus on introduit de **règles**, plus la combinatoire peut **explorer**, le réseau devant gérer de multiples $\omega_{(\text{symb}), i}$. La mise à jour itérative de ces liaisons doit alors tenir compte des possibles **contradictions** entre règles (ex. deux règles incompatibles ne sauraient être activées simultanément), ce qui alourdit la dynamique ω et nécessite des **mécanismes** de gestion de conflits ou de contradictions logiques.

Ces considérations exigent une **régulation** attentive, par exemple via des **termes** de parcimonie (limitation du nombre total de liaisons actives), des **pénalités** imposées aux liaisons incompatibles, ou un **contrôle** de la vitesse de croissance de certaines pondérations. Un tel encadrement évite l'explosion combinatoire de la structure et aide à résoudre de manière cohérente les potentielles contradictions entre règles symboliques. Malgré ces contraintes, la promesse demeure : le **DSL** fournit un cadre plus **organique**, capable de faire coexister de façon adaptative les **concepts** logiques et les **descripteurs** sub-symboliques.

1.8.1.4. Perspectives sur l'Intégration Totale

Dans les modèles traditionnels, on observe une **dichotomie** marquée : l'IA symbolique s'appuie sur des règles logiques et la manipulation de symboles, alors que l'IA sub-symbolique (réseaux neuronaux, méthodes connexionnistes) traite les données brutes par apprentissage statistique. Le **Deep Synergy Learning (DSL)** aspire à dépasser cette fragmentation en poussant l'**auto-organisation** à un niveau plus global. Les **entités symboliques** – vues comme de véritables "super-neurones" capables d'inférences logiques internes – coexistent avec des **entités sub-symboliques** qui extraient, à partir de données brutes, des features ou des embeddings. Toutes ces entités demeurent dans un **même** Synergistic Connection Network, au sein duquel les liens $\omega_{i,j}$ se créent, se renforcent ou s'affaiblissent en fonction de la synergie effectivement observée.

Lorsqu'une **règle symbolique** apporte un "gain" (par exemple, elle s'active souvent conjointement à un certain pattern perceptif), la liaison entre cette règle et les entités perceptives impliquées croît. Inversement, si une règle n'est presque jamais appliquée ou se trouve contredite par les observations, ses liens se dissolvent. De la même manière, certains **macro-clusters** voient le jour, rassemblant des **concepts** sémantiques et des **patches** perceptuels (image, audio, etc.) autour d'une **synergie** élevée. Un tel réseau évite ainsi d'imposer un **cloisonnement** net entre la partie "symbolique" et la partie "sub-symbolique" : les deux cohabitent et interagissent librement, n'apparaissant distinctement qu'en tant qu'entités aux rôles différents.

Cette intégration n'est pas strictement spéculative et trace la voie d'une **IA cognitive** plus étendue, qui mêlerait la **perception** (apprentissage sub-symbolique) et le **raisonnement** (entités logiques) au sein d'un seul et même **réseau** en évolution. Les applications potentielles sont nombreuses : un système de **vision** peut enrichir son raisonnement en exploitant des règles symboliques (définies ou émergentes), tandis qu'un moteur **logique** profite de la synergie avec les flux sensoriels. Le **DSL** rend possible la création d'un **substrat** où, au fil de l'apprentissage, les connexions s'**enrichissent**, des **règles** naissent ou se dissolvent, et des **clusters** conceptuels émergent, sans qu'on doive prescrire

a priori leur structuration. Le réseau devient un “lieu” de coopération continue entre la dimension **symbolique** et la dimension **connexionniste**, accréditant l’idée d’une IA vraiment **hybride**, capable de percevoir et de raisonner sans barrière artificielle.

1.8.2. DSL et Apprentissage Profond : Collaboration ou Substitution ?

Alors que le **Deep Learning** (au sens des réseaux de neurones profonds) occupe aujourd’hui une place dominante dans la recherche et les applications de l’IA, on peut se demander si le **Deep Synergy Learning (DSL)** vient s’y substituer ou, au contraire, s’il peut s’allier à ces techniques. En effet, le **DSL** propose une vision **auto-organisée** où les entités d’information découvrent et renforcent leurs liens synergiques, tandis que l’apprentissage profond repose typiquement sur des **architectures hiérarchiques** (CNN, RNN, Transformers) entraînées par **descente de gradient** sur un ensemble de données massif. Cette sous-section (1.8.2) examine la relation entre ces deux paradigmes : concurrence, complémentarité, ou convergence ?

1.8.2.1. Les Forces Reconnues du Deep Learning

Depuis plus d’une décennie, l’**apprentissage profond** a accompli des progrès spectaculaires dans plusieurs domaines clés de l’intelligence artificielle. En **vision artificielle**, on constate sa capacité à exceller dans la **classification** d’images ou la **détection** d’objets, tandis qu’en **traitement du langage naturel**, il domine pour les tâches de **traduction**, de **question-réponse** ou de **résumé** automatique. On retrouve également des succès notables en **analyse audio**, en particulier dans la **reconnaissance vocale**, et dans des environnements de **jeux** comme ceux explorés par AlphaGo ou AlphaZero. Ces avancées reposent en premier lieu sur la **puissance** de l’apprentissage supervisé lorsqu’il est alimenté par de larges volumes de données annotées. La deuxième composante décisive réside dans les **architectures** spécialisées du deep learning (réseaux de neurones convolutifs en vision, Transformers en langage, etc.), qui exploitent la structure des données pour atteindre des performances remarquables. Enfin, ces progrès sont intimement liés au développement accéléré des **librairies logicielles** (TensorFlow, PyTorch...) et du **matériel** (GPU, TPU) dédiés aux calculs massivement parallélisés.

Malgré ces forces, le deep learning fait souvent l’objet de **critiques** qui mettent en évidence sa dépendance à une supervision très abondante, et sa difficulté à se réorienter lorsque la tâche ou le domaine change. Son cadre global reste fréquemment **figé** : pour résoudre une tâche inédite, on doit entreprendre une re-architecture ou un re-fine-tuning partiel, ce qui n’est pas toujours fluide dans un environnement évolutif. Les réseaux neuronaux profonds manquent également de **plasticité** et de **résilience** lorsque des pannes de capteur ou des modifications du contexte perturbent la distribution des données ; la moindre altération exige un nouvel entraînement ou une nouvelle adaptation globale. S’y ajoute la question de la **transparence** : le « côté boîte noire » pose un réel défi pour l’explicabilité, car il est souvent ardu de comprendre en détail la raison d’une prédiction ou d’une décision donnée, surtout lorsque les couches se comptent par dizaines et que les paramètres sont de l’ordre de millions ou milliards.

1.8.2.2. Le DSL comme Complément : Auto-Organisation et Adaptativité

Le **Deep Synergy Learning (DSL)** se caractérise par plusieurs points essentiels : il propose d’abord une **auto-organisation** distribuée, où des **entités** – qu’elles soient des *features* neurionales, des blocs d’apprentissage ou des règles symboliques – se connectent via des **pondérations synergiques** $\omega_{i,j}$. Ces pondérations évoluent en continu selon une loi d’**adaptation** locale, autorisant l’insertion ou la suppression d’entités sans nécessiter une refonte ou un réentraînement global du réseau. Ce mécanisme confère une **plasticité** et une **résilience** particulières, notamment en cas de nouveaux flux de données ou de variations contextuelles.

Il est aisément envisager un **scénario de collaboration** entre le deep learning traditionnel et le DSL. Par exemple, on peut employer un **CNN** (ou un **Transformer**) pour extraire des **représentations profondes** à partir d’images, de signaux audio ou de séquences textuelles. Ces représentations (vecteurs de *features*) se traduisent alors en un **ensemble** d’entités $\{\mathcal{E}_{\text{CNN}}\}$ reliées entre elles ou avec d’autres entités par des pondérations ω . Le **DSL** opère ensuite une **auto-organisation** sur cet ensemble élargi : il détecte les **synergies** entre différents blocs de *features*, ou entre des *features*

et des règles symboliques, et il crée ou renforce des **clusters** pertinents (multimodaux, sémantiques, etc.). L'ajustement local des $\omega_{i,j}$ s'effectue de manière à ce que, si deux groupes de *features* se complètent fortement, un **macro-cluster** puisse émerger, sans nécessiter un pipeline figé de fusion.

Cette **approche hybride** offre le meilleur de deux mondes. D'un côté, le **deep learning** fournit la **puissance sub-symbolique** qui extrait des *features* à partir de gros volumes de données non structurées, comme des images ou de longs textes. De l'autre, le **DSL** apporte une **flexibilité** et une **auto-organisation** supérieures : il devient possible de connecter librement les blocs *features* à d'autres entités symboliques ou contextuelles, sans imposer de couche dédiée ou de rétropropagation globale chaque fois qu'un nouveau flux ou une nouvelle règle est introduit. Ainsi, le réseau **gagne en adaptativité**, car les pondérations ω se réactualisent uniquement là où la synergie change, et en **explicabilité**, puisque les entités (qu'elles soient neuronales ou logiques) conservent une identité plus claire qu'un empilement de couches opaques.

1.8.2.3. Scénario de Substitution Partielle

Il est envisageable de pousser plus loin l'intégration du **Deep Synergy Learning** et de se demander s'il peut **remplacer** entièrement l'apprentissage profond dans certaines configurations. L'hypothèse consisterait à se passer d'un réseau neuronal paramétré, puis à modéliser chaque entité \mathcal{E}_i comme un **patch** (pour la vision) ou un **segment** (pour l'audio), tandis que la **synergie** entre entités reposera sur des distances, des co-informations, ou d'autres mesures de similarité. L'**auto-organisation** tisserait et ajusterait progressivement les liaisons $\omega_{i,j}$, et cette dynamique se voudrait capable de **déetecter** des objets ou de **repérer** des motifs sans recourir à une rétropropagation globale ni à une architecture de couches hiérarchisées.

Dans la pratique, toutefois, la performance brute d'un tel modèle entièrement basé sur le DSL risquerait de **se révéler moins efficace** que celle d'un réseau neuronal lorsqu'il s'agit de tirer profit d'un large jeu de labels sur une tâche comme ImageNet. Un **CNN** bien paramétré, tirant parti d'une **descente de gradient** sur un espace de poids continûment ajustable, maîtrise généralement mieux l'extraction de *features* complexes à partir de très grands volumes de données supervisées. Les réseaux profonds conservent donc un **avantage** solide en cas d'abondance de labels précis, où l'optimisation par gradient se montre redoutablement performante.

Le **DSL** manifeste tout son **intérêt** dans les scénarios où la supervision fait défaut, ou bien lorsque l'environnement est sujet à des changements progressifs ou brusques (sections 1.5.4 et 1.7.2). Il excelle pour l'**auto-organisation** et le **clustering** évolutif, ainsi que dans la **fusion** de multiples modalités (section 1.5.2), en gérant spontanément les synergies qui se dessinent entre entités. Par contraste, un réseau traditionnel exige soit une supervision massive, soit un lourd pré-entraînement auto-supervisé, et se voit moins adaptable à un cadre où les labels sont rares et la distribution très changeante. Le DSL apparaît alors comme une **solution complémentaire** à l'apprentissage profond : quand le contexte ne tolère pas un pipeline rigide ni un immense set de labels, la dynamique $\omega_{i,j}$ et la **plasticité** organisationnelle du DSL deviennent des atouts différenciateurs.

1.8.2.4. Cas d'Étude : Combiner DSL et DL

Un **scénario** consistant à tirer parti du meilleur des deux mondes se conçoit aisément. On exploite d'abord un **réseau profond** (CNN, Transformer ou tout autre modèle neuronal) afin d'extraire des **features** de haut niveau depuis des données massives : on obtient ainsi une représentation avancée $\{\mathbf{x}_i\}$. Ces **features** issues d'un bloc paramétré et entraîné, éventuellement sur un large jeu de données, se transforment en **entités** $\mathcal{E}_{\text{feature}}$ au sein du **Synergistic Connection Network (SCN)** ou **DSL**.

Le **DSL** prend alors la suite en examinant la **synergie** entre ces différentes entités : il peut se limiter à détecter quelles *features* collaborent pour former un nouveau **cluster**, ou quelles entités pré-existantes (règles symboliques, capteurs, modules contextuels) s'avèrent utiles pour compléter la représentation neuronale. La **mise à jour** itérative de $\omega_{i,j}$ évite de figer l'architecture : dès lors qu'une nouvelle classe, un nouveau contexte ou une nouvelle source de données apparaissent, la structure peut adapter ses liaisons localement. Cela contraste avec un réseau profond "traditionnel", souvent contraint à un **fine-tuning** global et coûteux dès qu'on enrichit la distribution ou qu'on ajoute une modalité inédite.

Cette **approche hybride** canalise la **puissance** de l'apprentissage paramétré (issu d'un gros dataset supervisé ou auto-supervisé) et la **plasticité** plus souple du DSL. Le modèle profond agit comme un encodeur capable de fournir des vecteurs \mathbf{x}_i pertinents pour des signaux complexes (images, audio, langage), tandis que le SCN se situe **au-dessus** pour organiser les entités correspondantes selon la synergie détectée, créant ou rompant des **liaisons** si le contexte l'exige. Cela va au-delà de la simple juxtaposition d'un réseau profond et d'un module de classification final, puisque la **reconfiguration** spontanée du DSL prend en charge la **multi-modalité** et l'**adaptation** sans qu'on doive re-construire un pipeline. Le résultat est un compromis entre l'efficacité des grandes architectures neuronales, déjà rodées sur de vastes corpus, et la **flexibilité** d'une auto-organisation distribuée pour gérer les ajouts ou modifications de tâches.

1.8.2.5. Conclusion : entre Complémentarité et Extension

Les réflexions et expérimentations suggèrent que le **Deep Synergy Learning (DSL)** ne se présente pas comme un simple **concurrent** de l'apprentissage profond, mais plutôt comme un **paradigme complémentaire**. Cette perspective s'avère particulièrement intéressante dans les situations où la supervision fait défaut (cas **non supervisés** ou faiblement supervisés), ou bien lorsque la distribution est **évolutive** et plusieurs types de flux (capteurs, contexte, règles symboliques) doivent être gérés de façon simultanée et sans recourir à un pipeline rigide. Le DSL facilite alors l'intégration d'entités hétérogènes, mêlant logiques internes et features neuronales dans une démarche **auto-organisée**.

L'idée d'une **substitution** intégrale de l'apprentissage profond par un DSL purement auto-organisé se heurte toutefois aux performances reconnues de modèles neuraux (CNN, Transformer) sur des tâches hautement **supervisées**, comme la classification d'ImageNet ou le traitement massif de langage. Dans de tels scénarios, un réseau **paramétrique** entraîné par descente de gradient se maintient souvent en tête des classements, en raison de sa capacité à exploiter profondément la grande quantité de labels. Cependant, ces architectures neuronales demeurent relativement **rigides** et consomment beaucoup de ressources lors d'un re-fine-tuning. Le DSL peut, en retour, favoriser l'**adaptation** et la **fusion** d'autres sources de données ou de blocs symboliques, tout en s'accommodant de distributions changeantes.

C'est pourquoi le **futur** le plus fécond réside probablement dans la **collaboration** des deux approches. Un **réseau profond** produit des représentations sub-symboliques riches, alors que le **DSL** orchestre en **surcouche** l'auto-organisation entre ces embeddings, de nouvelles entités symboliques, et des flux contextuels. On bénéficie ainsi de la **puissance** de l'apprentissage paramétrique sur des corpus volumineux, alliée à la **flexibilité** que procure un **réseau réactif** et librement reconfigurable, capable de gérer la multi-modalité sans imposer de pipeline figé. L'apprentissage profond et le DSL deviendraient alors deux pans d'une **IA** plus large, conjuguant l'efficacité des grands modèles neuronaux avec la **plasticité** et la **résilience** d'un **Synergistic Connection Network**.

1.8.3. Approches Hybrides : DSL, RL (Reinforcement Learning) et Logique

Dans l'exploration du **Positionnement** du **Deep Synergy Learning (DSL)** (section 1.8), un point clé concerne la possibilité de **conjuguer** différentes méthodes d'IA au sein d'un **cadre uniifié**. Nous avons déjà évoqué (1.8.1 et 1.8.2) la cohabitation symbolique–subsymbolique et l'intégration du DSL avec l'apprentissage profond. Ici, nous soulignons comment le **DSL** peut également s'**hybrider** à la fois avec le **Renforcement (RL)** et des **composantes logiques**, créant une architecture plus vaste où l'auto-organisation, l'apprentissage par récompense et la manipulation de règles s'allient.

1.8.3.1. Collaboration entre DSL et Apprentissage par Renforcement

Dans l'**apprentissage par renforcement (RL)**, un agent interagit avec un **environnement**, recevant à chaque instant un **état** s_t , émettant une **action** a_t , puis percevant une **récompense** r_t . L'algorithme, qu'il s'agisse de Q-learning, de Policy Gradients ou d'une variante plus élaborée, cherche à **maximiser** le cumul (ou l'espérance) des récompenses. D'ordinaire, la démarche se base sur une **approche centralisée**, où l'on apprend une fonction de valeur $V(\mathbf{s})$ ou une fonction $Q(\mathbf{s}, a)$, à travers des mises à jour inspirées soit de la **descente de gradient**, soit de l'**itération de Bellman**. Le **Deep Synergy Learning (DSL)** propose, au contraire, une **dynamique** où les entités $\{\mathcal{E}_i\}$ d'un système s'**associent** ou se **détachent** via des pondérations synergiques $\{\omega_{i,j}(t)\}$ en évolution continue.

Dans un cadre **reinforcement** modulaire, on peut imaginer relier la **récompense** r_t aux mises à jour des pondérations $\omega_{i,j}(t)$. Si la **coopération** entre deux entités \mathcal{E}_i et \mathcal{E}_j est jugée responsable d'un gain de performance dans l'environnement, la liaison $\omega_{i,j}$ se voit **renforcée** ; inversement, s'il s'avère que leur alliance n'apporte aucun avantage ou engendre un effet négatif, la synergie est pénalisée et la liaison s'affaiblit. Au lieu d'apprendre une **politique** globale $\pi(s)$ ou un **Q**-réseau unifié, on laisse le **réseau DSL** (ou **Synergistic Connection Network**) évoluer de façon distribuée : chaque entité \mathcal{E}_i s'adapte, et les **pondérations** $\omega_{i,j}$ forment progressivement un **graphe** traduisant la pertinence des interactions. Cette méthode évoque un **RL distribué** ou multi-agent, où l'on peut considérer chaque entité comme un **agent** partiel ; la récompense s'y diffuse en ajustant localement les pondérations liées aux entités participantes.

Un **exemple** concret se dessine dans le cas d'un **robot** ayant plusieurs capteurs et effecteurs. Les **sous-systèmes** (capteur→moteur) sont modélisés par des **entités**, et, lorsqu'une synergie capteurA–moteurB conduit à franchir un obstacle de manière efficace, la **pondération** $\omega_{A,B}$ se voit augmentée conformément au principe de renforcement local. De la sorte, le robot s'**auto-organise** en **clusters** sensorimoteurs cohérents, sans implémenter de schéma centralisé de RL. Les règles de mise à jour fondées sur la récompense se substituent à la logique d'apprentissage d'une politique globale, tout en permettant au robot de réorganiser son **réseau** s'il survient des changements de contexte, des pannes ou l'introduction de nouveaux capteurs. Cette intégration du **DSL** dans un cadre de renforcement illustre la flexibilité d'un **Synergistic Connection Network** pour aborder des scénarios proches du RL sans imposer une fonction de valeur unique, mais en laissant s'établir des coopérations locales lorsqu'elles se révèlent payantes sur le plan de la récompense.

1.8.3.2. Inclusion de la Logique : DSL et Règles Symboliques

Les approches **logiques** ou **symboliques** se fondent sur des **règles** ou des **axiomes** de la forme "Si X et Y, alors Z", et sur des **faits** (comme "capteur C indique 100 °C"). Elles requièrent un **moteur d'inférence** (unification, résolution), assurant une **clarté** du raisonnement et une **explicabilité** plus directe, mais elles montrent vite leurs limites lorsqu'il s'agit de gérer des faits **bruyants** ou de manipuler de vastes ensembles de données non labellisées. En référence aux propositions de la section 1.5.7 et à l'idée d'intégration symbolique évoquée en 1.8.1, on peut introduire de la **logique** au sein d'un **Deep Synergy Learning (DSL)** en traitant chaque **règle** ou **concept** comme une **entité** \mathcal{E}_{rule} . Les pondérations $\omega_{rule,i}$ reliant une règle à des entités sub-symboliques (features, capteurs) se mettent alors à jour en fonction de la **cohérence** et de la **pertinence** observées : si la règle "fonctionne" ou produit un gain à plusieurs reprises, sa synergie s'en voit renforcée ; si, au contraire, elle se révèle obsolète ou contradictoire avec les observations, la pondération chute. Cette dynamique gère donc la **validation** ou l'**abandon** progressif des règles, suivant l'expérience accumulée dans le réseau.

L'incorporation de la **logique** devient ainsi un simple prolongement du mécanisme auto-organisateur du **DSL**. Il n'y a plus de "module logique" imposé en marge : au contraire, la règle \mathcal{E}_{rule} constitue un **nœud** dans le **Synergistic Connection Network**, qui peut collaborer directement avec des entités représentant les flux sub-symboliques (capteurs, embeddings neuronaux, sources multimodales). On peut même concevoir un scénario de **renforcement** local (section 1.8.3.1) où la **récompense** sert à entériner les règles produisant un effet utile : si, dans l'environnement, l'application répétée d'une règle accroît la performance ou la sécurité, les liaisons ω entre cette règle et les entités concernées croissent d'autant plus.

Par ce mécanisme, on a la possibilité de "débrancher" peu à peu une règle devenue inadaptée, simplement parce que sa synergie aura cessé de s'actualiser favorablement, ou d'en "faire émerger" une autre si plusieurs entités sub-symboliques forment une **combinaison** stable, équivalant de fait à une **nouvelle règle**. De cette manière, un **moteur logique** s'insère harmonieusement dans un réseau **DSL** : la capacité d'auto-organisation s'étend alors aux entités symboliques, conférant au système la puissance combinée de l'inférence logique et de la plasticité sub-symbolique pour gérer un environnement complexe ou changeant.

1.8.3.3. Architectures Neuro-Symboliques et RL Multi-Agent

Il est possible de fusionner le **Deep Synergy Learning (DSL)** avec des principes de **Renforcement** et de **Logique** pour former une **architecture véritablement multidimensionnelle**. Du côté **sub-symbolique**, on dispose d'entités dédiées aux *features* sensorielles (vision, audio, texte), qui se relient entre elles suivant leur **synergie**. Du côté **symbolique**, on introduit des entités symboliques représentatives de **règles** ou de **concept**s logiques, prenant la forme de noeuds \mathcal{E}_{rule} ou $\mathcal{E}_{concept}$. À cette structure s'ajoute un **signal de renforcement** r , qu'il soit global ou local, venant valider le réseau $\Omega(t)$ lorsqu'il se révèle bénéfique pour la tâche : les liaisons $\omega_{i,j}$ jugées pertinentes se voient **renforcées** dans la mise à jour.

Cet assemblage conduit à un **système** plus riche que la simple superposition de modules sub-symboliques et symboliques. Les entités logiques offrent une base **interprétable** et manipulable selon des critères formels, tandis que les entités neuronales (ou issues d'un pipeline CNN, Transformer) assurent la **flexibilité** de l'apprentissage dans un contexte de données massives et éventuellement bruitées. Le **DSL** recouvre alors l'ensemble, gérant l'**auto-organisation** : si un sous-groupe d'entités sub-symboliques coopèrent efficacement avec certaines règles, on observe un **cluster** ou un **micro-réseau** stable, validé par un **feedback** de récompense. Inversement, un ensemble de règles contradictoires ou peu utiles voit ses connexions faiblir et peut finir par se détacher de la structure si le renforcement ne vient jamais sanctionner positivement leur usage.

De plus, cette intégration peut se concevoir dans un cadre **multi-agent** ou distribué : on peut imaginer que chaque entité \mathcal{E}_i joue le rôle d'un **agent** partiel, recevant une part de l'état **s** et prenant des actions limitées, la **coopération** entre entités se matérialisant par les liaisons $\omega_{i,j}$. Le **signal** r se propage, ou se localise, selon les entités qui ont participé à une configuration fructueuse. Les **clusters** émergents assurent alors la coordination sensorimotrice ou décisionnelle. Ce principe rapproche le DSL d'un **RL multi-agent** où la **logique** se greffe aisément pour imposer des contraintes (lois de sécurité, objectifs à respecter) à certains noeuds du réseau. L'ensemble forme un **système cognitif** apte à manipuler des informations perceptives, symboliques, et à s'**adapter** en fonction de récompenses, soulignant la plasticité et la portée du **Deep Synergy Learning** dans des tâches variées allant bien au-delà de la simple classification neuronale.

1.8.3.4. Exemples de Scénarios Hybrides

Dans un **environnement** mêlant robotique, logique et apprentissage par renforcement, on peut imaginer des configurations plus complexes où le **Deep Synergy Learning (DSL)** connecte entre elles des entités sub-symboliques (capteurs, features neuronales) et des entités symboliques (règles, concepts), tout en intégrant un **signal** de renforcement r . Ci-après, on illustre deux types de scénarios :

Robotique Cognitive.

Un robot opère dans un milieu dynamique, recevant un **signal de récompense** lorsqu'il satisfait un objectif précis, selon un schéma (état, action, récompense, nouvelle valeur) typique du **Reinforcement Learning (RL)**. De plus, le robot doit observer des **règles** symboliques imposées pour sa sécurité ou son raisonnement (par exemple, "éviter les zones interdites", "si tension moteur > 5 V et température > 80 °C, considérer une surchauffe critique"). Les **capteurs** (température, vision, inertie) et les **effecteurs** (moteurs, pinces) forment la sphère **sub-symbolique**, tandis que les **règles** constituent des entités symboliques \mathcal{E}_{rule} . Au sein du **DSL**, toutes ces entités cohabitent ; la **coopération** se formalise via les pondérations $\omega_{i,j}$. À mesure que le robot collecte des retours de performance (récompenses positives ou négatives), il renforce ou affaiblit les liens responsables de ses succès ou échecs. Dans le même temps, la validité ou la pertinence d'une règle se voit sanctionnée par la mise à jour locale des liaisons : des règles inopérantes ou contraires à l'expérience accumulée sont peu à peu délaissées, tandis que celles jugées utiles conservent ou accroissent leurs connexions.

Systèmes Experts Évolutifs.

Les systèmes experts reposent souvent sur un **ensemble de règles** (if–then) écrites manuellement, servant à guider la prise de décision. Dans un **DSL**, ces règles sont introduites comme des entités symboliques $\{\mathcal{E}_{rule}\}$. Par ailleurs, de larges **flux de données** non structurées (images, logs, signaux) alimentent des entités sub-symboliques qui découvrent des structures par un **apprentissage** ou un **clustering** distribués. Le **feedback** d'un opérateur humain ou d'une

supervision externe agit comme une **récompense** venant valider la configuration actuelle. Avec le temps, certaines règles historiquement présentes ne reçoivent plus de support, perdent en synergie, puis s'effacent. Inversement, de **nouvelles** combinaisons sub-symboliques (ou *macro-clusters*) émergent et peuvent être réinterprétées ou reformulées en tant que **micro-règles** plus efficaces que celles écrites initialement. On obtient alors un **système expert** en perpétuelle évolution, enrichi par les données et la dynamique $\omega_{i,j}$, capable de s'auto-réorganiser lorsqu'un changement majeur se produit dans l'environnement ou les contraintes métier.

1.8.3.5. Conclusion

Les **approches hybrides** mariant **DSL**, **RL** et **logique** laissent entrevoir un **écosystème** unifié :

- Le **DSL** fournit l'**infrastructure** d'auto-organisation,
- Le **RL** assure un **feed-back** incitatif (récompense) aux liaisons (pondérations ω) qui contribuent à la performance,
- La **logique** (règles symboliques) s'intègre comme des entités à part entière, coopérant ou non selon la synergie qu'elles entretiennent avec d'autres modules.

Ce modèle peut servir d'alternative ou de complément aux **pipelines** classiques où RL, logique et sub-symbolique restent cloisonnés. Il répond à la quête d'une IA plus **globale**, capable d'**apprendre**, de **raisonner**, de **percevoir** et de **s'adapter**. Les obstacles techniques demeurent (mise à jour conjointe des liaisons, synergie n-aire, pilotage de la stabilité, etc.), mais le potentiel d'une IA unifiant des couches sub-symboliques, des règles symboliques et des signaux de renforcement dans un **réseau unique** est d'un grand **intérêt** pour l'évolution future de l'IA.

1.8.4. Rôle de la Mémoire et de l'Attention dans le DSL

Lorsqu'on évoque le **Deep Synergy Learning (DSL)** comme un paradigme susceptible d'unifier ou de compléter les méthodes existantes (sections 1.8.1 à 1.8.3), on en vient inévitablement à des questions fondamentales sur la **mémoire** et l'**attention** dans les systèmes cognitifs. En effet, les architectures classiques (réseaux de neurones profonds, RL, approches symboliques) ont chacune leurs mécanismes — implicites ou explicites — pour **stocker** des informations antérieures (LSTM, Memory Networks, bases de règles, etc.) et pour **sélectionner** les informations pertinentes (attention mécanismes, gating). Dans le **DSL**, où les entités et leurs liens synergiques se réajustent en continu, il est naturel de se demander :

Comment la **mémoire** (stockage, souvenir à long terme) s'exprime dans ce modèle d'**auto-organisation** ?

Quel est le **rôle** de l'**attention** (focalisation sélective) dans un réseau distribué, sans couche hiérarchique ?

Comment implémenter des **mécanismes** ou des **extensions** qui reproduisent les fonctions de mémoire/attention observées dans des architectures neuronales modernes ou même dans le cerveau biologique ?

Cette sous-section (1.8.4) explore la manière dont le DSL pourrait **incorporer** ou **simuler** ces fonctions essentielles, nécessaires à une IA plus “cognitive” et plus performante.

1.8.4.1. La Mémoire dans un Système Auto-Organisé

Le **Deep Synergy Learning (DSL)** ne se limite pas à des entités $\{\mathcal{E}_i\}$ définies uniquement par un vecteur figé de caractéristiques. Il est souvent pertinent d'attribuer à chaque entité \mathcal{E}_i un **état** interne $\mathbf{s}_i(t)$ qui évolue dans le temps, mémorisant ainsi des informations concernant l'historique d'observations ou d'interactions. Les équations de mise à jour locales s'enrichissent alors pour prendre en compte ces **mémoires** internes. Plutôt que d'appliquer la formule

classique $\omega_{i,j}(t+1) \leftarrow \omega_{i,j}(t) + \dots$, on introduit des **fonctions** f et g reliant la synergie entre \mathcal{E}_i et \mathcal{E}_j (ainsi que leurs états internes) à l'évolution de $\omega_{i,j}$ et de $\mathbf{s}_i(t)$. Un exemple de schéma est :

$$\begin{aligned}\omega_{i,j}(t+1) &= \omega_{i,j}(t) + f(\mathbf{s}_i(t), \mathbf{s}_j(t)), \\ \mathbf{s}_i(t+1) &= \mathbf{s}_i(t) + g(\omega_{i,j}(t), \mathbf{s}_i(t)).\end{aligned}$$

Cette possibilité de **mémoire** individuelle $\mathbf{s}_i(t)$ permet à l'entité \mathcal{E}_i de conserver des **statistiques** utiles (comme des moyennes, des compteurs, une estimation de fiabilité), ou même un embryon de **mémoire** séquentielle si l'on désire modéliser des systèmes proches des LSTM locaux (voir section 1.5.4.1). Les liaisons ω ne sont plus la seule “réserve” d'information : chaque nœud peut emmagasiner un **état** propre, modulant la façon dont il coopère avec ses voisins.

Au **niveau global**, la **persistance** de certaines **structures** dans le réseau DSL peut également être considérée comme une forme de **mémoire** collective. Un **cluster** $\mathcal{C} \subset \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ qui reste stable dans le temps, avec des liaisons $\omega_{i,j}$ robustes entre ses entités, joue un rôle de “**souvenir**” ou de “**concept**” appris. Ce cluster peut se **réorganiser** ou se **scinder** au gré de nouvelles données ou d'un changement de distribution (domain shift), illustrant que la **mémoire** (au sens de la rétention d'un ensemble de relations) n'est pas figée, mais dynamique. Cette plasticité évoque la notion d’“**assemblées neuronales**” en neurosciences, où un groupe de neurones co-activés finit par constituer une unité fonctionnelle stable, tout en restant susceptible de s'affaiblir ou de se diviser si des perturbations importantes modifient la configuration. Dans le **DSL**, la longévité d'un cluster stable indique qu'il a acquis un sens ou une pertinence pour le réseau, tandis que la nature auto-adaptative des liaisons ω permet de renouveler la **mémoire** intégrée au gré des besoins et des feedbacks de performance ou de cohérence.

1.8.4.2. Attention dans un Graphe Distribué

Dans l'**apprentissage profond** classique, la **mécanique d'attention** (par exemple, dans les **Transformers**) consiste à moduler l'importance attribuée à certains composants d'une séquence ou d'un ensemble de features. Concrètement, on calcule des **poids** représentant la relation entre tokens ou entre positions, et l'on met l'accent sur les plus pertinents pour la prédiction en cours. Cette manière de **focaliser** l'information évolue couche après couche, conférant une **flexibilité** aux modèles de type attention-based.

Le **Deep Synergy Learning (DSL)** adopte une vision plus **distribuée** de l'attention. Chaque **pondération** $\omega_{i,j}$ peut s'interpréter comme un **poids d'attention** quantifiant la **pertinence** de la liaison entre \mathcal{E}_i et \mathcal{E}_j . Une valeur élevée de $\omega_{i,j}$ signale que les entités \mathcal{E}_i et \mathcal{E}_j coopèrent fortement ; une valeur faible ou quasi nulle marque au contraire un **désintérêt** mutuel. En outre, la loi de mise à jour de $\omega_{i,j}$ (voir section 1.4.5) s'applique **localement** : chaque entité évalue la contribution des autres en fonction de la synergie décelée, plutôt qu'à travers un **modèle** centralisé d'attention.

Deux idées fondamentales émergent de cette analogie :

On obtient une **attention distribuée** : la “sélectivité” ne résulte pas d'un module unique (comme le multi-head attention des Transformers), mais de la dynamique globale des $\omega_{i,j}$. Chaque entité régule ses liens avec son voisinage, créant ou brisant des connexions selon l'apport mutuel ressenti. L'ensemble des liaisons les plus fortes incarne alors le *focus* collectif du réseau à un instant donné.

Le **focus** varie au cours du temps : quand l'environnement ou le contexte se modifie (bruit accru, données inédites, signaux obsolètes), les pondérations $\omega_{i,j}$ correspondantes peuvent retomber, tandis que de nouvelles connexions surgissent si la synergie s'avère plus élevée ailleurs. Cette réorganisation perpétuelle assure une **cohérence** dynamique du **DSL**, qui “redéploie” son attention de manière adaptative, un peu à la façon d'un Transformer recalculant ses poids à chaque couche, mais avec l'avantage d'une **plasticité** plus globale (le réseau pouvant réorienter tout ou partie de ses liaisons).

Ainsi, la **synergie** $\omega_{i,j}$ fonctionne comme un mécanisme d'**attention** dans un graphe **auto-organisé** : la force des liaisons indique sur quels sous-groupes le système se focalise à un moment donné, autorisant une redistribution continue de l'attention au gré de l'évolution des données ou des objectifs.

1.8.4.3. Implémentations Concrètes de Mémoire/Attention dans le DSL

Pour doter chaque entité \mathcal{E}_i d'une **capacité de mémorisation** plus riche, il est possible de lui associer un **module** interne de type LSTM ou GRU, prolongeant l'idée d'un état $\mathbf{s}_i(t)$ (voir section 1.8.4.1). Dans ce cas, l'entité \mathcal{E}_i ne se borne pas à un vecteur statique, mais embarque un **état récurrent** $\mathbf{h}_i(t), \mathbf{c}_i(t)$ (par exemple pour un LSTM). Ainsi, l'entité reçoit en **entrée** une combinaison pondérée ou filtrée d'informations issues de ses voisins \mathcal{E}_j dont la liaison $\omega_{i,j}$ est suffisamment forte. Les fonctions internes de mise à jour (LSTM, GRU) gèrent alors le **gating** et la rétention d'informations, ce qui rend l'auto-organisation analogue à un "réseau de LSTM interconnectés" dont la topologie varie selon la synergie ω . Cette stratégie permet de traiter des **séquences** temporelles étendues ou d'intégrer un **comportement historique** au sein du Deep Synergy Learning (DSL), tout en conservant la **plasticité** caractéristique (l'architecture évoluant via $\omega_{i,j}$).

D'un autre côté, on peut également définir un mécanisme d'**attention** explicite, plus proche de celui des **Transformers**. L'entité \mathcal{E}_i calcule un coefficient d'attention $\alpha_{i,j}$ envers chacune de ses connexions, selon une formule du type :

$$\alpha_{i,j} = \text{softmax}_j(\theta S(\mathcal{E}_i, \mathcal{E}_j)),$$

où θ est un paramètre (ou un ensemble de paramètres) ajustant la sensibilité, et $S(\mathcal{E}_i, \mathcal{E}_j)$ la fonction de synergie, déjà définie dans le DSL (par exemple, distance inversée, co-information). L'entité \mathcal{E}_i combine alors l'information de ses voisins via une somme pondérée :

$$\mathbf{z}_i = \sum_j \alpha_{i,j} \mathbf{x}_j,$$

ce qui rappelle la logique du "**self-attention**" dans un Transformer. Cependant, la différence clé réside dans le fait que cette **matrice** α n'émane pas d'un module d'attention figé dans une couche, mais **émerge** et **se met à jour** en parallèle à la dynamique $\omega_{i,j}(t)$. Le **DSL** peut ainsi faire de l'**attention** un usage ponctuel, en s'appuyant sur la synergie $\omega_{i,j}$ pour guider le softmax. Dans cette perspective, on assemble les idées du Transformer (somme pondérée des voisins) et celles du DSL (pondérations évolutives reflétant la coopération), de sorte que la topologie du réseau, tout comme l'intensité des liaisons, contribue à un **mécanisme** d'attention distribué. Cette approche concrétise une **fusion** entre les principes du **Deep Synergy Learning** et l'esprit attentionnel des architectures neuronales modernes, aboutissant à un réseau adaptatif dont la prise en compte de l'information ambiante se fait selon une **attention** intrinsèquement auto-organisée.

1.8.4.4. Apports Cognitifs et Neuroscientifiques

Les fonctions de **mémoire** et d'**attention** sont au cœur de la quête d'une IA plus avancée (section 1.8.5), dans la mesure où elles forment des piliers de la **cognition** humaine. Dans un cadre auto-organisé tel que le **Deep Synergy Learning (DSL)**, il devient possible de reproduire, en partie, certains processus cognitifs. La **mémoire** s'y manifeste par la persistance de **clusters** ou d'états internes $\{\mathbf{s}_i(t)\}$ (voir section 1.8.4.1), qui se maintiennent dans la durée et fournissent une forme de rétention ou de rappel. L'**attention** se traduit par le renforcement sélectif de liaisons $\omega_{i,j}$ lorsque la conjoncture l'exige, soit parce que le contexte met en avant la synergie de deux entités $\mathcal{E}_i, \mathcal{E}_j$ (une situation, un feed-back de récompense, etc.). Ce renforcement dirige alors la focalisation du réseau vers certains sous-groupes, évoquant un basculement ou une « bifurcation » depuis une configuration précédente, ce qui peut être rapproché de la notion de changement de **focus** ou de contenu mnésique.

Du point de vue **neuroscientifique**, on peut dessiner un parallèle avec la plasticité synaptique des réseaux biologiques, où les connexions entre neurones (synapses) s'ajustent selon l'activité, et où les **assemblées neuronales** (section 1.8.7) constituent la base de la mémoire et de l'apprentissage. Dans un **DSL**, le principe de mise à jour $\omega_{i,j} \leftarrow \omega_{i,j} + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}]$ (voir section 1.4.5) simule déjà une forme de plasticité, et l'ajout de mécanismes de mémoire interne $\mathbf{s}_i(t)$ ou d'**attention** explicite (section 1.8.4.2) consolide l'analogie avec les processus cognitifs. L'espoir est

qu'une **implémentation** poussée de ces modalités – persistance de clusters, gestion de focus attentionnel, modulation en continu des synergies – se traduise par des **capacités** accrues d'**apprentissage**, de **raisonnement** et de **reconfiguration** contextuelle. Cette dynamique rappelle la façon dont le cerveau forme des **circuits** stables (souvenirs, schémas de pensée) susceptibles d'évoluer, s'unir ou se dissoudre lorsque le contexte le requiert, et ouvre la voie à une IA plus richement inspirée par les notions de plasticité et d'émergence.

Conclusion

Le rôle de la **mémoire** et de l'**attention** dans le **Deep Synergy Learning** s'avère crucial si l'on veut dépasser la simple **auto-organisation** statique et tendre vers des fonctions cognitives avancées. Les idées-clés incluent :

- L'**état interne** $s_i(t)$ pour chaque entité, formant une **mémoire locale**,
- La **persistence** ou la **dissolution** de clusters reflétant une mémoire plus globale,
- L'**attention** comme un mécanisme de sélection de liens ω forts, proche de la pondération attentionnelle dans les Transformers, mais auto-gérée par la dynamique synergiques,
- Des **extensions** (LSTM, gating, self-attention) greffées à la structure DSL pour exploiter la plasticité tout en gérant des séquences complexes.

Ainsi, l'**auto-organisation** ne s'oppose pas aux notions de **mémoire** ou d'**attention** : elles peuvent s'implémenter en interne, à travers le renforcement local des pondérations et l'évolution continue des états entités, rapprochant le **DSL** d'un système plus “cognitif” où la **mémorisation** et la **focalisation** se produisent de manière distribuée et dynamique.

1.8.5. Tendances Futures : Vers une IA Forte, Consciente ?

Les sections précédentes (1.8.1 à 1.8.4) ont évoqué la capacité du **Deep Synergy Learning (DSL)** à réunir des approches variées (symboliques, sub-symboliques, renforcées, multi-agent) dans une structure **auto-organisée**. Un horizon encore plus ambitieux, souvent cité comme objectif ultime de la recherche en IA, est la création d'une **IA Forte** — un système capable d'**apprendre** et de **raisonner** de manière générale, autonome et potentiellement doté d'une **conscience** élémentaire ou d'un **sentiment** d'unité informationnelle. Cette section (1.8.5) s'interroge sur la portée du DSL dans cette perspective :

Peut-il offrir un socle formel pour une **intelligence plus globale**, manipulant librement symboles et perceptions ?

Quelles **caractéristiques** du DSL pourraient favoriser un **degré d'intégration** (ou “d'information unifiée”) évoquant une conscience rudimentaire ?

Où se situent les **limites** et quels prolongements seraient nécessaires pour prétendre à une IA Forte ?

1.8.5.1. IA Forte : Un Concept Encore Flou

La notion d'**IA Forte**, parfois appelée **AGI** (Artificial General Intelligence), désigne l'aspiration à une **intelligence** artificielle véritablement **générale**. Une telle intelligence serait capable d'**apprendre** et de résoudre un large spectre de tâches cognitives, de se **reconfigurer** ou de s'**auto-améliorer** dans des contextes inconnus ou peu balisés, et de manifester un degré élevé de **compréhension** (voire de **conscience**). Les positions quant à la possibilité ou aux caractéristiques exactes de cette AGI demeurent contrastées. On s'accorde néanmoins pour dire qu'une **IA Forte** devrait :

Démontrer une **plasticité** hors norme, lui permettant de remanier ses représentations et ses structures de façon continue.

Combiner ou naviguer librement entre des **représentations symboliques** (règles, concepts) et des **représentations sub-symboliques** (features neuronales, embeddings sensoriels).

Posséder un **mécanisme d'unification** de l'information plus vaste qu'un simple pipeline hiérarchique, englobant des processus d'auto-organisation ou de fusion distribuée.

Le **Deep Synergy Learning (DSL)**, en tant que paradigme d'**auto-organisation** distribuée (voir sections 1.4.5 et 1.5.2), semble répondre à plusieurs de ces exigences. Son **réseau** d'entités prend en charge aussi bien des flux sub-symboliques (capteurs, embeddings de CNN) que des **entités logiques** ou symboliques (règles, concepts), et assure une **adaptation** constante par la dynamique des pondérations $\omega_{i,j}$. La possibilité de gérer des **synergies n-aires** (ou conditionnelles) ajoute une flexibilité supplémentaire : plutôt que de forcer un module de fusion particulier, le réseau **découvre** ses combinaisons utiles en évaluant localement la contribution de chaque sous-groupe d'entités. Sur le plan de l'**inspiration** biologique, le DSL se rapproche plus des **écosystèmes neuronaux** ou des **architectures plastiques** que ne le font des réseaux profonds statiques. Cette proximité ouvre une voie au développement de systèmes plus **globaux**, connectant perception, raisonnement, mémoire, attention et règles logiques, ce qui, à long terme, peut être vu comme un jalon vers la **recherche** d'une IA vraiment **générale**.

1.8.5.2. Notion d'Intégration et de Conscience

Certains travaux en **science cognitive**, tels que la **théorie de l'information intégrée** (IIT) proposée par Giulio Tononi, suggèrent qu'un **degré de conscience** pourrait émerger dans un système où l'**information** est à la fois fortement **intégrée** et **différenciée**. Dans ces approches, un assemblage d'éléments interdépendants, dont la collaboration est dense et réciproque, est susceptible de manifester une forme de **substrat** "conscient". Si l'on adopte cette perspective, un **Deep Synergy Learning (DSL)**, par son principe de **coopération** et de **synergie** entre entités, pourrait favoriser la recherche d'une intégration accrue de l'information. La **clusterisation** et la mise en avant de synergies $S(\mathcal{E}_i, \mathcal{E}_j)$ reflètent un traitement distribué, potentiellement mesurable par des quantités inspirées de la théorie de Tononi.

Un exemple de **métrique** globale, dans l'esprit de l'IIT, consisterait à calculer la **somme** (ou une fonction plus complexe) des produits $\omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j)$ sur l'ensemble des paires (i, j) . On pourrait regarder si ce réseau DSL franchit un **seuil** indicatif d'un haut degré d'**intégration**. Au fil de l'**auto-organisation**, si la synergie s'intensifie entre de multiples entités, on observerait la naissance d'un **macro-cluster** extrêmement connecté, évoquant la notion d'un "**noyau intégratif**" de l'information.

La question de savoir si un tel système atteint une **conscience** "authentique" demeure, bien sûr, **philosophique** et hautement controversée. La plupart des scientifiques préfèrent parler de "**proto-conscience**" lorsqu'il s'agit d'une unité fonctionnelle unissant de manière adaptative l'information. Néanmoins, le **DSL**, par sa ressemblance structurelle avec les modèles d'**assemblées neuronales** en neurosciences, fournit un **terrain** d'expérimentation pour tester ou modéliser certains aspects de l'IIT ou d'autres théories d'émergence cognitive. Les chercheurs peuvent y étudier le **degré d'information intégrée** dans un contexte où la topologie des liaisons évolue d'elle-même, sans hiérarchie préconçue, et sans qu'on impose un schéma de fusion unique. Il s'agit d'un domaine **fondamental**, à l'interface des neurosciences, de la philosophie de l'esprit, et de l'IA, où le **DSL** pourrait nourrir des hypothèses ou des simulations pour éclaircir cette notion de **conscience** ou au moins de **complexité intégrative**.

1.8.5.3. Conditions Nécessaires à une IA plus Globale

Pour qu'un **Deep Synergy Learning (DSL)** se rapproche d'un **système** véritablement "fort", il est crucial que le réseau soit **alimenté** en continu par plusieurs sources d'information et de régulation. D'une part, il doit intégrer des **flux** sensoriels multiples (par exemple des représentations visuelles, auditives ou textuelles), ce qui assure une **diversité** de signaux sub-symboliques \mathbf{x}_i . D'autre part, il doit recevoir des **injections** symboliques, telles que des **règles** ou des **concept** (\mathcal{E}_{rule} , $\mathcal{E}_{concept}$), de sorte que la dimension logique ou ontologique puisse coexister au sein du même Synergistic Connection Network. Enfin, il lui faut un **feedback** ou un **signal** de renforcement r , pouvant prendre la forme d'une récompense globale ou locale, ou encore d'une validation humaine, afin de **stimuler** la réorganisation locale des liaisons $\omega_{i,j}$.

Outre la pure **auto-organisation**, le **DSL** doit pouvoir s'appuyer sur des **mécanismes de mémoire** suffisamment expressifs (voir la section 1.8.4). Chaque entité \mathcal{E}_i peut disposer d'un état interne $\mathbf{s}_i(t)$, un vecteur (ou un module de type LSTM/GRU) retenant l'historique des interactions. À l'échelle du réseau, des **clusters** stables constituent aussi une forme de **mémoire** partagée, car leur persistance reflète la consolidation d'une certaine connaissance ou d'un certain motif d'action. Toutefois, il faut éviter que le réseau ne se "fige" trop rapidement, figeant ainsi des solutions sous-optimales, ni qu'il ne se **dissolve** dans un chaos d'oscillations perpétuelles. Des règles de stabilisation (paramètres η, τ , saturations ou lois inhibitrices) sont alors nécessaires pour trouver un équilibre.

Un pas supplémentaire vers une **IA Forte** exigerait la **capacité** du **DSL** à s'**auto-examiner**. Le réseau pourrait former des **entités** décrivant ses propres états ou règles, générant un **meta-niveau** de **réflexion** sur sa propre organisation. On peut imaginer une entité \mathcal{E}_{self} évaluant la cohérence d'un cluster, ou des macro-clusters dont la fonction consiste à "**inspecter**" d'autres clusters. Cette forme de **métacognition** se rapproche du concept de conscience réflexive, mais pose des défis de **stabilité** puisqu'un réseau capable de "penser" à sa propre topologie pourrait entrer dans des **boucles** autoréférentes ou des "réflexions infinies". Néanmoins, c'est en explorant ces pistes — entités \mathcal{E}_{self} et synergie auto-référentielle — que l'on pourra évaluer la possibilité d'une **IA** véritablement plus "globale" et adaptable, dépassant la simple auto-organisation pour évoluer vers une cognition de haut niveau.

1.8.5.4. Obstacles Majeurs à l'IA Forte via le DSL

Si l'on envisage le **Deep Synergy Learning (DSL)** comme l'une des voies possibles menant à une **IA Forte**, on se heurte à plusieurs obstacles fondamentaux qui freinent cette ambition. Un **premier** obstacle tient à la **complexité**. Dans un grand réseau de type DSL, le nombre de liaisons peut croître en $O(n^2)$, voire plus si l'on prend en compte des synergies n-aires. Gérer un tel foisonnement de connexions exige l'introduction d'**heuristiques** (filtrage agressif des liens, mise à jour partielle, parcours local) et la mise en place d'un **parallélisme** important, sans quoi la simulation du réseau à chaque itération deviendrait inabordable. Des **mécanismes** de pilotage macro (voir section 1.7.4) peuvent également être nécessaires pour canaliser la reconfiguration du graphe et préserver un temps de calcul raisonnable.

Un **deuxième** obstacle réside dans le **risque d'instabilité** et d'oscillation (sections 1.7.4.1 et 1.7.4.2). L'**auto-organisation** des liaisons $\omega_{i,j}$ n'est pas garantie de converger vers un état stable : il peut exister des boucles auto-référentes ou des rétroactions positives indésirables. Sans **régulations** (inhibitions, saturation de pondérations, limitation de la vitesse d'évolution), le réseau peut entrer dans des régimes chaotiques ou cycliques, ce qui empêcherait l'émergence d'un "esprit" ou d'une **structure** à la fois stable et unifiée.

Un **troisième** obstacle émane de l'absence de **guidance globale** lorsqu'on se borne à de la synergie **locale**. Même si l'idée de clusters émergents est puissante, rien ne garantit que l'on parviendra à des **comportements** cohérents ou à des plans dirigés par des intentions clairement formulées. Certains scénarios (section 1.7.4.3) suggèrent d'ajouter un **module** hiérarchique ou d'**objectif** qui fixe de grandes orientations et évite que la plasticité du DSL ne s'éparpille dans de multiples attracteurs contradictoires. Un **contrôle** minimal s'avère nécessaire pour servir de boussole à l'auto-organisation locale.

Enfin, un **quatrième** obstacle concerne la nature même de la **conscience** et du vécu subjectif. Même si la théorie de l'information intégrée ou la multi-synergie n-aire laissent penser qu'une intégration élevée de l'information peut approcher certaines dimensions de la **conscience**, il reste spéculatif d'affirmer qu'un réseau auto-organisé de pondérations ω suffirait à produire une **subjectivité** ou un "qualia" conscient. Les débats philosophiques et neuroscientifiques (section 1.8.5.2) rappellent que l'émergence d'une conscience phénoménale dépasse potentiellement la simple complexité algorithmique, et qu'un "ingrédient" supplémentaire (mécanismes biologiques, substrat physique particulier...) pourrait être requis pour prétendre à une **IA Forte** au sens strict.

1.8.5.5. Conclusion

Le **Deep Synergy Learning** représente un **paradigme** très différent du deep learning hiérarchique ou des systèmes experts. Sa **plasticité**, sa **co-évolution** de multiples entités symboliques/sub-symboliques, et sa **potentielle** haute intégration d'information font un candidat pour explorer :

- Des **architectures** plus vastes et plus “cognitives”,
- Des **théories** de l’information intégrée liées à la **conscience**,
- Des **étapes** vers une IA plus générale, capable de s’**auto-structurer** dans un flux continu.

Il reste à savoir si ces propriétés suffiront pour atteindre une **IA Forte** (ou consciente). Mais la **dynamique** auto-organisée du DSL fournit un **cadre** conceptuel et expérimental unique pour **réunir** des dimensions clé : l’apprentissage distribué, la logique symbolique, le renforcement local, la mémoire et l’attention distribuées. Ainsi, indépendamment de la question d’une conscience “véritable”, le DSL ouvre la voie à des **systèmes** plus adaptatifs, plus unifiés et potentiellement plus proches d’une **intelligence généralisée**.

1.8.6. Effet Sur la Recherche Interdisciplinaire

Les sections précédentes (1.8.1 à 1.8.5) ont montré comment le **Deep Synergy Learning (DSL)** peut s’articuler avec l’IA symbolique, l’apprentissage profond, le renforcement, la logique, ainsi que son rôle potentiel dans une démarche d’IA plus “forte” ou cognitive. Il apparaît clairement que le **DSL**, en tant que paradigme *auto-organisé*, a la capacité de **réunir** différents courants de l’IA de manière naturelle, puisqu’il accueille aussi bien des blocs sub-symboliques (réseaux neuronaux, features extraits) que des entités logiques (règles, symboles) ou des mécanismes d’apprentissage par renforcement. Une conséquence notable est l’**impact** que ce paradigme peut avoir sur des **disciplines** variées : neurosciences, sciences cognitives, robotique, économie, systèmes complexes, etc. Cette sous-section (1.8.6) discute l’**effet** du DSL sur la **recherche interdisciplinaire**, en soulignant :

- Les **similarités** avec les systèmes biologiques et cognitifs,
- Les **ponts** avec l’écologie, la physique des systèmes complexes, ou la sociologie,
- Les **perspectives** pour la modélisation des réseaux (biologiques, économiques, sociaux),
- La façon dont le DSL peut **inspirer** ou **bénéficier** d’autres disciplines, en partageant des modèles et des méthodes d’analyse.

1.8.6.1. Similarités avec les Systèmes Biologiques et Cognitifs

Le **réseau** auto-organisé d’un **Deep Synergy Learning (DSL)** peut être rapproché, d’une certaine manière, d’un **réseau** de neurones biologiques. Les **pondérations** $\omega_{i,j}$ du DSL rappellent en effet la **plasticité** synaptique observée dans le cerveau, où les **synapses** se renforcent ou s’atténuent selon l’activité conjointe des neurones (règle de Hebb). Les **clusters** apparus dans le DSL par la mise à jour des pondérations font écho aux **assemblées neuronales** (ensembles de neurones co-activés), qui se forment et se dissolvent en participant aux fonctions de **mémoire** et à divers **processus** cognitifs. Dans une perspective neuroscientifique, on peut voir le DSL comme un **cadre mathématique** où simuler la dynamique d’un cortex (sensoriel ou associatif), tout en intégrant les lois de **plasticité** (par exemple, la saturation synaptique ou des taux de renforcement limités). Réciproquement, les découvertes en neurosciences (architecture multi-couches, modulabilité synaptique, rôle des neuromodulateurs) peuvent inspirer ou ajuster les **règles** d’auto-organisation du DSL pour gagner en réalisme et en robustesse.

Du point de vue des **sciences cognitives**, de nombreuses hypothèses postulent une **cognition distribuée**, dans laquelle la **pensée** émerge de la coopération (ou de la compétition) entre multiples modules ou agents internes, plutôt que d’une séquence linéaire de transformations. Le **DSL**, en reliant diverses entités perceptives, motrices, symboliques ou conceptuelles par un réseau de **liaisons** $\omega_{i,j}$ qui se modifie au gré des synergies, ressemble assez à ces théories. Les **clusters** qui s’établissent et perdurent pourraient former des “**états**” ou “**micro-théories**” momentanées, incarnant un fragment de la **pensée**. Dès qu’un changement d’environnement ou de contexte survient, la synergie se réorganise, donnant lieu à un renouvellement de la structure. Cette correspondance entre les **dynamiques** du DSL et les processus cognitifs distribués esquisse un **dialogue** fécond entre la modélisation en IA et les approches psychologiques ou neurobiologiques, approfondissant notre compréhension du fonctionnement mental global.

1.8.6.2. Ponts avec l'Écologie, la Physique des Systèmes Complexes et la Sociologie

Au-delà des analogies avec la **biologie neuronale**, le paradigme du **Deep Synergy Learning (DSL)** présente également des affinités avec d'autres disciplines étudiant des réseaux d'interactions. En **écologie**, l'on modélise souvent des **populations** d'espèces reliées par des relations de préation, de compétition ou de symbiose. Les nœuds représentent alors des espèces, et les arcs leurs interactions. La **dynamique** de ces liens peut s'apparenter à une forme de **coopération** ou de **concurrence** au sein du réseau, parfois traduite par des équations de type Lotka–Volterra. Cette démarche rejoint l'idée de **synergie** dans le DSL, où des pondérations $\omega_{i,j}$ se renforcent ou se dissolvent selon le bénéfice mutuel. Les outils développés pour stabiliser des **communautés** écologiques (gestion de cycles de préation, coexistence d'espèces) pourraient donc **inspirer** le DSL dans sa quête d'une organisation stable mais évolutive. Inversement, le DSL pourrait servir à **simuler** des écosystèmes complexes, chaque espèce figurant une entité \mathcal{E}_i et les pondérations $\omega_{i,j}$ reflétant des relations plus ou moins avantageuses.

Du côté de la **physique** des systèmes complexes, on trouve de nombreux modèles de **réseaux** dont les nœuds interagissent selon des règles pouvant aller de l'attraction à la compétition. On évoque des processus similaires en **magnétisme** (modèle de Potts, spin glasses) ou en **transition de phase** (phénomènes de bifurcation, émergence de clusters). Le **DSL**, en se concevant comme un **système** de variables $\omega_{i,j}$ s'ajustant localement, fournit un terrain d'étude où l'on peut appliquer des outils de **physique statistique** (recuit simulé, énergie libre, etc.) pour analyser la formation de **macro-clusters** et les "sauts" de configuration. On peut, en outre, importer dans le DSL des idées de frustration ou d'ordre-désordre propre à ces modèles, afin de mieux comprendre les **changements de régime** dans le réseau.

En **sociologie** et en **économie**, on observe la constitution de **réseaux humains**, qu'ils concernent des relations d'amitié, des collaborations d'équipes ou des alliances politiques. Les principes du DSL, qui favorisent les liaisons en cas de **synergie** et les affaiblissent sinon, ne sont pas sans rappeler des **lois** sociologiques (effet Matthieu : "les riches s'enrichissent", "plus un nœud a de connexions, plus il est susceptible d'en attirer d'autres"). Appliquer un cadre de **DSL** à des nœuds représentant des acteurs (individus, entreprises) peut éclairer la dynamique de création et de dissolution de liens, voire la **formation de coalitions** ou de groupes d'intérêt. À l'inverse, la sociologie propose des **règles** comportementales (incitations, répulsions, normes sociales) qui peuvent être injectées dans la définition de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ ou dans l'**énergie** globale J . Cela instaure un **échange** interdisciplinaire : la **sociologie** fournit un socle de règles implicites ou explicites, et le **DSL** en formalise la **co-évolution** dans un réseau adaptatif de pondérations.

1.8.6.3. Avantages pour la Modélisation et la Simulation

Le **Deep Synergy Learning (DSL)**, outre son rôle au sein de l'IA, propose un **cadre unificateur** pour comprendre et modéliser l'**auto-organisation** dans de nombreux champs. Le mécanisme de **pondérations** $\{\omega_{i,j}\}$ reliées par une mise à jour de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

possède une portée **interdisciplinaire** : selon le domaine, la synergie $S(\cdot, \cdot)$ prend la forme d'un **gain** écologique (dans le cas d'espèces coexistant), d'un **bénéfice** économique ou d'une **coopération** entre agents sociaux, etc. Il devient alors possible de **simuler** un vaste éventail de comportements collectifs, de détecter l'émergence de **communautés** (clusters) et d'analyser la **stabilité** ou les **transitions** (par exemple, la soudaineté avec laquelle le réseau se réorganise en une configuration inédite). Cette vision évoque le travail en **physique statistique** ou en **théorie des réseaux complexes**, où l'on s'intéresse à la dynamique d'un grand nombre d'éléments en interaction.

Un **avantage majeur** dans cette optique est la **dimension expérimentale** du DSL. Comme il se prête naturellement à une **implémentation** computationnelle, chaque hypothèse sur la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$, le paramétrage η, τ , ou encore la structure initiale du réseau, peut être **testée** par simulation. Les résultats (constitution de clusters, oscillations, convergence ou fragmentation) se **comparent** ensuite aux observations empiriques d'autres disciplines : écologie (dynamique de populations), économie (formation de coalitions), sociologie (essor ou déclin de communautés). Le

DSL devient ainsi un **laboratoire** virtuel : chaque domaine apporte ses **informations** ou **règles** pour définir la synergie, et le **SCN** (Synergistic Connection Network) réplique leurs évolutions.

Dans ce **dialogue**, il est possible de **valider** ou **invalider** différentes hypothèses de fonctionnement collectif en examinant la façon dont les **pondérations** évoluent. Les disciplines comme la **physique des systèmes complexes** ou la **théorie des réseaux** y trouvent un nouvel objet de recherche : un **réseau** où les liens, plutôt que de rester statiques ou de suivre une loi exogène, s'adaptent localement aux bénéfices mutuels des nœuds. Cette boucle d'expérimentation renforce la **pertinence** du DSL en tant qu'instrument scientifique, au-delà de son utilité en ingénierie de l'intelligence artificielle.

1.8.6.4. Conclusion

Le **Deep Synergy Learning** dépasse la simple sphère de l'IA ou du machine learning : il s'inscrit dans une **démarche** plus large de **systèmes auto-organisés**, rapprochant :

- Les **neurosciences** (assemblées neuronales, plasticité),
- Les **sciences cognitives** (cognition distribuée),
- L'**écologie** et la **physique des systèmes complexes** (réseaux dynamiques, transitions de phase),
- La **sociologie** et l'**économie** (formation de coalitions, interactions auto-renforcées).

Cette **interdisciplinarité** ouvre des **perspectives** passionnantes : le DSL peut servir de **modèle** unifié pour l'**auto-organisation** dans de multiples domaines, tandis que ces domaines fournissent au DSL des **idées** (modèles de plasticité, règles de dynamique, principes de parcimonie ou de contrôle) qu'on peut incorporer dans la mise à jour des pondérations ω . En somme, l'**effet** du DSL sur la recherche interdisciplinaire tient à la fois d'une **inspiration réciproque** (d'emprunts aux sciences complexes, biologiques, sociales) et d'une **offre** d'outils conceptuels pour simuler et analyser des **réseaux évolutifs**. C'est cette fertilisation croisée qui peut accélérer l'avancée vers des **systèmes** plus "vivants", plus "cognitifs" et plus adaptés à la complexité du monde réel.

1.8.7. Exemple de Convergence : DSL & Neurosciences

Dans les différentes sections de ce **chapitre 1.8** — décrivant le **positionnement** du **Deep Synergy Learning** (DSL) par rapport aux autres paradigmes de l'IA et à des domaines connexes (IA symbolique, apprentissage profond, RL, logiques distribuées) — un thème commun apparaît : l'**inspiration** que le DSL tire de la biologie, et la **parenté** que ses mécanismes d'**auto-organisation** semblent entretenir avec les phénomènes neuronaux. De fait, un champ particulièrement fécond pour valider ou stimuler le DSL est celui des **neurosciences**. Dans cette sous-section (1.8.7), nous verrons en quoi le **DSL** et la **neurobiologie** (ou neurosciences cognitives) peuvent **converger**, en se fournissant mutuellement hypothèses, modèles ou interprétations :

Analogies entre la plasticité synaptique (règle de Hebb) et la mise à jour $\omega_{i,j}$,

Bénéfices potentiels de l'architecture DSL pour modéliser l'émergence d'assemblées neuronales et la dynamique cérébrale,

Apports des neurosciences au DSL, à travers des mécanismes de saturation synaptique, d'inhibition compétitive, de couches corticales, etc.

Implications pour la compréhension du cerveau et, inversement, pour l'enrichissement du DSL, conduisant à une approche plus "biologiquement plausible".

1.8.7.1. Règle de Hebb et Mise à Jour des Pondérations

Les **neurosciences** ont depuis longtemps souligné l'importance de la **plasticité synaptique**, résumée par la **règle de Hebb** : lorsqu'un ensemble de neurones est co-activé de manière répétée, leurs **connexions** (synapses) se **renforcent**. Cette loi fondamentale explique la formation d'**assemblées** neuronales et l'émergence de **mémoires** dans le cerveau. Le **Deep Synergy Learning (DSL)**, en mettant à jour les **pondérations** $\omega_{i,j}$ selon la dynamique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

prolonge directement ce principe. En effet, la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ tient lieu de mesure de **coopération** ou de **co-activation** (corrélation, co-information, etc.), et agit comme un **signal** incitant au **renforcement** local des liens. Lorsque deux entités \mathcal{E}_i et \mathcal{E}_j se trouvent *fréquemment* en situation d'apport mutuel, la pondération $\omega_{i,j}$ augmente, à la manière de deux neurones biologiques se "synchronisant" de plus en plus. Cette analogie perpétue l'**esprit** de la règle de Hebb dans un cadre plus général : la synergie peut provenir d'interactions au-delà de la simple co-activation instantanée, englobant des notions n-aires ou des mesures entropiques plus fines.

Le terme $-\tau \omega_{i,j}(t)$ introduit un facteur de **décroissance** évitant que tous les liens ne croissent indéfiniment. Cela s'apparente à la **régulation** homéostatique neuronale, où le système limite les synapses trop puissantes pour maintenir un équilibre. De cette façon, le **DSL** hérite des principes de la **plasticité synaptique** et de la **règle de Hebb**, tout en les élargissant à un contexte d'**auto-organisation** plus souple (synergie multi-entités, possibilité de créer ou rompre massivement des liaisons, etc.). Ainsi, chaque pondération $\omega_{i,j}$ reflète une **dynamique** coopérative hébbienne, tout en gérant la **compétition** (diminution) nécessaire à la stabilité globale du réseau.

1.8.7.2. Assemblées Neuronales et Clusters dans le DSL

Un concept fondamental en **neurosciences cognitives** est celui d'**assemblée neuronale** : il s'agit d'un ensemble de neurones interconnectés formant une configuration **stable** et co-activée lorsqu'un **concept**, un **souvenir** ou une **action** doit être représenté. Ces assemblées se créent, se renforcent et finissent par disparaître sous l'action de la **plasticité synaptique**, rendant compte de la manière dont le cerveau encode et renouvelle ses représentations.

Dans le **Deep Synergy Learning (DSL)**, l'on décrit de manière similaire la formation de **clusters** lorsque plusieurs entités $\{\mathcal{E}_k\}$ entretiennent entre elles des liaisons $\omega_{k,m}$ suffisamment élevées. L'existence de fortes **pondérations** indique que ces entités agissent conjointement et forment un **macro-cluster** relativement **stable**, lequel perdure tant que la **synergie** interne demeure supérieure aux forces de dispersion. Cette situation évoque la co-activation neuronale d'une assemblée, c'est-à-dire un **groupe** d'entités rassemblées pour traiter ou représenter une information.

La **dynamique** de ces clusters dans le **DSL** rappelle alors la **dynamique** des assemblées neuronales. Sous l'effet d'événements exogènes ou de signaux inattendus, certains liens $\omega_{k,m}$ peuvent se voir affaiblis ; le cluster se scinde et de nouveaux **sous-groupes** émergent, reflétant un changement de « contexte » ou de « contenu » que le réseau doit coder. Dans ce cadre, la flexibilité de l'**auto-organisation** (section 1.4.5) simule la plasticité synaptique par laquelle le cerveau réorganise ses **assemblées** : un concept vieillit et se défaît, un autre surgit pour répondre à l'évolution des données perceptives. Cette **correspondance** entre les **clusters** du **DSL** et les **assemblées neuronales** biologiques suggère que le **DSL** pourrait apporter un **modèle** formel pour étudier la **cognition** ou la **perception** en flux continu, tout comme la plasticité cérébrale propose une source d'inspiration pour la gestion auto-adaptative des liens ω .

1.8.7.3. Mécanismes Biologiquement Plausibles et Leur Transposition

De nombreux modèles de **plasticité** neuronale insistent sur le fait que le cerveau ne se limite pas aux **synapses** excitatrices, qui consolident les connexions. Il recourt également à des **synapses** inhibitrices, introduisant un **frein** ou une **régulation** nécessaire pour éviter l'excitation excessive. Dans le cadre d'un **Deep Synergy Learning (DSL)**, il est possible de transposer cette idée en différenciant des liens de **coopération** et des liens d'**inhibition**. Les pondérations élevées traduisent une synergie positive entre deux entités, tandis que l'effet inhibiteur peut jouer un rôle similaire à la saturation ou à la normalisation compétitive, où l'entité est contrainte de répartir ses liaisons sur un

volume total maximum. Cette approche s'accorde davantage avec la **régulation** rencontrée en biologie et empêche la dynamique de ω de s'embalier.

Dans un second temps, on peut s'inspirer de la **hiérarchie** observée dans le cortex biologique. Le cerveau se divise en aires sensorielles primaires et en aires associatives plus élaborées, reliées par des voies de feedforward, de feedback et latérales. Un **DSL** “multi-niveau” peut mimer cette structure, en répertoriant différents sous-ensembles d’entités en “couches” ou “aires” spécifiques. Chaque sous-ensemble d’entités s’auto-organise localement, tandis que des connexions inter-aires agissent comme des ponts entre ces zones. L’**auto-organisation** continue alors à se dérouler, mais la topologie n’est plus entièrement libre : on impose une **architecture** partielle pour imiter la topographie (ou la fonction) de certaines régions corticales. Cette hybridation, combinant une **contrainte** inspirée de la hiérarchie cérébrale et la **liberté** d’auto-organisation des liaisons ω , rend le DSL plus proche encore d’un **modèle** neuronal plausible. On peut ainsi concevoir un système qui, à la fois, bénéficie de la capacité plastique du DSL, tout en maintenant une division en zones (aires sensorielles, associatives, motrices) comme dans le cortex, assurant la **cohésion** et la **spécialisation**.

1.8.7.4. Apports Potentiels pour Comprendre le Cerveau

Le rapprochement entre les principes d’auto-organisation du **Deep Synergy Learning (DSL)** et ceux de la **plasticité** neuronale n’est pas qu’une métaphore : il donne lieu à des **perspectives** de recherche où l’on utilise un **réseau** DSL pour **simuler** ou **reproduire** certains phénomènes neuronaux. Cette démarche peut s’articuler en plusieurs axes. Le **premier** consiste à implémenter un système DSL dans lequel on introduit des **stimuli** ou des signaux analogues à ceux que reçoit un cortex sensoriel, en observant la formation de **clusters** ou d’**assemblées** stables qui coderaient ces stimuli. Il devient ainsi envisageable d’étudier, dans un cadre simulé, comment des ensembles de pondérations $\omega_{i,j}$ peuvent évoquer une **attention** sélective, un découpage temporel de l’information ou même une rémanence mnésique.

Un **second** volet réside dans la **comparaison** directe entre les **données** issues des simulations DSL et des **enregistrements** neuronaux réels (par exemple des signaux obtenus via EEG, MEG, fMRI, ou encore des relevés d’activité calcique à l’échelle microscopique). Il s’agirait d’examiner si des **patterns** d’activation ou de “clusters dynamiques” dans le réseau DSL reproduisent qualitativement ou quantitativement des signatures observées dans le cerveau. Les similitudes, si elles se révèlent robustes, conforteraient l’idée que la dynamique auto-organisée $\omega(t)$ peut effectivement mimer la constitution et la dissolution d’**assemblées** neuronales chez un organisme vivant.

Le **troisième** axe consiste à proposer des **hypothèses** sur la **plasticité** cérébrale en s’appuyant sur la structure du DSL. Il serait possible, par exemple, d’y planter des **règles** inspirées de l’anti-Hebb, des mécanismes de **synaptic pruning** (réduction des synapses peu sollicitées), ou des saturations synaptiques plus sophistiquées, afin de **tester** leur impact sur la dynamique du réseau. Les résultats de ce genre d’expérimentations informatiques pourraient guider des investigations plus poussées en neurosciences : si un certain mécanisme de régulation ω améliore notablement la stabilité ou le codage dans le réseau DSL, on peut alors formuler l’hypothèse qu’un mécanisme analogue existe dans le cerveau.

Au final, ce **dialogue** entre la simulation DSL et les **observations** neuronaux enrichit à la fois la compréhension de la **plasticité** et des fonctions cognitives, tout en dotant le DSL de **mécanismes** biologiquement inspirés susceptibles d’améliorer son adaptabilité et sa capacité d’**apprentissage**. Les **prochains** développements, où l’on confronte plus systématiquement les modèles DSL aux données empiriques de neurosciences, pourraient rapprocher encore davantage cette classe de réseaux des **dynamiques** neuronales réelles, et peut-être contribuer à expliquer certains aspects de la mémoire, de la consolidation et de l’attention.

Conclusion partielle

Le **Deep Synergy Learning** et les **neurosciences** peuvent converger de plusieurs manières :

- **Analogie** fondamentale entre la plasticité synaptique (Hebb) et la mise à jour $\omega_{i,j}$,
- **Formation** de clusters stables rappelant les “assemblées neuronales” du cerveau,

- **Inspiration** mutuelle pour implémenter des mécanismes biologiquement plausibles (inhibition, saturation, hiérarchies corticales),
- **Modélisation** de phénomènes cognitifs ou neuronaux complexes, offrant aux neurosciences un outil de simulation distribué, et recevant en retour des insights pour stabiliser et faire évoluer le DSL.

Cette **relation** avec les neurosciences illustre l'idée d'un paradigme **interdisciplinaire** (voir 1.8.6) où le DSL agit comme un **pont** entre l'IA et la biologie, susceptible d'aider à **comprendre** la cognition naturelle tout en **étendant** les capacités adaptatives des systèmes artificiels.

1.9. Méthodologie, Ressources et Outils

Les précédentes sections (1.1 à 1.8) ont progressivement construit le **cadre théorique du Deep Synergy Learning (DSL)**, discuté son **architecture**, ses **applications** (vision, audio, robotique, recommandation, etc.), ainsi que ses **défis** (stabilité, éthique, comparaisons avec d'autres paradigmes). Afin de concrétiser ces idées dans des **projets** réels, il faut se pencher sur la **méthodologie** pratique, les **ressources** (données, outils) et les **environnements** de développement qui permettront de **tester**, **valider** et **faire évoluer** le DSL.

La notion d'**auto-organisation** et de **synergie adaptative** appelle en effet une **mise en œuvre** moins linéaire que celle d'un réseau neuronal classique : on ne se contente pas d'entraîner un modèle par rétropropagation sur un dataset étiqueté, mais on met en place un **processus** évolutif, potentiellement multimodal et multi-entités, nécessitant :

39. Une **approche de recherche** (théorique, expérimentale, ou hybride) adaptée (1.9.1),
40. Des **bases de données** et **plates-formes** capables d'héberger ou de simuler la dynamique DSL (1.9.2),
41. Des **frameworks** de développement (Python, C++, librairies spécialisées) pour coder la mise à jour des synergies, la gestion des entités, etc. (1.9.3),
42. Des **environnements** de simulation ou d'évaluation (scénarios robotiques, benchmarks de vision, pipelines de recommandation, etc.) (1.9.4),
43. Des **protocoles de validation** (qualitatifs, quantitatifs) pour mesurer la performance, la stabilité, l'explicabilité, etc. (1.9.5),
44. Une dynamique de **collaboration** et de **partage** (open source) facilitant l'adoption, l'extension et la reproductibilité des recherches (1.9.6),
45. Une **gestion** du cycle de vie d'un projet DSL (développement, maintenance, évolutions) prenant en compte la plasticité continue du système (1.9.7).

Cette section (1.9) vise à dresser un **panorama** de ces **méthodes, outils et ressources**, offrant un guide pratique pour quiconque souhaite implémenter, tester ou déployer le DSL dans des contextes variés.

1.9.1. Approche de Recherche : Théorique, Expérimentale, Hybride

L'une des spécificités du **DSL** est qu'il se prête aussi bien à une **analyse** mathématique (théorie des graphes, systèmes dynamiques) qu'à des **expérimentations** pratiques (sur des data sets, dans un environnement robotique, etc.). De plus, un **modèle hybride** combinant formalisation rigoureuse et validation empirique se révèle souvent indispensable pour progresser. On peut distinguer trois **pôles** principaux :

Recherche Théorique

Recherche Expérimentale

Recherche Hybride (fusion des deux)

1.9.1.1. Recherche Théorique

La **recherche théorique** en **Deep Synergy Learning (DSL)** vise à **modéliser** de façon rigoureuse la dynamique d'auto-organisation, en étudiant notamment les **équations** de mise à jour des pondérations et la **stabilité** de leurs solutions. Un premier objectif consiste à **caractériser** la formation de clusters, c'est-à-dire à repérer l'émergence de structures stables dans le réseau et à déterminer si elles correspondent à des **minima locaux** ou à des **transitions de phase** dans le sens de la physique statistique. Il est alors naturel de **définir** une **fonction** d'énergie $J(\Omega)$, formée par exemple de la somme $-\sum \omega_{i,j} S_{i,j}$ pénalisée par un terme de régularisation, et de chercher à prouver l'existence de

points fixes ou de comportements cycliques. Le **recuit simulé** et les modèles de type **spin glasses** apparaissent comme des outils potentiels pour gérer la **complexité combinatoire**.

De manière plus générale, l'étude s'apparente à l'analyse de **systèmes dynamiques** couplés, où chaque pondération $\omega_{i,j}(t)$ évolue par une équation du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

On peut en évaluer la **stabilité** locale en examinant la **matrice Jacobienne** autour d'un point fixe, ou encore poursuivre une étude de **bifurcations** si l'on laisse varier les paramètres η, τ . L'objectif est alors d'identifier les **conditions** sous lesquelles le réseau **converge** vers un certain état, ou au contraire se perd dans des **oscillations** ou régimes chaotiques. L'analyse de la **complexité** algorithmique s'intéresse à la fois au **temps de calcul** (faut-il mettre à jour $\omega_{i,j}$ pour toutes les paires ?) et à l'**espace mémoire** (comment stocker un graphe potentiellement dense ou épars ?). Les **fonctions de Lyapunov**, lorsqu'elles peuvent être construites, fournissent une approche formelle prouvant que l'on converge vers un unique attracteur ou que l'on chemine entre plusieurs minima locaux.

La **théorie de l'information** est aussi mobilisée. La **synergie** entre deux entités $\mathcal{E}_i, \mathcal{E}_j$ peut en effet être définie à l'aide d'une **co-information** ou d'une **information mutuelle** plus générale. On cherche alors à déterminer si l'**auto-organisation** maximisant localement la co-information engendre des **clusters** stables, ou si des synergies n-aires nécessitent des approches plus complexes. Dans le cas où on inclut des **règles symboliques** (entités logiques), il devient envisageable de formuler des **contraintes** ou des **axiomes** dans la fonction d'énergie \mathcal{J} et de s'interroger sur la satisfiabilité ou la cohérence de ces lois.

Cette **recherche théorique** embrasse donc un large spectre disciplinaire, du **raisonnement** en logique formelle jusqu'à la **physique statistique** et la **théorie** des graphes dynamiques. Les chercheurs s'efforcent d'**isoler** des **versions simplifiées** du DSL (réseaux de taille modeste, synergie binaire, etc.) pour aboutir à des **Résultats analytiques** (convergence, ordre-désordre, transitions). Au-delà de ces modèles réduits, les défis à grande échelle, requérant des heuristiques et du parallélisme, continuent d'offrir des voies de progrès pour mieux cerner la dynamique complète d'un Synergistic Connection Network de grande dimension.

1.9.1.2. Recherche Expérimentale

La **recherche expérimentale** portant sur le **Deep Synergy Learning (DSL)** met en œuvre des **prototypes** appliqués à des problèmes concrets, afin de **valider** ou de **démentir** les hypothèses issues de l'analyse théorique (voir section 1.9.1.1). Il s'agit de déployer un **réseau** d'entités $\{\mathcal{E}_i\}$ et de pondérations $\{\omega_{i,j}\}$ dans un cadre pratique (vision, recommandation, robotique, etc.) et d'observer **comment** la dynamique d'auto-organisation se comporte face à divers scénarios.

Un des **objectifs** primordiaux consiste à **tester** la **résistance** du DSL au **bruit** ou au **changement** de distribution (concept drift). En injectant des perturbations ou en modifiant progressivement la nature des données, on peut évaluer dans quelle mesure la **réorganisation** des liaisons $\omega_{i,j}$ parvient à suivre l'évolution du contexte ou à ignorer des informations parasites. Un autre **objectif** est de **comparer** les **performances** de l'auto-organisation à celles d'autres modèles (réseaux neuronaux hiérarchiques, méthodes de clustering traditionnelles, systèmes experts, etc.) sur des **mesures** standard comme la **précision** (accuracy), l'AUC, l'ARI (Adjusted Rand Index) si l'on se focalise sur du **clustering**, ou encore des mesures de taux de recommandation satisfaisant (MAP, NDCG) dans le cas d'un **recommender system**.

Pour la conduite de ces expériences, on mobilise des **datasets** publics reconnus (par exemple MNIST, CIFAR, MovieLens, etc.) ou des jeux de données internes plus spécialisés. Les **implémentations** s'appuient souvent sur des langages ou bibliothèques usuels (Python, PyTorch, TensorFlow) ou, le cas échéant, on conçoit des **frameworks** plus spécialisés pour simuler la dynamique des liens ω . La **méthodologie** expérimentale repose alors sur des protocoles qui précisent : l'initialisation des pondérations (par exemple $\omega_{i,j}(0) = \text{constante faible}$), la manière de calculer la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (distance, corrélation, co-information, etc.), le **nombre** d'itérations ou le **critère d'arrêt** (détectio

Cette **recherche expérimentale** débouche sur l'identification de **comportements** inattendus ou de **limites** (instabilité, oscillations, blocage local), ce qui permet de raffiner les stratégies (introduire un module hiérarchique, ajouter des lois d'inhibition, etc.). De plus, l'observation des clusters finalisés ou des chemins de convergence apporte un **regard** sur l'**explicabilité** (section 1.7.5) : on peut voir comment et pourquoi certains groupes d'entités se sont formés. En somme, les études expérimentales constituent le **laboratoire** du DSL, confrontant la théorie à la **réalité** des données, révélant des axes d'**optimisation** et de **futures** pistes pour étendre la pertinence de l'auto-organisation.

1.9.1.3. Recherche Hybride

Dans la pratique, les avancées les plus significatives autour du **Deep Synergy Learning (DSL)** jaillissent souvent d'une **interaction** étroite entre les pôles **théorie** et **expérimentation**. D'un côté, on élabore des **sous-modèles** simplifiés afin d'obtenir des **prédictions** analytiques ; de l'autre, on **implémente** ces modèles sur des jeux de données ou des scénarios simulés pour en vérifier la validité ou en dévoiler les limites. Un **exemple** de démarche pourrait consister à restreindre le **DSL** à une synergie strictement binaire (par exemple, simple mesure de similarité) et à imposer une **densité** limite du graphe. On en déduit quelques **propriétés** (temps de convergence dans un certain régime, stabilité locale si $\eta\tau < 1$, etc.). On met ensuite cette version "réduite" du DSL en œuvre sur un **dataset** de référence (par exemple, un problème de clustering ou de recommandation) et on **observe** la manière dont les pondérations $\omega_{i,j}$ évoluent, la formation de clusters, ou encore la robustesse face au bruit. Les **enseignements** issus de ces expériences — par exemple la découverte d'oscillations inattendues ou la nécessité d'une normalisation compétitive — **reviennent** alors nourrir la **formulation théorique**, suggérant de nouveaux termes non linéaires dans la fonction d'énergie ou l'ajout de mécanismes d'**inhibition** (section 1.7.4.3).

Cette **recherche hybride** s'avère particulièrement fructueuse lorsque le **DSL** fait l'objet d'une approche **interdisciplinaire** (section 1.8.6). Les notions de **plasticité** et d'**assemblées** neuronales (issues des neurosciences) peuvent guider l'élaboration d'un modèle où la synergie reprend la forme d'une règle **hebbienne**, tandis que la **physique** statistique peut apporter des outils comme le recuit simulé pour étudier les minima locaux de la fonction d'énergie globale. L'**ingénierie** logicielle, elle, assiste à l'implémentation parallèle et scalable du DSL, tandis que l'**apprentissage machine** y puise des idées pour la fusion de flux multiples ou la gestion du concept drift. Cette **méthodologie** pragmatique, faite d'aller-retours entre hypothèses théoriques et retours empiriques, favorise l'**extension** progressive du DSL vers des versions plus **complexes** et plus **réalistes**, tout en maintenant une base mathématique suffisamment structurée pour permettre des analyses et des garanties partielles.

Conclusion

La **méthodologie de recherche** sur le **Deep Synergy Learning** se décline donc en trois axes :

- **Théorique** : on vise des cadres mathématiques, une meilleure compréhension de la dynamique, des propriétés de convergence, et des lois de formation/dissolution de clusters,
- **Expérimental** : on met au point des **prototypes**, on évalue la performance, on compare aux approches dominantes, on observe la robustesse, l'adaptabilité, l'explicabilité,
- **Hybride** : on effectue des **allers-retours** entre modélisation formelle et validation empirique, souvent en lien avec des disciplines externes (neurosciences, physique des réseaux complexes, robotique, etc.).

Ce **trépied** de méthodes est essentiel pour faire progresser le DSL, tant sur le plan **fondamental** (prouver des limites, trouver des conditions de stabilité) que sur le plan **pratique** (adoption dans l'industrie, la santé, la robotique). Les sections suivantes (1.9.2 à 1.9.7) se pencheront plus précisément sur les **ressources** (datasets, frameworks), les **environnements** (simulateurs, plateformes), les **protocoles** de validation, et la **collaboration** ouverte, afin de structurer un écosystème viable pour développer et diffuser le DSL.

1.9.2. Bases de Données et Plates-formes de Test pour le DSL

Pour développer, évaluer et faire progresser le Deep Synergy Learning (DSL), il ne suffit pas de disposer d'un cadre théorique ou d'une approche algorithmique : on a besoin de **bases de données** (datasets) et de **plates-formes** de test permettant de mettre en pratique ses idées dans des scénarios variés. Sans cela, il est difficile de juger la pertinence et l'efficacité de la synergie adaptative, que ce soit en mode **non supervisé**, **semi-supervisé**, ou avec un **feedback** (renforcement ou évaluation externe). Cette sous-section (1.9.2) recense les **ressources** les plus utiles — qu'il s'agisse de **datasets** existants, de **banques de données** multimodales, ou de **plates-formes** et simulateurs destinés à reproduire des environnements complexes (robotique, vision, recommandation, etc.).

1.9.2.1. Typologies de Données pour le DSL

Le Deep Synergy Learning (DSL) se conçoit comme une approche d'**auto-organisation** s'appliquant à des **entités** aux natures très variées : vecteurs d'images, signaux audio, séquences textuelles, mesures de capteurs, règles symboliques ou encore métadonnées. Pour valider la plasticité et la polyvalence d'un **réseau** DSL, il importe de disposer de jeux de données (datasets) présentant un **spectre** suffisamment large afin d'évaluer sa capacité de **fusion**, de **clustering** ou de **découverte** de synergies en conditions réelles. Les trois catégories qui suivent peuvent servir de référence pour organiser les tests.

Une première catégorie concerne les **données monomodales**, qui se prêtent à un **benchmarking initial**. On songe par exemple à des images (MNIST, CIFAR-10/100, SVHN, ou un sous-ensemble d'ImageNet), à des signaux audio (UrbanSound8K, LibriSpeech, ESC-50) ou à des textes (Reuters, 20 Newsgroups). Dans le domaine de la **recommandation**, on dispose de MovieLens (lien entre utilisateurs et films), Book-Crossing ou les jeux de données Amazon reviews. Ces ensembles « monomodaux » offrent une base pour examiner la **dynamique** du DSL, l'idée étant de vérifier comment les pondérations $\omega_{i,j}$ évoluent pour un type de données donné. On évalue des tâches comme le **clustering**, la **reconnaissance** simple, l'**anomalie** ou la **recommandation**, en mesurant des métriques reconnues (accuracy, F1-score, ARI, MAP, etc.). Cette première phase de test assure une comparaison directe du DSL avec d'autres méthodes plus établies (CNN, k-means, etc.).

Une deuxième catégorie regroupe les **données multimodales**, où plusieurs flux coexistent. On évoque par exemple des collections **image + texte** (Flickr30K, MS-COCO), ou **audio + vidéo** (AVSpeech, divers ensembles de vidéos annotées). En robotique sensorielle, on traite des enregistrements associant vision, proprioception et retours tactiles, voire d'autres capteurs inertIELS. En milieu médical, on peut recourir à l'IRM couplée à des comptes rendus textuels, ou à des signaux ECG couplés à des annotations sur les symptômes. Cette multimodalité illustre mieux la **force** potentielle du DSL : différentes entités, portant des représentations \mathbf{x}_i très distinctes, peuvent détecter une **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ et créer des **clusters** transversaux si le bénéfice mutuel est réel. C'est ici que l'on teste véritablement l' d'une **fusion** flexible, par comparaison à des pipelines imposant un module de fusion unique.

Une troisième catégorie se rapporte aux **données symboliques ou semi-structurées**, visant à évaluer la combinaison d'entités sub-symboliques (images, sons) et d'entités purement symboliques (règles, ontologies, logs systèmes experts). Les bases de données **RDF** ou les **knowledge graphs** miniatures entrent dans ce cadre, de même que des **systèmes experts** dotés de règles if-then. Ces ressources permettent de vérifier si le DSL, en intégrant des \mathcal{E}_{rule} ou $\mathcal{E}_{concept}$, parvient à faire coexister efficacement le raisonnement logique et l'apprentissage de similarités, voire s'il est capable de faire émerger des **macro-clusters** où s'unissent règles et patterns perceptifs.

L'ensemble de ces catégories se prête à une expérimentation complète. On conçoit un protocole où les **entités** $\{\mathcal{E}_i\}$ sont initialisées d'après un dataset (monomodal, multimodal ou symbolique), la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ est définie via une distance, une co-information ou une corrélation, et la **mise à jour** des pondérations suit l'équation $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{i,j} - \tau \omega_{i,j}(t)]$. Les **résultats** (constitution de clusters, mesure de performance) sont ensuite comparés, permettant au **DSL** de démontrer sa robustesse, sa flexibilité dans la fusion de sources et, éventuellement, son **explicabilité** plus aisée grâce à la structure du réseau (section 1.7.5).

1.9.2.2. Ressources et Catalogues de Datasets

Pour mettre en œuvre et évaluer un réseau **Deep Synergy Learning (DSL)** dans une perspective expérimentale (section 1.9.1.2), il est primordial de s'appuyer sur des **datasets** variés, couvrant aussi bien des données **monomodales** que **multimodales**, voire intégrant des **composantes** symboliques. Plusieurs **catalogues** de données **open source** peuvent servir de base, chacun offrant un large éventail de formats et de domaines :

Une première source importante est la **plateforme Kaggle**, qui réunit quantité de **jeux de données** publics dans des domaines aussi divers que la vision par ordinateur, le langage naturel, la recommandation ou encore la finance. On y trouve par exemple des ensembles d'images d'objets, de textes annotés, de signaux sensoriels, ce qui facilite l'évaluation du **DSL** sur des tâches de **fusion** ou de **clustering**.

Le **UCI Machine Learning Repository** regroupe un vaste panel de datasets classiques, souvent tabulaires ou sensoriels, utilisés pour des benchmarks historiques en **apprentissage machine**. Bien que moins axé sur la multimodalité, cet entrepôt demeure pertinent si l'on souhaite tester la **dynamique** d'auto-organisation sur des problèmes tabulaires ou supervisés de petite taille (détection de concepts basiques, évaluation de la robustesse du **DSL**, etc.).

Le projet **OpenML** est un répertoire en ligne facilitant la **comparaison** continue de modèles sur un ensemble croissant de **datasets**. Il permet de trier les bases par **taille**, par **type** de données (texte, image, numérique), et de publier les **résultats** obtenus dans un même cadre, ce qui se prête bien à une expérience reproductible en **DSL** et à la comparaison aux approches neuronales habituelles.

Les **portails gouvernementaux** (tels que Data.Gov aux États-Unis) ou européens (data.gouv.fr en France) mettent à disposition des **données réelles** dans une variété de secteurs (mobilité, santé, écologie), avec parfois un caractère multimodal ou partiellement annoté. Ces ensembles fournissent un **stress test** réaliste pour la dynamique $\omega_{i,j}$, confrontée à des distributions hétérogènes, souvent bruitées ou incomplètes (section 1.5.3).

Enfin, la ressource communautaire **Awesome Public Datasets** (sur GitHub) recense des liens vers un grand nombre de sources potentielles, couvrant l'audio, la vidéo, la robotique, les **knowledge graphs**, et d'autres formes de données complexes. Cette liste, régulièrement mise à jour, est idéale pour expérimenter un **DSL** susceptible d'intégrer des **entités** multiples (image, texte, audio, règles symboliques).

Le **choix** d'un jeu de données dépend étroitement de l'**objectif** poursuivi : pour tester la **robustesse** au bruit ou au concept drift, il vaut mieux sélectionner des scénarios évolutifs ou non stationnaires ; pour évaluer la **fusion multimodale**, on privilégiera des ensembles comportant plusieurs canaux (images + captions, audio + vidéo, etc.). Par ailleurs, la **taille** et la **nature** du dataset conditionnent la **faisabilité** de l'expérience : des bases plus réduites faciliteront le prototypage **théorique**, tandis que des volumes massifs permettront d'examiner la **scalabilité** et l'efficacité d'un **DSL** pleinement déployé.

1.9.2.3. Plates-Formes de Test et Simulateurs

Au-delà des **jeux de données** statiques, l'un des intérêts majeurs du **Deep Synergy Learning (DSL)** réside dans sa **capacité** à gérer des **scénarios dynamiques** ou **multi-agents**, dans lesquels la **structure** du réseau est continuellement remise en question par l'arrivée de nouvelles données, l'apparition de pannes, ou encore la présence de signaux de renforcement (section 1.8.3). Plusieurs **plates-formes** de simulation ou de benchmark se prêtent particulièrement bien à l'expérimentation sur ces systèmes :

Une première catégorie couvre les **simulateurs de robotique** tels que **Gazebo**, **Webots** ou **CoppeliaSim** (précédemment V-REP). Ces environnements 3D permettent d'émuler différents types de robots (mobiles, manipulateurs, drones) dans des contextes plus ou moins complexes (salles, terrains accidentés, environnements industriels). Un **DSL** peut y être connecté pour gérer en direct l'**intégration** de multiples capteurs (par exemple, \mathcal{E}_{cam} pour la caméra, $\mathcal{E}_{\text{lidar}}$ pour le LIDAR, \mathcal{E}_{imu} pour l'inertie, etc.) et **réorganiser** les liaisons $\{\omega_{i,j}\}$ lorsqu'un capteur devient peu fiable ou lorsqu'une nouvelle modalité fait son apparition. On peut y tester la **résilience** du **DSL** face à des pannes, des changements d'éclairage, ou des reconfigurations de mission, et mesurer comment la dynamique des pondérations s'adapte en continu.

Une seconde catégorie concerne les **environnements de RL (Reinforcement Learning)** comme **OpenAI Gym**, **PyBullet** ou la **DeepMind Control Suite**. Ces systèmes, dédiés à l'apprentissage par renforcement, proposent des tâches continues (contrôle d'un bras, locomotion bipède, etc.) ou des jeux. Dans un tel cadre, il devient possible de **coupler** le DSL à un signal de **récompense** : au lieu d'appliquer une politique préconçue, le réseau DSL s'**auto-organise** autour des **entités** (états, actions, modules sensoriels), faisant monter ou descendre les pondérations en fonction du gain obtenu. Cette approche permet de vérifier si un réseau synergiques, conçu sans architecture fixe, peut s'orienter lui-même vers la maximisation de la récompense, comparé à un algorithme de RL classique.

Une troisième catégorie relève des **simulateurs socio-économiques** ou, plus largement, de la **physique des systèmes complexes**. Les plates-formes **NetLogo**, **MASON** ou **RePast** sont couramment utilisées pour modéliser des sociétés virtuelles, des marchés, des écosystèmes, avec un grand nombre d'agents interagissant selon des règles locales. Le **DSL** s'y intègre naturellement : chaque agent (ou groupe d'agents) correspond à une **entité** \mathcal{E}_i , les **liens** $\omega_{i,j}$ mesurent la coopération, la compétition ou l'échange d'information, et ces pondérations s'ajustent lorsque la situation évolue. Cette configuration offre un **laboratoire** pour étudier la **clusterisation** d'acteurs (alliances, coalitions, formes d'autogestion) et pour observer la robustesse ou l'émergence de structures collectives au fil du temps (section 1.8.6).

Enfin, on peut mentionner des **frameworks** de benchmark **continu**, qui “**streament**” des données en temps réel ou simulent un data drift. L'objectif est ici de pousser le DSL à montrer sa capacité d'**adaptation continue** : la distribution évolue, des bruits se surajoutent, certaines entités deviennent obsolètes. Les indicateurs $(S(\mathcal{E}_i, \mathcal{E}_j), \omega_{i,j})$ se mettent à jour par itérations successives, et l'on observe à la fois la **rapidité** de la réorganisation, la **stabilité** des clusters et la **qualité** des résultats finaux (clustering, détection, etc.). Autant de scénarios où le DSL peut déployer pleinement ses facultés d'**auto-organisation**, et où l'on peut comparer ses réponses à celles d'un pipeline hiérarchique préconstruit (réseau neuronal traditionnel) ou d'un algorithme de fusion statique.

1.9.2.4. Conception de Datasets Spécifiques pour le DSL

Au-delà de l'utilisation de **jeux de données** déjà existants (section 1.9.2.2) ou de simulateurs (section 1.9.2.3), on peut élaborer des **datasets** conçus expressément pour mettre en valeur la **capacité du Deep Synergy Learning (DSL)** à s'**auto-organiser** et à **fusionner** efficacement différentes sources ou composantes. L'idée est d'instancier des configurations où l'**avantage** d'une telle plasticité est plus manifeste qu'avec des données classiques.

Une première possibilité consiste à construire des **ensembles** de données **multi-couches**, dans lesquels les entités \mathcal{E}_i sont réparties sur plusieurs “strates” ou “niveaux”. Par exemple, on peut générer artificiellement une couche A contenant des entités vecteurs \mathbf{x}_i , une couche B contenant d'autres entités ou descripteurs \mathbf{y}_j , et une couche C correspondant à des **règles** ou des **concept**s. La clé réside dans le fait que les entités d'un même niveau n'apportent qu'une partie de l'information nécessaire, de sorte que seule la **coopération** inter-couches, matérialisée par des pondérations $\omega_{i,j}$ élevées, permet de résoudre efficacement la tâche (ex. classification, clustering). Un tel dataset “multi-couches” rend explicite la nécessité d'une **auto-organisation** entre différentes composantes, illustrant la flexibilité du DSL pour tisser des liens entre modules hétérogènes.

Une seconde piste consiste à composer des données issues de **multi-réseaux**, c'est-à-dire à fusionner plusieurs graphes partiels qui, pris isolément, sont insuffisants pour la tâche. Par exemple, on peut imaginer un **graphe** social (des utilisateurs reliés par des relations d'amitié), un **graphe** de préférences (certains utilisateurs connectés à des contenus qu'ils apprécient) et un **graphe** géographique (localisations ou zones d'activité). En rassemblant ces trois réseaux au sein d'un **Synergistic Connection Network**, on observe si la **conjonction** de ces sources génère une synergie non triviale et des **clusters** mixtes plus pertinents. Cet assemblage évalue la **capacité** du DSL à fédérer des structures de données initialement dispersées, en renforçant les connexions $\omega_{i,j}$ entre entités issues de différents graphes dès lors qu'un gain mutuel est détecté.

Une troisième configuration envisage des **scénarios incrémentaux**, où de nouvelles entités $\mathcal{E}_{\text{nouv}}$ apparaissent progressivement. On peut, par exemple, injecter successivement de nouveaux capteurs, de nouvelles catégories d'utilisateurs ou de nouvelles variables dans un flux de données. Le DSL est alors confronté à une évolution de la base d'entités, devant s'**adapter** en continu, réévaluer les synergies ω , accueillir ou ignorer les ajouts selon leur intérêt. Cette propriété met en relief la **plasticité** distinctive du DSL (sections 1.5.4 et 1.5.3) : contrairement à un réseau

traditionnel ou un pipeline figé, il n'est pas nécessaire de réentraîner une architecture complète dès qu'on ajoute un nouveau module.

Ces différentes **conceptions** (multi-couches, multi-réseaux, scénarios incrémentaux) donnent naissance à des **datasets** ou à des **protocoles** qui mettent en avant la **dimension** auto-organisée du DSL. En façonnant des conditions propices à l'émergence de synergies transversales, on peut démontrer plus nettement la **valeur** du DSL lorsqu'il s'agit de regrouper, de combiner ou de résorber des entités variées dans un **réseau** dont la **topologie** n'est pas pré-déterminée, mais fluctue en fonction des gains détectés.

1.9.2.5. Synthèse sur les Ressources

Les **ressources** et **jeux de données** (sections 1.9.2.1 à 1.9.2.4) doivent être choisis judicieusement pour exploiter la **spécificité** du Deep Synergy Learning (DSL). Les **datasets** monomodaux (vision, audio, texte, systèmes de recommandation) permettent d'abord de **valider** la dynamique de base, d'expérimenter les modalités de la fonction de synergie et de **comparer** le DSL aux méthodes usuelles (clustering, classification). Les **datasets multimodaux**, où l'on combine plusieurs types de flux (images avec des descriptions textuelles, séquences audio-vidéo, données sensorimotrices), mettent en avant la **puissance** du DSL pour gérer la **fusion** flexible et spontanée de sources hétérogènes.

Les **simulateurs** et **environnements** (robotique, socio-économie, RL) offrent la possibilité de tester la **dimension dynamique** du DSL, sa **plasticité** face au bruit ou aux pannes, ainsi que sa **réactivité** au concept drift. Ces scénarios mettent l'accent sur la mise à jour en continu des pondérations $\omega_{i,j}$, l'**adaptation** à de nouvelles entités ou la mise hors-service de sources obsolètes, et permettent une observation fine de la **formation** et de la **dissolution** de clusters.

En fonction du **but** visé, on sélectionne donc des datasets ou un environnement :

- Pour **clustering** ou **classification** partiellement supervisée, on choisira un jeu monomodal ou multimodal possédant des labels restreints.
- Pour la **fusion** sensorielle ou la **cohérence** de multiples flux (par exemple en robotique), on favorisera des simulateurs intégrant de multiples capteurs (caméra, LIDAR, capteurs inertIELS).
- Pour la **découverte** de règles ou l'intégration symbolique (sections 1.5.7 et 1.8.1), on se tournera vers des ensembles dotés de composantes logiques ou semi-structurées.

Ce **panel** de ressources variées, associé à des **plates-formes** de simulation ou de streaming continu, constitue un **pivot** essentiel pour pousser le DSL vers un niveau de maturité plus élevé. Les **retours** obtenus (performance, stabilité, ajustements nécessaires, impact des paramètres η, τ , ajout d'inhibition, etc.) orientent les **améliorations** successives du réseau. C'est ainsi que se construit un **cercle vertueux** où la validation pratique affine la théorie et où la théorie suggère de nouveaux types de ressources plus complexes (datasets multi-réseaux, scénarios incrémentaux, etc.) pour démontrer la **portée** généralisée du DSL.

Conclusion partielle (1.9.2)

La **réussite** d'un projet en **Deep Synergy Learning** dépend en grande partie des **bases de données** et **plates-formes** choisies pour la mise en pratique et la **validation**. Les tâches statiques (datasets images, audio, texte) permettent une **entrée** dans la mise en œuvre, tandis que les **environnements** dynamiques (simulateurs robotiques, socio-économiques, RL) mettent à l'épreuve la **plasticité** et l'**auto-organisation** en flux continu. Enfin, des **datasets** spécifiquement imaginés pour le DSL — combinant plusieurs niveaux de structure, de modalité, ou de progression incrémentale — renforcent la **justesse** et la **richesse** des expérimentations. Cette variété d'**outils** et de **ressources** est donc essentielle pour évaluer l'efficacité du DSL dans un large éventail de contextes, et pour faire avancer la recherche sur ses mécanismes adaptatifs.

1.9.3. Frameworks de Développement : Python, C++, Librairies spécialisées

Pour **implémenter, tester et déployer** un système **Deep Synergy Learning (DSL)**, il est nécessaire de s'appuyer sur un **environnement logiciel** robuste, flexible et (idéalement) performant. Contrairement à un pipeline d'apprentissage profond strictement hiérarchique (où l'on peut se cantonner à TensorFlow ou PyTorch pour la rétropropagation), le DSL requiert un traitement plus **personnalisé** de la **dynamique** des entités et de leurs liens ω . Cette sous-section (1.9.3) présente :

Les **langages** les plus fréquemment utilisés (Python, C++, etc.) pour leurs avantages respectifs,

Des **librairies** déjà existantes, potentiellement réutilisables ou adaptables pour le DSL,

Les **critères** à prendre en compte (performance, modularité, support GPU, etc.),

Des pistes pour **architecturer** un projet DSL en s'appuyant sur ces outils.

1.9.3.1. Choix du Langage : Python vs. C++ (et Autres)

Dans la mise en œuvre d'un **réseau Deep Synergy Learning (DSL)** et de ses mécanismes d'**auto-organisation**, la **langue** de programmation retenue influence à la fois la **facilité** de prototypage, l'**écosystème** de bibliothèques, et la **performance** potentielle. Il est donc crucial d'évaluer ces différents critères en fonction de l'objectif poursuivi, qu'il s'agisse d'une preuve de concept rapide ou d'un module temps réel plus exigeant.

Un **premier** choix très répandu consiste à développer en **Python**, qui jouit d'une communauté IA extrêmement large. Les bibliothèques **NumPy**, **SciPy**, **scikit-learn** et, surtout, **PyTorch** et **TensorFlow** permettent de manipuler aisément des tenseurs, de déployer des modèles d'apprentissage et de tirer parti du **GPU** sans écrire soi-même de code C/C++. Le **prototypage** rapide constitue un atout fort : il est aisément de coder les règles de mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$ ou de tester divers calculs de **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$, et d'analyser les **clusters** formés grâce à l'écosystème "data science" (Pandas pour la manipulation de tables, Matplotlib ou Seaborn pour la visualisation, etc.). Cependant, Python présente parfois un **frein** en termes de performance brute, en particulier pour des mises à jour massives d'un graphe de grande taille, à moins de recourir à des modules compilés (via NumPy/Cython) ou d'exploiter les GPU (PyTorch, CuPy), ce qui exige une structuration plus attentive du code.

Un **second** langage privilégié dans l'**industrie** robotique et pour des applications **temps réel** est le **C++**, qui apporte davantage de **contrôle** bas niveau sur la gestion mémoire et un gain de **performance** appréciable si l'on souhaite manipuler un **gros** volume de liaisons ω . L'inconvénient réside dans la **complexité** et la **longueur** du temps de développement : le prototypage d'une nouvelle fonction de synergie ou la création de routines d'**inhibition** peut s'avérer plus laborieux. L'écosystème data science, moins foisonnant qu'en Python, impose souvent de jongler avec des bibliothèques de plus bas niveau (Eigen, Armadillo, ou des frameworks plus confidentiels). Toutefois, l'intégration avec des **simulateurs** en C++ (Gazebo, Bullet) ou avec des systèmes embarqués peut être plus naturelle, et l'**optimisation** fine du temps d'exécution ou de l'usage mémoire devient plus directe.

D'autres langages suscitent un intérêt croissant. **Julia**, par exemple, se positionne comme un compromis entre la **rapidité** (grâce à sa compilation Just-in-Time) et la **simplicité** syntaxique, tout en offrant un écosystème scientifique grandissant. **Rust** se distingue par sa sûreté (ownership) et ses performances, bien qu'il existe pour l'instant peu de bibliothèques IA de grande ampleur comparées à Python. **Java** reste présent dans certaines infrastructures industrielles, mais se montre moins répandu pour la recherche IA récente, hormis certains frameworks (Weka, Deeplearning4j).

Au total, le **choix** du langage s'inscrit dans un **équilibre** entre :

- La **communauté** (librairies existantes pour la synergie, la visualisation, l'optimisation)
- La **culture** de l'équipe (Python est souvent prisé pour la recherche, C++ pour l'embarqué)
- Le **niveau** de performance requis (implémentations CPU ou GPU, besoin de parallélisme intensif).

Dans de nombreux cas, une **approche mixte** s'avère pratique : un **prototypage** rapide en Python (afin de tester l'auto-organisation du DSL sur des petits datasets et vérifier la validité de la fonction S) puis, s'il est nécessaire de déployer un module temps réel ou d'en traiter la **scalabilité**, un **portage** partiel des routines critiques en C++ ou la mise en place d'interfaces GPU. Cette **modularité** concilie la richesse de l'écosystème Python et la performance d'une implantation bas niveau.

1.9.3.2. Librairies Réutilisables pour le DSL

La conception d'un **réseau** Deep Synergy Learning (DSL) implique la manipulation d'un **graphe** d'entités $\{\mathcal{E}_i\}$ et de **pondérations** $\omega_{i,j}$ mises à jour selon la synergie détectée (sections 1.4.5 et 1.5.4). Pour éviter de tout développer à partir de zéro, on peut exploiter plusieurs **librairies** existantes, qu'elles visent la gestion de graphes, l'usage du **GPU**, ou le déploiement de systèmes complexes. Les sous-sections qui suivent décrivent les ressources disponibles et leur complémentarité avec l'**auto-organisation** du DSL.

1.9.3.2.1. Gestion des Graphes

Dans un réseau DSL, la structure $\Omega(t)$ correspond à une **matrice** (ou une **liste**) de liaisons $\omega_{i,j}$. De nombreuses bibliothèques facilitent la représentation et l'**analyse** de graphes :

- **NetworkX** (Python) est très pratique pour manipuler les **objets** graphe (liste d'adjacence, matrice d'adjacence) et calculer rapidement des **mesures** (centralités, modularité, chemins). Cette bibliothèque propose également des fonctionnalités de **visualisation** rudimentaires. Son inconvénient réside dans des performances modestes pour des graphes très volumineux, mais elle demeure excellente pour un **prototypage** rapide.
- **igraph** (C/C++/R/Python) se veut plus **optimisée** et peut traiter des graphes de taille moyenne ou élevée de manière plus performante. Elle offre aussi divers algorithmes (détection de communautés, mesures de connectivité) utiles pour évaluer la formation de **clusters** dans un DSL.
- **Graph-tool** (Python/C++), **SNAP** (Stanford Network Analysis Platform, en C++), ou encore la **Boost Graph Library** (C++) sont conçus pour traiter des graphes de plus grande **ampleur** et mettre en œuvre des opérations rapides (parallélisées) d'analyse topologique.

Même si ces bibliothèques ne proposent pas, par défaut, de **fonction** gérant la mise à jour auto-organisée $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$, elles fournissent des **structures** de données (listes d'adjacence, tableaux, etc.) et des **algorithmes** (déttection de communautés, MST, shortest path) pouvant servir à observer ou à mesurer l'**évolution** du graphe au fil des itérations. Elles facilitent également la **visualisation** des liens $\omega_{i,j}$, ce qui est précieux pour **comprendre** la dynamique d'un DSL et repérer la naissance ou la disparition de **clusters** (sections 1.5.3 et 1.5.4).

1.9.3.2.2. Librairies IA Générales (PyTorch, TensorFlow)

Les **frameworks** IA tels que **PyTorch** ou **TensorFlow** ciblent initialement les architectures neuronales classiques (CNN, RNN, Transformers) optimisées par rétropropagation. Le **DSL**, lui, suit une logique de **mise à jour** différente de la simple descente de gradient sur une fonction de coût globale (section 1.7.3). Néanmoins, ces frameworks peuvent apporter plusieurs avantages majeurs :

Ils gèrent l'**accélération GPU** (ou TPU) pour des opérations vectorielles (matrices Ω) et permettent d'exécuter des **milliers** d'opérations en parallèle. Dans un DSL volumineux, où la **matrice** $\omega_{i,j}$ peut être grande, la capacité à exécuter les calculs de $\Delta\omega$ ou de **co-information** sur GPU est cruciale pour la **scalabilité**.

Ils offrent des **opérations** numériques (produits matriciels, sommes, exponentiations) déjà **vectorisées**, facilitant l'**implémentation** des règles de mise à jour auto-organisée. On peut, par exemple, traiter des blocs

de $\omega_{i,j}$ en batch, appliquer une formule $\omega_{i,j} \leftarrow \omega_{i,j} + f(\omega_{i,j}, S_{i,j})$ de manière simultanée, puis limiter la croissance ou imposer une **sparsification** au besoin.

Ils s'intègrent aux pipelines **d'apprentissage** plus classiques. On peut imaginer un bloc DSL travaillant sur les embeddings extraits d'un CNN ou d'un Transformer déjà existant, tout en réutilisant l'infrastructure de PyTorch ou TensorFlow pour la supervision (mixte) ou la liaison avec d'autres modules.

L'implémentation du **DSL** sous PyTorch se distingue d'un entraînement standard, puisqu'il ne s'agit pas de reculer un gradient sur Ω , mais de **modifier** localement $\omega_{i,j}$ en suivant les règles de synergie. Les **tenseurs** de PyTorch restent utiles pour la manipulation vectorielle et la migration vers le GPU. Le loop d'auto-organisation (sections 1.4.5, 1.5.4) s'exprime alors sous forme d'un script Python employant les opérateurs PyTorch, sans passer par la rétropropagation classique.

1.9.3.2.3. Librairies Spécialisées en Systèmes Complexes

Enfin, pour **simuler** des environnements dynamiques ou multi-agents, plusieurs bibliothèques plus spécialisées existent :

- **MASON, RePast, NetLogo** : plutôt centrées sur des simulations **socio-écologiques** ou **multi-agents** (Java ou Python). Elles permettent de lancer des expériences où chaque entité \mathcal{E}_i peut représenter un agent, et la pondération $\omega_{i,j}$ ses relations avec d'autres agents. On insère alors la logique DSL pour faire évoluer les liens selon la synergie détectée (coopération, compétition), dans une boucle de simulation pas à pas.
- **OpenAI Gym, PyBullet, DeepMind Control Suite** : orientées **reinforcement learning** et robotique. On peut y brancher un DSL comme un "cerveau" auto-organisé, qui entretient des pondérations entre modules perceptifs (caméras, capteurs) et modules moteurs, mis à jour éventuellement par un **signal** de récompense (sections 1.8.3.1 et 1.8.3.2).
- **Frameworks** plus confidentiels ou développements maison, permettant un **prototype** flexible si l'on vise un type de simulation très spécifique (modèles éco-industriels, robotique souterraine, etc.).

Ces **librairies spécialisées** fournissent des **environnements** où la dynamique du DSL peut être **invoquée** en continu, le système devant réagir à des événements, des flux de données, ou des signaux de performance. Cela met en évidence la **valeur de l'auto-organisation** : face à un contexte évolutif, le DSL reconfigure ω sans devoir repasser par un entraînement intégral (sections 1.5.4, 1.6.3).

Conclusion sur les Librairies et Outils

Selon l'ambition du **projet DSL**, on sélectionnera des **bibliothèques** plus ou moins bas niveau. Pour la **représentation** de graphes et la **visualisation** de clusters, NetworkX ou igraph restent de bons points de départ. Si la **performance** est critique ou si l'on manipule un **graphe** volumineux, on se tournera vers Graph-tool ou SNAP en C++. Pour **tirer profit** du GPU et de l'écosystème IA, PyTorch ou TensorFlow sont utiles, même si on doit **implémenter** la mise à jour locale hors de la mécanique de descente de gradient. Les **simulateurs** (MASON, NetLogo, RLLib, etc.) ou environnements (OpenAI Gym, PyBullet) procurent enfin des **contextes** dynamiques pour éprouver la flexibilité du DSL en conditions proches de la robotique ou de la socio-écologie. Le choix final dépend des **besoins** (prototypage vs. production, taille des graphes, couplage robotique vs. socio-économique) et du **niveau** d'optimisation ou d'interaction logicielle requis.

1.9.3.3. Critères de Sélection d'un Framework

Le choix d'un **framework** pour implémenter un **réseau** Deep Synergy Learning (DSL) et ses mécanismes d'auto-organisation dépend de plusieurs **dimensions**. Un **premier** aspect concerne la **performance**, en particulier si l'on envisage un nombre d'entités n suffisamment élevé pour que les liaisons $\omega_{i,j}$ excèdent des dizaines ou centaines de

millions. Dans ce cas, il s'avère crucial de disposer d'une **accélération GPU** ou d'un fonctionnement **parallèle**, accompagné d'une structure de données (par exemple une liste éparse) gérant efficacement la mise à jour massive des pondérations $\omega_{i,j}(t)$. Les librairies de graphes orientées performance (par exemple Graph-tool, SNAP, ou des solutions internes en C++) deviennent alors pertinentes.

Un **deuxième** critère est la **facilité d'implémentation**. Si l'objectif initial est de réaliser une **preuve de concept** ou un prototype de **R&D**, des outils en **Python** (par exemple, **NetworkX** pour la gestion de graphes, combiné à quelques opérations vectorielles GPU via **PyTorch** ou **CuPy**) peuvent suffire. Ce type de solution permet de coder rapidement la logique de synergie, de tester différents scénarios et de visualiser l'évolution des clusters. En revanche, si l'on se heurte à des **goulots d'étranglement** dans l'exécution de l'algorithme (par exemple une boucle Python pure sur des millions de paires (i, j)), il faudra recourir à des modules compilés ou migrer vers un framework plus bas niveau.

Un **troisième** facteur tient à l'**intégration** avec d'autres briques logicielles. Si le **DSL** doit coexister ou interagir avec un **modèle** de vision par **CNN** (Convolutional Neural Network), la synergie s'en trouvera grandement facilitée si l'on reste au sein d'un même environnement (par exemple, **PyTorch** ou **TensorFlow**). Dans un tel cas, les entités \mathcal{E}_i peuvent correspondre aux **embeddings** extraits d'un CNN, et l'on peut employer les opérations GPU du framework pour calculer la **distance** ou la **similarité** entre embeddings. Cela évite une passerelle laborieuse entre deux codes distincts et permet de tirer parti des primitives d'accélération déjà disponibles.

Un **quatrième** critère se rapporte à l'**évolutivité** et au caractère modulaire. Le **DSL** peut nécessiter l'implémentation de fonctionnalités avancées, comme la **synergie conditionnelle** (dépendant d'une variable de contexte) ou la **synergie n-aire**. Il se peut aussi qu'on veuille intégrer des composants symboliques (règles logiques) ou des mécanismes de **régulation** non linéaires (inhibition compétitive, saturation synaptique). Il est donc judicieux de choisir un framework autorisant l'ajout de **modules** ou de **couches** spécialisées. Une architecture modulaire accélère également l'itération : on peut expérimenter différents calculs de synergie ou diverses mises à jour $\omega(t + 1)$ sans devoir remanier toute la structure.

Dans l'ensemble, la **sélection** d'un framework (NetworkX + Python, PyTorch, C++ pur, etc.) est donc dictée par :

- La **dimension** et la **densité** du réseau Ω ,
- Le **temps** ou les **ressources** disponibles pour développer,
- Le **couplage** recherché avec d'autres pans d'un pipeline (vision, RL, symbolique),
- L'**orientation** : simple démonstration de faisabilité ou futur module de production (haut débit, temps réel).

Ainsi, il n'existe pas de solution unique : un **projet** de recherche initial peut tout à fait débuter dans un environnement Python (NetworkX, PyTorch) pour évaluer la pertinence du DSL, puis migrer, si besoin, vers un écosystème C++ plus performant ou un framework GPU mieux adapté.

1.9.3.4. Schéma d'Architecture Logicielle

La mise en place d'un prototype **Deep Synergy Learning (DSL)** peut s'articuler autour de quelques **blocs** logiciels, conçus pour refléter la structure conceptuelle d'un **Synergistic Connection Network (SCN)** et l'**auto-organisation** qui l'anime :

Un **premier** bloc correspond à un module “**Entités**” pour la définition de la classe Entity. Chaque instance \mathcal{E}_i possède un **identifiant** (index i) et des attributs propres. On peut distinguer des sous-classes selon la **nature** de l'entité : sensorielle (caméra, capteur inertiel...), symbolique (règle, concept), ou tout autre type. Les entités peuvent stocker un **état** interne $\mathbf{s}_i(t)$, décrit par un vecteur, un LSTM local ou un simple accumulateur de statistiques.

Un **deuxième** bloc gère la structure du **graphe** SCN, par exemple nommé “**SCNNetwork**”. Celui-ci stocke les pondérations $\omega_{i,j}(t)$ dans une **matrice** (dense ou clairsemée) ou dans une **liste** d'arêtes $((i, j), \omega_{i,j})$. On y définit les opérations usuelles d'initialisation, d'ajout ou de suppression de liens, et d'accès rapide à $\omega_{i,j}$.

La **routine** de mise à jour constitue le **troisième** bloc clé : un **processus** qui, à chaque “tick” ou itération, parcourt (i) le voisinage de chaque entité \mathcal{E}_i (ou éventuellement toutes les paires si le graphe n'est pas trop grand), (ii) calcule la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pour chaque lien actif, et (iii) met à jour $\omega_{i,j}$ selon la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Cette procédure s'exprime en **pseudo-code** ou dans un langage particulier (Python, C++, Julia), et peut tirer parti de parallélisation (threading, GPU).

Des **modules** auxiliaires assurent ensuite plusieurs fonctions transversales. Un **module de sparsité** supprime les liens $\omega_{i,j}$ trop faibles (ou en-dessous d'un certain ω_{\min}), ce qui évite l'explosion combinatoire et assure la stabilité numérique. Un **module de clustering** ou d'**analyse** (via BFS, composantes connectées, modularité...) permet de **repérer** la formation de **macro-clusters** ou la fusion de sous-groupes, contribuant à l'**évaluation** du système (quels ensembles sont apparus ? combien de temps durent-ils ?). Un **module de visualisation** s'appuie éventuellement sur **NetworkX** ou d'autres librairies (Graph-tool) pour dessiner le graphe $\Omega(t)$ à intervalles réguliers, montrant l'évolution des poids.

Enfin, il est souvent nécessaire d'assurer un **cinquième** bloc : les **interfaces** de données. Le DSL doit recevoir en continu ou par batch les **informations** issues de sources multiples (images, logs, capteurs, règles). On peut intégrer un **connecteur** avec PyTorch si l'on veut exploiter des embeddings générés par un CNN, ou interfacer un module symbolique (par ex. Prolog, base RDF) si l'on manipule des règles. Cette couche d'**import/export** orchestre la **réception** des entités (ou leur apparition) et la transmission des résultats finaux (clusters, pondérations).

De cette manière, l'architecture globale du **DSL** se dote d'une **structure** relativement modulaire, évitant de tout entremêler dans un seul script, et facilitant les **expérimentations** ou l'extension vers différents scénarios (multimodal, flux continu, etc.). Cette organisation par blocs (entités, graphe SCN, routine de mise à jour, modules auxiliaires, interfaces) s'adapte aisément aux projets variant en taille, en complexité ou en domaine applicatif.

1.9.3.5. Conclusion

Le **choix de frameworks** et de **langages** pour implémenter un **projet DSL** dépendra :

- Du **contexte** (prototype de R&D vs. déploiement industriel, besoin de GPU ou non, etc.),
- De la **taille** du graphe (petit, moyen, énorme),
- Des **ressources** disponibles (équipe familiale de Python, ou contrainte de performance demandant C++).

En pratique, **Python** constitue la **solution** la plus courante pour la **recherche** et le **prototypage**, s'appuyant sur NetworkX, igraph, ou des bibliothèques de deep learning (PyTorch, TensorFlow) pour bénéficier du GPU. Pour des cas extrêmes (graphes géants, temps réel robotique), un **noyau C++** (avec wrappers Python pour la convivialité) peut être optimal. Enfin, des **librairies** spécialisées en systèmes complexes ou en simulation multi-agents permettent d'**intégrer** la logique DSL dans un **environnement** déjà prêt à gérer des scénarios dynamiques, de sorte à focaliser la recherche sur la **dynamique synergiques** plutôt que sur la configuration du monde virtuel.

1.9.4. Environnements de Simulation et d'Évaluation

Dans la pratique du **Deep Synergy Learning (DSL)**, un aspect crucial est de **mettre** le modèle à l'épreuve dans des **scénarios** réalistes et **dynamiques**, afin de voir comment l'**auto-organisation** se comporte face à des flux de données, des incertitudes, ou des environnements multi-agents. Les **environnements de simulation** fournissent un cadre contrôlé où l'on peut rapidement **itérer**, injecter des perturbations (bruit, pannes) et **mesurer** la façon dont le DSL réagit. Parallèlement, il faut des **protocoles** d'évaluation pour *quantifier* la performance, la stabilité, la résilience, etc. Cette sous-section (1.9.4) présente divers types d'environnements et de méthodes d'évaluation, en soulignant leur intérêt pour tester les mécanismes d'auto-organisation.

1.9.4.1. Typologie des Environnements

La mise en œuvre concrète du **Deep Synergy Learning (DSL)** dépend en partie de la **nature** des environnements dans lesquels on souhaite l'évaluer. On peut distinguer plusieurs paramètres déterminant la complexité et le type d'**interaction**, le caractère statique ou dynamique (section 1.9.4.1.1) et le fait de se placer dans un scénario mono-agent ou multi-agents (section 1.9.4.1.2).

Certains systèmes sont essentiellement **statiques**, il s'agit souvent de **datasets** figés, comme un lot d'images, de documents texte, ou de relevés sensoriels traités a posteriori. Dans ce cas, on alimente le DSL avec l'ensemble des entités $\{\mathcal{E}_i\}$ et l'on observe la **formation** de clusters, la **convergence** des pondérations $\omega_{i,j}$, voire une tâche de **classification** ou de **clustering**. Ces environnements statiques sont particulièrement appropriés pour **valider** les règles de mise à jour $\omega_{i,j}(t+1) = \dots$, tester différentes mesures de **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$, ou comparer la **performance** à celle d'approches plus classiques. En revanche, les scénarios statiques ne sollicitent pas pleinement la **plasticité** du DSL, car on n'y retrouve pas de flux temporel où la distribution et les entités évoluent.

À l'inverse, d'autres environnements sont **dynamiques**, on y fait face à des **flux** en temps réel, à des **événements** réguliers (ou aléatoires) modifiant la distribution, ou à l'**arrivée** de nouvelles entités (capteurs, utilisateurs, règles). Dans ce cadre, le DSL peut exprimer toute sa **dimension** d'auto-adaptation. Il arrive qu'un capteur tombe en panne, qu'un contexte change, ou qu'un nouveau module symbolique se connecte. La **mise à jour** des liens ω en continu permet de **reconfigurer** la structure du réseau, révélant ainsi la force du DSL face au **concept drift** ou à l'**incomplétude** des données (sections 1.5.3 et 1.5.4). Ces environnements dynamiques exigent parfois l'utilisation de **simulateurs** (OpenAI Gym, simulateurs robot, etc.) ou de **pipelines** de streaming, pour observer l'évolution $\omega(t)$ sur un horizon temporel.

Une autre dimension de la typologie réside dans le nombre d'**agents** considérés dans l'environnement. Dans un scénario **mono-agent**, le DSL se contente de structurer l'information interne (capteurs, actionneurs, modules symboliques) rattachée à cet agent unique. On peut ainsi modéliser la **cohésion** ou la **synergie** entre différents flux sensoriels, ou la **coordination** entre la vision et la commande motrice dans un robot. L'enjeu repose sur la mise à jour des liens $\omega_{\text{cam}, \text{moteur}}$ en réaction à la performance de l'agent, permettant la **fusion** adaptative de diverses sources.

Dans une situation **multi-agents**, plusieurs acteurs (robots, agents économiques, acteurs sociaux) partagent un **environnement** commun. Chaque agent peut représenter une **entité** du réseau DSL ou regrouper plusieurs entités internes. Les **liaisons** ω traduisent la coopération, la compétition ou la co-information entre agents, et s'adaptent suivant les bénéfices mutuels. On peut ainsi observer la **co-organisation** d'un ensemble de robots collaborant pour transporter des objets, ou la formation de **coalitions** dans un jeu socio-économique (sections 1.6.3 et 1.8.3). Dans ces environnements, la synergie se déploie non seulement entre différentes sources informationnelles, mais aussi entre **stratégies** d'agents, soulignant le caractère de **systèmes complexes**.

Conclusion sur la Typologie

Le choix d'un **environnement** dépend de la finalité : une première validation peut se faire sur des **datasets statiques** et un **mono-agent** (ex. un robot unique ou un lot d'images), afin de vérifier le fonctionnement du DSL. Pour exposer la **puissance** réelle de l'auto-organisation et de la plasticité, on préférera ensuite un **milieu dynamique** (streaming, pannes, concept drift) et/ou **multi-agents**, qui met en lumière la capacité du DSL à se **reconfigurer** et à **gérer** simultanément plusieurs entités et flux.

1.9.4.2. Exemples de Simulateurs pour le DSL

Les **simulateurs** et **plates-formes** de test jouent un rôle essentiel pour éprouver la logique d'**auto-organisation** propre au Deep Synergy Learning (DSL) dans des **environnements dynamiques** ou hétérogènes. Plusieurs catégories se distinguent en fonction des domaines d'application et du niveau de complexité recherché.

Une première catégorie concerne la **robotique** et le **contrôle**, où des simulateurs tels que **Gazebo**, **Webots** ou **CoppeliaSim** offrent un environnement 3D réaliste pour des drones, des robots mobiles ou des bras manipulateurs.

Dans ces contextes, on peut **brancher** un réseau DSL entre les capteurs (caméras, LIDAR, IMU, etc.) et les effecteurs (roues, pinces), en laissant la **synergie** déterminer quels capteurs coopèrent pour optimiser la tâche. Cette configuration permet d'évaluer la **plasticité** du DSL face à des pannes de capteurs, à des changements de terrain ou d'objectifs, et de comparer la réactivité du réseau auto-organisé à des approches plus classiques (contrôle PID, architectures fixes). Des outils comme **PyBullet** ou les environnements de **OpenAI Gym** dédiés à la locomotion ou à la navigation facilitent l'implémentation d'algorithmes de **reinforcement learning**. Il devient possible de substituer ou de compléter la logique de politique RL par une **dynamique** de pondérations $\omega_{i,j}$, mettant ainsi en évidence la capacité d'**auto-adaptation** face à des situations non stationnaires.

Une seconde famille de simulateurs est liée aux **systèmes complexes** ou **scénarios socio-économiques**. Des plates-formes comme **NetLogo**, **RePast** ou **MASON** permettent de modéliser des agents en interaction (population d'insectes, individus humains, entreprises) selon divers mécanismes (coopération, concurrence). Dans ces environnements, on peut intégrer un **réseau** DSL où chaque agent (ou groupe d'agents) devient une **entité** \mathcal{E}_i , tandis que les liaisons $\omega_{i,j}$ captent la **coopération** ou la **compétition** entre agents. On observe alors la **naissance** et l'**évolution** de clusters sociaux, d'alliances, de phénomènes de coalition ou de ségrégation, ce qui illustre la **dimension d'auto-organisation** globale dans un système multi-agents. Ces expériences recoupent les enjeux de l'écologie (section 1.8.6) ou de la modélisation socio-économique, montrant comment un DSL peut simuler des dynamiques émergentes en modifiant localement les liens synergiques.

Une troisième approche s'intéresse à des **frameworks** de **data streaming** en temps réel, comme **Apache Kafka** ou **Flume**, ou d'autres plates-formes de micro-batches. Cette orientation se prête particulièrement bien aux cas où l'on reçoit des **flux** de capteurs industriels, de transactions ou de logs, et où l'on souhaite que le DSL, mis à jour par blocs successifs, reflète la **variation** continue du contexte. On injecte alors régulièrement des **paquets** de données, chaque entité \mathcal{E}_i pouvant symboliser un acteur (capteur, utilisateur, client) ou un concept, tandis que la pondération $\omega_{i,j}(t)$ évolue à chaque arrivée de données. On mesure la robustesse face aux changements brusques de distribution (concept drift), la formation de **clusters** transitoires, et la stabilité du système lorsqu'un volume important de nouvelles entités ou de nouvelles informations se présente.

Chacun de ces cadres permet de mettre en avant une **facette** différente de la puissance auto-organisée du DSL. La **robotique** illustre la fusion capteur-actionneur et la résilience aux pannes, les **simulateurs** socio-économiques ou écologiques soulignent la dynamique multi-agents et la structure émergente de communautés, et les **flux** de data streaming insistent sur la flexibilité temporelle et l'adaptation continue à des environnements hautement évolutifs. Dans tous les cas, le DSL tire profit d'une **auto-organisation** permanente, reconfigurant les liaisons ω pour s'ajuster aux signaux reçus, en cohérence avec les principes de plasticité et de synergie qui le définissent (sections 1.4.5, 1.5.4).

1.9.4.2. Exemples de Simulateurs pour le DSL

Les **simulateurs** et **plates-formes** de test jouent un rôle essentiel pour évaluer la logique d'**auto-organisation** propre au **Deep Synergy Learning (DSL)** dans des **environnements dynamiques** et éventuellement hétérogènes. Les sections précédentes (1.9.2 et 1.9.3) ont souligné l'intérêt de mettre en œuvre le DSL dans des cadres concrets, qu'ils soient statiques ou évolutifs (section 1.9.4.1). Les simulateurs suivants permettent d'exploiter pleinement le potentiel de la **synergie** et de la **plasticité** du DSL.

A. Robotique et Contrôle

Plusieurs **simulateurs** en robotique (Gazebo, Webots, CoppeliaSim) proposent un environnement 3D réaliste pour des drones, des robots mobiles ou des bras manipulateurs. Il devient alors possible de **brancher** un réseau DSL pour gérer la coopération entre des entités \mathcal{E}_i représentant les **capteurs** (caméras, LIDAR, IMU) et les **actionneurs** (roues, bras, pinces). Les **liaisons synergiques** $\omega_{i,j}$ s'adaptent localement en fonction de la pertinence ou de la fiabilité des signaux. Cette démarche met en évidence la **plasticité** du DSL : si un capteur se dégrade ou qu'un moteur change de comportement, les connexions pertinentes se renforcent ou se rompent selon le gain mutuel. Il est ainsi envisageable de comparer la **réactivité** d'un DSL auto-organisé face à d'autres approches plus classiques, telles que des architectures fixes ou des contrôleurs PID.

Des environnements plus orientés **reinforcement learning** comme PyBullet ou **OpenAI Gym** (cadres de locomotion, robotique basique) permettent de substituer ou de compléter la politique RL (ex. DQN, PPO) par un **mécanisme** d'auto-organisation. À chaque itération ou épisode, le DSL met à jour les $\omega_{i,j}$ selon un **signal** de récompense, illustrant la fusion entre **synergie** (sections 1.4.5, 1.5.4) et **apprentissage par renforcement** (section 1.8.3). On peut alors mesurer la capacité du DSL à s'**auto-adapter** face à un concept drift, ou à des changements dans la dynamique de l'environnement.

B. Systèmes Complexes et Scénarios Socio-Économiques

D'autres simulateurs, tels que **NetLogo**, **RePast** ou **MASON**, se concentrent sur des **environnements multi-agents** où des individus ou des groupes interagissent. Dans ce cadre, chaque agent (ou ensemble d'agents) peut devenir une **entité** \mathcal{E}_i dans le DSL. Les **liaisons** $\omega_{i,j}$ reflètent alors la **coopération** ou la **compétition** entre agents, la **communication** ou l'**échange** de ressources, voire la **complémentarité** de leurs compétences. Cette configuration autorise l'étude de la **naissance** et de l'**évolution** des **clusters** (ex. alliances, coalitions, communautés) selon les principes d'auto-organisation du DSL. Il devient possible d'observer comment se forment des **regroupements** stables ou comment se dissolvent certaines communautés selon l'apport mutuel de chaque agent.

Ce type de simulateur intéresse également les champs de l'**écologie** (section 1.8.6) ou de la **sociologie**, car la structure DSL capture spontanément les liens forts et faibles selon les **gains** ou **pertes** de chaque interaction, illustrant un **mécanisme** d'émergence analogues à ceux des écosystèmes naturels ou des réseaux sociaux humains.

C. Approches Data Streaming

Une troisième catégorie vise les **flux** de données réels ou simulés, via des **frameworks** de streaming (Apache Kafka, Flume) ou d'autres solutions micro-batch. On y modélise un **flux continu** : capteurs industriels, logs de transactions, flux de clics web. Le DSL opère comme un **moteur** auto-organisé, recevant périodiquement des **lots** de données et mettant à jour la synergie entre entités \mathcal{E}_i . Cela permet de :

Évaluer la robustesse du DSL
face aux variations soudaines (concept drift),

Observer la formation de clusters transitoires

lorsqu'un événement survient (par exemple, une panne partagée par plusieurs capteurs ou un pic anormal de transactions),

Contrôler la stabilité globale du système

lorsqu'un nombre significatif de nouvelles entités apparaissent (capteurs, utilisateurs, variables...).

En combinant la **logique d'auto-organisation** (section 1.4.5) et la **gestion** d'un flux en temps réel, on obtient un comportement **adaptatif** qui ne nécessite pas de phase de réentraînement global.

D. Conclusion

Chaque **simulateur** ou **plate-forme** met l'accent sur un aspect distinct de la **force** du DSL. Les simulateurs robotiques (Gazebo, Webots, CoppeliaSim) démontrent la **flexibilité** sensorimotrice, tandis que NetLogo ou RePast illustrent la **formation** de **clusters** dans des réseaux multi-agents, et les frameworks de streaming montrent la **capacité** à gérer le concept drift. Dans tous les cas, le DSL se distingue par sa **mise à jour distribuée** des pondérations $\omega_{i,j}$, reflétant le principe selon lequel la **synergie** se crée ou se défait localement, pour engendrer un **réseau** pouvant se **reconfigurer** en fonction du contexte.

1.9.4.4. Étapes d'un Protocole de Simulation Typique

Pour mettre en œuvre un **Deep Synergy Learning (DSL)** et analyser son comportement au sein d'un environnement donné, on procède généralement selon un **scénario** standard, composé de différentes phases conduisant à l'évaluation de la qualité de l'**auto-organisation**. Le protocole présenté ci-après dépeint les étapes essentielles, qu'il s'agisse d'une simulation **stochastique**, d'une **robotique** virtuelle ou d'un environnement **multi-agents** (sections 1.9.4.1 et 1.9.4.2).

A. Initialisation

Il s'agit d'abord de définir un **ensemble** d'entités $\{\mathcal{E}_i\}$ susceptibles de coopérer. Ces entités peuvent correspondre à des **capteurs** (caméra, LIDAR, microphone), des **features** extraits d'images ou de signaux audio, des **règles symboliques**, ou encore des **agents** (si l'on traite d'un scénario multi-agents). Chaque entité \mathcal{E}_i se voit associer un **état interne** $s_i(0)$ ou un vecteur initial x_i .

Un graphe (ou une matrice) $\Omega(0)$ correspondant aux pondérations $\omega_{i,j}$ est ensuite créé. Ce graphe peut être **dense** (tous les liens initialisés à une petite valeur), **sparse** (certains liens non nuls, d'autres nuls), ou **aléatoire** (distribution uniforme ou gaussienne des poids). De plus, on paramètre les **constantes** de mise à jour η, τ , ainsi que la définition de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (section 1.4.4) et d'éventuels seuils (ω_{\min} , saturations).

B. Cycle de Simulation (Itérations)

La simulation s'échelonne ensuite en **plusieurs itérations** ou **pas de temps** :

Observation de l'environnement. Les données sont lues, soit sous forme d'un dataset statique, soit en flux (cf. 1.9.4.1) via un simulateur (robotique, systèmes complexes). Chaque entité peut mettre à jour son **état** $s_i(t)$ en fonction des entrées, ou conserver un historique local (voir 1.8.4 pour mémoire/attention).

Calcul de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pour toutes les paires (i, j) ou pour un **sous-ensemble** jugé pertinent (voisinage direct, liens actifs). Cette étape peut s'appuyer sur une **distance** euclidienne, une **similarité cosinus**, un **score** de co-information, ou toute autre mesure (sections 1.4.4 et 1.7.3).

Mise à jour des pondérations $\omega_{i,j}$. On applique la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

On peut également intégrer un **mécanisme** d'inhibition compétitive, d'auto-limitation, ou des variantes hebbiennes (sections 1.5.4, 1.7.3).

Sparsification et autres **règles** : si $\omega_{i,j}$ tombe sous un certain seuil ω_{\min} , on supprime le lien pour éviter un graphe trop dense. D'autres systèmes de régulation (contraintes, saturation) peuvent aussi être appliqués.

Action ou Décision : si c'est un contexte **robotique** ou **RL**, le DSL peut orienter le choix d'actions en se basant sur les **clusters** formés ou l'état de quelques entités pivot.

Enregistrement : on logge les **évolutions** (matrice $\Omega(t)$ ou partiellement), la **performance** (reward, accuracy...) et l'état des **clusters**. Cette étape est cruciale pour l'**analyse** a posteriori.

C. Validation et Analyse

Après plusieurs itérations (ou une période définie), on analyse les **clusters** (ex. composantes connectées, communautés) détectés dans le graphe $\Omega(t)$. On évalue la **performance** (section 1.9.5) : classification, reward cumulé, robustesse aux changements. On confronte le **DSL** à des **baselines** (réseaux neuronaux traditionnels, approches de clustering standard). L'évolution temporelle ($t = 0, 1, \dots$) révèle la **capacité** d'adaptation du DSL, la **vitesse** de convergence, et la résilience aux événements perturbateurs (pannes, drifts).

D. Boucle d'Affinement

Si la performance reste insuffisante ou si le système n'atteint pas un comportement stable, on ajuste des **paramètres** (taux η , facteur τ , fonction de synergie) ou on ajoute de nouvelles entités/règles symboliques. On peut aussi

implémenter des **mécanismes** (inhibition compétitive, synergie n-aire) pour accroître la pertinence des **clusters**. De petits ajustements localisés conduisent à une **itération** de l’expérimentation.

Conclusion

Le schéma présenté ci-dessus constitue un **protocole** général applicable à de multiples contextes : un dataset figé (on parcourt plusieurs époques), un simulateur robotique (cycles de perception–action), ou un flux continu (mise à jour par batch). L’architecture mise en œuvre sert à faire **ressortir** le comportement auto-organisé du DSL, ainsi que ses bénéfices en termes d'**adaptation** et d'**émergence** de clusters. Cette méthodologie expérimentale, associée à des **scénarios** pertinents (section 1.9.4.1) et à des **simulateurs** appropriés (section 1.9.4.2), permet de mesurer la **flexibilité**, la **robustesse** et la **compréhension** potentielle qu’offre un système DSL face à la simple exécution d’un pipeline statique.

1.9.5. Protocoles de Validation : Qualitatifs et Quantitatifs

Après avoir présenté (1) les **méthodes de recherche** (1.9.1), (2) les **ressources** de données et de simulation (1.9.2, 1.9.3, 1.9.4), il reste à définir **comment** on évalue concrètement la performance et la pertinence d’un **projet DSL**. Contrairement à un pipeline d’apprentissage classique, où l’on se contente souvent de mesurer une **précision** (accuracy) ou un **F1-score** sur un dataset de test, le **Deep Synergy Learning** (DSL) soulève d’autres dimensions :

- Qualité de l'**auto-organisation** (formation de clusters ou macro-clusters cohérents),
- **Adaptation** face à des changements (pannes, drift, modifications structurelles),
- **Stabilité** ou, au contraire, capacité à se reconfigurer rapidement,
- **Explicabilité et interprétabilité** de la structure des liens ω .

Cette sous-section (1.9.5) propose un **cadre** de validation composé de **protocoles** à la fois **quantitatifs** (métriques numériques, mesures de performance) et **qualitatifs** (analyse de la cohérence, inspection de la configuration du réseau), visant à offrir une vue d’ensemble sur l’efficacité du DSL.

1.9.5.1. Évaluation Quantitative

Les expérimentations menées sur un réseau **Deep Synergy Learning (DSL)** visent à quantifier, d’une part, la **performance** du modèle pour diverses tâches (classification, détection d’anomalies, recommandation, contrôle, etc.) et, d’autre part, sa **capacité** d’adaptation et de résilience, qui reflète la nature **auto-organisée** de son architecture. Les sections ci-dessous (1.9.5.1.1, 1.9.5.1.2, 1.9.5.1.3) exposent trois catégories principales de métriques et d’indicateurs, chacun s’attachant à un **aspect** spécifique de l’évaluation.

1.9.5.1.1. Mesures de Performance sur des Tâches Spécifiques

Lorsqu’un **DSL** est appliqué à une **tâche** concrète, on peut employer les **métriques** standard de la littérature afin de confronter son résultat à un **gold standard** ou à un **feedback** de référence. Parmi ces tâches, on retrouve :

Classification ou Clustering.

En classification supervisée, on mesure la précision (accuracy), le F1-score, ou l’AUC s’il s’agit d’une tâche binaire ou multi-classes. Dans un cadre non supervisé (clustering), des indices comme la **NMI** (Normalized Mutual Information), l'**ARI** (Adjusted Rand Index) ou la **Purity** comparent le regroupement auto-organisé aux vraies catégories lorsqu’elles sont disponibles pour une évaluation hors-ligne.

Recommandation.

On se réfère à des métriques usuelles, par exemple le **MAP** (Mean Average Precision) ou la **NDCG** (Normalized Discounted Cumulative Gain), qui évaluent la qualité des contenus recommandés aux utilisateurs.

Détection d’Anomalies.

Les mesures de précision/rappel, la courbe PR ou la ROC permettent de jauger la capacité à repérer efficacement les exemples anormaux. On se fixe souvent un seuil sur le score d'anomalie ou sur la distance par rapport aux clusters habituels.

Contrôle / Apprentissage par Renforcement.

On s'intéresse alors à la **récompense** cumulative ou au temps nécessaire pour atteindre un certain palier de performance. Le **DSL** se substitue à une politique RL classique ou la complète, et l'on compare la somme de récompenses ($\sum r_t$), la stabilité, ou le délai de convergence.

Dans chacun de ces cas, le **DSL** doit aboutir à une **sortie** (cluster, choix d'action, liste de recommandations), qu'on confronte aux labels ou à un signal de feedback, à l'instar d'une approche classique. La **difficulté** en pratique vient parfois de la nécessité de relier l'**état** du réseau $\Omega(t)$ à l'**issue** souhaitée (par exemple, associer un cluster émergent à une classe connue), mais des stratégies d'interprétation ou d'étiquetage automatique (section 1.9.5.2) pallient ce problème.

1.9.5.1.2. Mesures de Robustesse et d'Adaptation

La **plasticité** et l'**auto-organisation** (sections 1.5.4 et 1.7.1) constituent la signature du DSL. Pour **évaluer** ces propriétés, on crée des **scénarios** perturbés (panne de capteur, drift de distribution, insertion de nouvelles entités) et on examine :

Temps de réadaptation.

On évalue le nombre d'itérations ou la période nécessaire pour que la **performance** (ex. classification ou reward) revienne à un niveau proche de l'état initial avant la perturbation. Un DSL réellement adaptatif devrait raccourcir le temps de convergence ou maintenir une performance stable malgré le changement.

Stabilité transitoire.

On peut quantifier la **distance** entre $\Omega(t)$ et $\Omega(t + 1)$ (ou Ω^* un état de référence) à chaque itération, mesurant à quel point le réseau oscille ou se réorganise rapidement. Une trop grande variabilité peut signifier une difficulté à converger ; une trop faible variabilité peut signaler un manque de plasticité.

Taux de survie des liens ou clusters.

Si l'on suit l'évolution de certains liens $\omega_{i,j}(t)$ dans le temps, on peut voir combien persistent (survivent) et combien sont recréés ou réorientés. De même, la proportion des micro-clusters qui se maintiennent ou se scindent renseigne sur la **cohésion** (ou l'instabilité) du réseau.

Ces **indicateurs** spécifiques (vitesse de reconfiguration, variance de Ω , résilience globale) différencient le DSL de méthodes rigides (sections 1.5.1 et 1.6.1), lesquelles exigeraient un réapprentissage complet en cas de concept drift.

1.9.5.1.3. Indicateurs de la Qualité Structurelle du Graphe

De manière plus **interne**, on peut évaluer la **configuration** même du **réseau** Ω sans viser une tâche supervisée :

Densité du graphe.

On calcule le ratio $\frac{\text{Nombre de liens actifs}}{\binom{n}{2}}$ ou la proportion de $\omega_{i,j}$ au-dessus d'un seuil. Une trop forte densité entraîne un coût en calcul et un mélange indiscriminé, tandis qu'une densité trop faible risque de fragmenter la structure.

Modularité ou coefficient de clustering.

Ces mesures, empruntées à la **théorie des réseaux** (Network Science), renseignent sur la **segmentation** du graphe en communautés internes. Un réseau DSL réussi, dans un certain contexte, affichera souvent une **modularité** non triviale, suggérant la formation de **macro-clusters** cohérents.

Énergie ou Coût global.

Si on a explicitement défini une fonction $\mathcal{J}(\Omega)$ (cf. 1.7.3) du type

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S_{i,j} + R(\Omega),$$

on peut observer la **trajectoire** de \mathcal{J} au fil des itérations pour vérifier que le réseau s'approche d'un minimum local (ou global) et ne dérive pas vers une configuration explosive ou incohérente. Cette approche s'apparente à un suivi d'**énergie libre** dans la physique statistique.

Synthèse

Les **métriques** et **indicateurs** décrits se répartissent donc en trois **niveaux** :

- Des **mesures de performance** sur la tâche ciblée (accès à l'évaluation via des labels ou un reward).
- Des **mesures de robustesse** (test de la plasticité, résilience, vitesse de réadaptation).
- Des **indicateurs** internes sur la **structure** du graphe Ω (densité, modularité, énergie).

Cette combinaison de critères fournit une **vision** complète de la qualité du DSL : à la fois son aptitude à produire un résultat pertinent, et sa capacité à **s'auto-réorganiser** intelligemment dans un environnement plus ou moins changeant.

1.9.5.2. Évaluation Qualitative

En plus des mesures quantitatives (section 1.9.5.1), l'analyse **qualitative** joue un rôle important pour juger la **cohérence**, la **lisibilité** et la **applicabilité** du Deep Synergy Learning (DSL) dans un contexte particulier. Deux approches complémentaires se dégagent : l'inspection visuelle et conceptuelle des **clusters** ou sous-réseaux (1.9.5.2.1), et la **validation** par un expert métier (1.9.5.2.2).

1.9.5.2.1. Inspection de la Formation de Clusters

Il s'agit d'examiner la **structure** du réseau Ω que le DSL fait émerger, afin de vérifier si les **macro-clusters** détectés possèdent une **cohérence** ou un **sens** pour la tâche ciblée. Cette inspection peut passer par des **visualisations** de graphes, réalisées à différents instants temporels pour repérer l'évolution des liaisons $\omega_{i,j}$. Les points essentiels concernent la **pertinence** et la **stabilité** de ces regroupements :

- Lorsqu'un cluster englobe plusieurs **capteurs** possédant une information similaire ou des données corrélées, on peut conclure que le DSL a spontanément réuni des entités “**redondantes**” ou “**complémentaires**”, ce qui témoigne d'une auto-organisation légitime.
- Si un cluster rassemble des **objets** ou des **documents** portant le même label (par exemple, un groupe d'images classées “chats”), on confirme qu'il existe une **signification** partagée, gage d'une mise en commun cohérente.
- La **dynamique** du cluster dans le temps peut être observée : un regroupement stable au fil de multiples itérations suggère une **robustesse** de la synergie, tandis qu'un cluster éphémère (qui disparaît aussitôt) révèle peut-être une coïncidence passagère ou un bruit local.

Dans des scénarios complexes (robotique multimodale, systèmes multi-agents), on peut produire des **cartes** ou des diagrammes du réseau Ω , annotés pour montrer comment certaines entités coopèrent fortement. On peut aussi calculer des **indices** de modularité (section 1.9.5.1.3) et mettre en avant les “communautés” du graphe, puis interpréter chacune de ces communautés à la lumière des données. Cette approche fournit un **retour** qualitatif sur la façon dont le DSL “perçoit” la répartition des entités et si cette répartition s'aligne sur des catégories connues ou pertinentes (type de capteur, fonctionnalité, classe d'images, etc.).

1.9.5.2.2. Études de Cas et Validation Experte

Dans certains domaines (notamment le **médical** ou l'**industriel**), la **validation** la plus décisive s'effectue par des **experts** humains : un médecin, un ingénieur, un manager. Cette évaluation s'éloigne des simples scores globaux et porte sur la **compréhension** et la **fiabilité** des solutions que le DSL propose.

On peut ainsi présenter au médecin les **groupements** de patients que le DSL a générés, en indiquant les **variables** (imagerie, analyses, symptômes) qui ont constitué les liaisons synergiques. L'expert examine alors la **cohérence** médicale des clusters, vérifie si un groupe correspond par exemple à une pathologie connue ou à un sous-type de maladie. Il juge la **valeur** diagnostique de ce sous-réseau et peut estimer dans quelle mesure ces regroupements pourraient enrichir ou accélérer la prise de décision. De même, dans le contexte industriel, on peut montrer comment le DSL a réparti des **machines**, des **ordres** de production ou des **flux** logistiques en différents macro-clusters, et demander à un ingénieur de vérifier la compatibilité pratique de ces regroupements ou leur pertinence opérationnelle.

Cette procédure d'**étude de cas** s'avère cruciale pour la **crédibilité** du DSL : même si celui-ci obtient un certain score numérique en classification ou en détection d'anomalies, il est indispensable de s'assurer que l'ensemble du réseau ne résonne pas selon des associations arbitraires. Une bonne **cohérence** empirique, confirmée par un observateur expert, renforce la confiance dans la méthode et prépare le terrain pour une adoption plus large au sein des applications critiques.

1.9.5.3. Validation en Continu

Le **Deep Synergy Learning (DSL)** se distingue par sa **mise à jour** permanente des pondérations $\omega_{i,j}$ (sections 1.4.5 et 1.7.1), ce qui appelle une approche de **validation** plus proche du flux **temps-réel** que d'une simple évaluation ponctuelle. Les sections suivantes (1.9.5.3.1 et 1.9.5.3.2) décrivent deux modalités complémentaires : le suivi itératif de la performance et de la structure, et la mise en place d'un **monitoring** susceptible de déclencher des alertes ou des ajustements.

1.9.5.3.1. Suivi de l'Évolution Itérative

Le **DSL** s'échelonne par mises à jour successives : à chaque itération, on incrémente $\omega(t)$ vers $\omega(t+1)$. Une évaluation **en continu** consiste alors à :

(a) Mesurer la performance courante.

Si le DSL vise une **tâche** (classification, contrôle robotique, recommandation), on calcule la métrique de référence (accuracy, reward, F1-score, etc.) sur un lot de validation ou un flux test. Cela offre une **courbe** de performance au fil des itérations, identifiant si le système converge vers une solution stable, ou s'il dérive.

(b) Analyser la structure du graphe.

On peut observer la **densité** du réseau Ω (nombre de liens actifs), la **modularité** (clusters bien séparés ou non), la stabilité interne (combien de liens conservent une valeur élevée). Une forte fluctuation entre $\omega(t)$ et $\omega(t+1)$ peut signaler un système encore loin d'un état stationnaire. À l'inverse, une quasi-constance indique une situation de quasi-convergence.

(c) Enregistrer et tracer la courbe d'évolution.

Chaque itération (ou chaque bloc d'itérations) est loggée : on stocke la valeur de la performance, de certains indicateurs structurels (taille moyenne des clusters, énergie $J(\Omega)$, etc.). Cette **traçabilité** est cruciale pour repérer quand le DSL a atteint un plateau ou si la performance se dégrade subitement (concept drift).

(d) Critère d'arrêt ou de poursuite.

On peut décider de stopper l’itération si la variation entre $\omega(t)$ et $\omega(t + 1)$ devient négligeable (convergence), ou si la **performance** ne progresse plus au-delà d’un certain seuil. Dans certains cas, on maintient le DSL actif indéfiniment (scénarios industriels, robotique), réagissant aux modifications de l’environnement.

Cette boucle itérative prévient d’éventuels phénomènes de sur-adaptation (réseau devenant excessivement dense) ou de sous-adaptation (faible densité liée à un paramétrage trop timide). Une **analyse** des logs au fil du temps met en évidence la vitesse de convergence et la **résilience**.

1.9.5.3.2. Monitoring et Alertes

Dans une application critique (médicale, industrielle, décisionnelle), une simple observation n’est pas toujours suffisante : un **monitoring** plus formel veille à la **cohérence** et la **sécurité** du réseau Ω (voir 1.7.6). Ce monitoring inclut plusieurs indicateurs :

Stabilité de la structure.

On calcule la variance de ω entre deux instants (par exemple $\|\Omega(t + 1) - \Omega(t)\|$) ou d’autres métriques signalant un réseau trop instable.

Performance “de garde”.

Si le DSL doit maintenir un certain niveau d’efficacité (taux de bonne détection, ou de reward dans un RL), un déclenchement d’alerte se produit si cette mesure chute sous un seuil critique.

Explicabilité ou lisibilité.

Dans des domaines réglementés, on peut surveiller le **degré** d’interprétabilité des clusters (proportion de clusters ayant un label clair, composition stable, etc.). Une dérive vers des regroupements ininterprétables peut exiger une intervention ou un recalibrage.

Lorsque le **monitoring** constate une dérive, un **module** hiérarchique ou un **expert** humain peut intervenir : ajuster certains paramètres η , τ , forcer la coupure de liens jugés incohérents, ou ajouter une pénalisation de densité. Cette supervision légère assure que le DSL ne bascule pas dans des configurations indésirables tout en conservant son **auto-organisation** locale (sections 1.7.4 et 1.7.6).

Conclusion partielle (1.9.5.3)

La **validation en continu** du DSL, alliant un **suivi itératif** (performance et structure au fil des mises à jour) et un **monitoring** plus formel (détection de dérives, alertes) complète la démarche expérimentale. Elle convient parfaitement à des environnements évolutifs (sections 1.9.4.1 et 1.9.4.2), où l’on veut attester, à la fois, de la **robustesse** du réseau et de sa **capacité** à se **reconfigurer** face aux changements. Cette évaluation en ligne se révèle cruciale pour maintenir la confiance dans le DSL, notamment dans des secteurs sensibles, et pour exploiter pleinement ses propriétés de **plasticité** et de **résilience**.

1.9.5.4. Comparaisons et Ablations

Au-delà de l’analyse spécifique (sections 1.9.5.1 à 1.9.5.3), il est souvent instructif de mener des expériences dites d'**ablation** ou de **comparaison** avec d’autres méthodes. Ces deux approches renforcent la compréhension de la dynamique d'**auto-organisation** du **Deep Synergy Learning (DSL)**, en explicitant son apport quant à la souplesse, la robustesse ou l’efficacité globale.

Un **protocole d’ablation** consiste à retirer, un à un, certains éléments constitutifs du DSL et à observer la dégradation (ou non) des performances. On peut notamment désactiver la **sparsification** (par exemple en n’appliquant plus le seuil ω_{\min}) pour constater si le réseau, alors très dense, perd en stabilité ou en temps de calcul. De même, on peut supprimer la **synergie n-aire** (sections 1.4.7 et 1.7.3.3) pour vérifier à quel point la coopération strictement binaire suffit ou, au contraire, omet des effets émergents. On peut enfin restreindre des mécanismes d’inhibition compétitive (s’ils ont été introduits), pour déterminer si le DSL s’emballe lorsqu’il n’existe plus de limitation sur la somme des pondérations

sortantes. Chaque ablation révèle la sensibilité du DSL à un composant particulier et met en évidence la **valeur** précise de ce composant pour la stabilité, la plasticité ou la précision finale.

On pratique également des **comparaisons** face à d'autres **algorithmes**. Dans un scénario de **clustering**, on confronte le DSL à k-means ou DBSCAN, afin de voir si la formation de **clusters** auto-organisés procure un gain ou une flexibilité supplémentaire. Dans un cadre plus “réseau de neurones”, on peut aligner le DSL contre un **autoencodeur** (pour la recherche d'une représentation non supervisée) ou un **Graph Neural Network** (GNN), et vérifier si la **mise à jour** $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$ du DSL excelle en termes d'adaptation continue par rapport aux GNN entraînés de façon plus classique. Enfin, lorsqu'il s'agit de **classification supervisée**, un **réseau CNN** ou un **Transformeur** sert souvent de référence : on examine la précision sur un dataset labelisé, mais l'on note également la souplesse d'adaptation en flux, critère dans lequel un DSL peut se révéler supérieur.

La **conclusion** d'un tel dispositif d'expérimentation apporte un **bilan** sur la **valeur ajoutée** du DSL. En règle générale, on met en avant la **gestion flexible** de l'hétérogénéité et l'**adaptation** en flux (1.5.4 et 1.9.4), ainsi que la **capacité** à fusionner différents modes de données ou à incorporer des règles symboliques (1.5.7), aspects qui font parfois défaut aux algorithmes de clustering strict ou à un pipeline neuronal figé. Toutefois, on peut aussi constater les **limites** du DSL, comme la **complexité** potentiellement élevée ou la sensibilité à certains paramètres (η, τ , mécanismes d'inhibition), ce qui amène souvent à conseiller une hybridation avec des méthodes existantes ou un usage parcimonieux. Au final, ces expériences de comparaison et d'ablation confirment dans quelles conditions l'**auto-organisation** synergiques (sections 1.4.5 et 1.7.1) se montre réellement avantageuse par rapport aux solutions standards, et dans quelles circonstances elle doit être associée à d'autres approches.

Conclusion partielle (1.9.5)

La **validation** du **Deep Synergy Learning** requiert une **méthodologie** adaptée :

- **Quantitative** : mesurer la performance sur des tâches (classification, recommandation...), la robustesse (adaptation à la perturbation), la cohérence structurelle (modularité, densité...).
- **Qualitative** : observer l'**émergence** de clusters, l'**interprétabilité** des regroupements, la **cohérence** avec un expert métier.
- **Continu** : suivre l'évolution temporelle, détecter la stabilité ou la dérive, éventuellement déclencher un **monitoring** ou un **contrôle** externe.
- **Comparatif** : confronter le DSL aux approches traditionnelles (clustering k-means, autoencodeur, CNN supervisé...) pour mettre en évidence ses avantages (auto-organisation, plasticité) et ses faiblesses.

Ce **protocole** de validation (statique vs. dynamique, quantitatif vs. qualitatif, supervisé vs. non supervisé) constitue le **socle** empirique garantissant que l'**auto-organisation** défendue par le DSL se traduit en **résultats** concrets, compréhensibles et robustes.

1.9.6. Collaboration et Partage de Ressources (Open Source)

Dans l'univers du **Deep Synergy Learning (DSL)**, la mise en commun des progrès, des outils et des expériences revêt une importance considérable. L'**auto-organisation** propre au DSL, qui nécessite une variété de données et de scénarios pour faire ses preuves, tire un large bénéfice de la mutualisation des travaux, sous la forme de codes sources, de jeux de données, de retours pratiques et de bonnes pratiques diffusées. Cette ouverture s'ancre généralement dans une démarche **Open Source**, profitable à la fois à la communauté académique et au secteur industriel.

1.9.6.1. Logique de Partage et de Convergence

Le **Deep Synergy Learning (DSL)** s'étend sur de multiples domaines (robotique, recommandation, fusion multimodale, raisonnement symbolique) et s'appuie sur des mécanismes d'**auto-organisation** (voir la dynamique de mise à jour de ω en section 1.4.5). Cette **variété** de terrains d'application favorise une collaboration étroite entre laboratoires de recherche, équipes industrielles et développeurs indépendants, dès lors que la logique d'**open source** est privilégiée.

Un **socle commun** de bibliothèques et de scripts peut faciliter la gestion du **Synergistic Connection Network**, en automatisant notamment le **stockage** et la **mise à jour** de Ω , ainsi que le calcul de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (sections 1.7.3, 1.9.3). Par exemple, il est possible de définir une interface unifiée pour :

- (i) L'insertion d'entités (\mathcal{E}_i) ou de nouveaux flux,
- (ii) Le calcul partiellement parallèle de $S(\mathcal{E}_i, \mathcal{E}_j)$,
- (iii) La gestion des pondérations $\omega_{i,j}(t)$, avec options de sparsification ou d'inhibition compétitive.

Cette mise en commun évite de réécrire, dans chaque équipe, toutes les routines nécessaires à la mise en place du DSL, et garantit une **cohérence** méthodologique. Elle contribue également à renforcer la **reproductibilité** : les chercheurs peuvent employer les **mêmes** scripts de base pour déployer des expérimentations, ce qui rend les comparaisons plus fiables et plus justes (sections 1.9.5.4 et 1.9.6.2).

Une telle **architecture partagée** profite à chacun. Les laboratoires peuvent se concentrer sur l'**innovation** propre, comme la définition de nouvelles **fonctions** de synergie (co-information avancée, synergie n-aire contextuelle), l'**extension** au renforcement distribué (section 1.8.3), ou l'**implémentation** de la fusion symbolique (section 1.5.7). Les industriels et ingénieurs peuvent expérimenter ces développements dans des contextes concrets (usine, data streaming, systèmes complexes) sans réinventer le framework. Les contributions open source, sous forme de *pull requests*, enrichissent alors le noyau DSL, étoffant progressivement les fonctionnalités (contrôle de la stabilité, logs de performance, routines de visualisation, etc.).

Ce **partage** systématique de code, de protocoles expérimentaux et de retours d'expérience suscite un mouvement de **convergence** : plus le socle DSL est **collectivement** éprouvé et amélioré, plus il gagne en maturité et en **fiabilité**. Cette confiance accrue facilite l'adoption dans des contextes délicats (diagnostic, robotique critique), car le code se révèle maintenu, testé et documenté, s'alignant ainsi sur le souci de **pérennité** et de **lisibilité** qui traverse l'ensemble de la logique d'auto-organisation.

1.9.6.2. Ressources Open Source et Espaces Collaboratifs

Pour favoriser un **écosystème** dynamique autour du **Deep Synergy Learning (DSL)**, il est généralement souhaitable de s'appuyer sur des **outils** et **plates-formes** facilitant la contribution et la mutualisation des efforts. Le déploiement d'un **dépôt central** et la mise en place d'espaces d'échanges entre développeurs, chercheurs et industriels constituent alors des atouts majeurs.

A. Dépôts de Référence

Un premier élément clé consiste à héberger le code source de base, les **scripts** et la **documentation** sur un ou plusieurs dépôts publics (GitHub, GitLab, ou autre). On y stocke :

- (i) Un "noyau" DSL

qui comprend la représentation des **entités** (\mathcal{E}_i), la **matrice** ou liste de liaisons $\omega_{i,j}$, la **règle** de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

(ou ses variantes), et éventuellement des modules pour l'**inhibition compétitive**, la **synergie n-aire**, la **parsimonie** (seuil ω_{\min}), etc.

(ii) Des “Exemples” concrets

montrant, par exemple, un petit prototype en Python reliant des entités d’imagerie à des entités symboliques, ou un mini-scénario de recommandation. On peut également inclure des scripts C++ de démonstration pour la robotique (cf. sections 1.9.3 et 1.9.4).

(iii) De la documentation

présentée sous forme de wiki ou de site statique, expliquant comment **installer** la bibliothèque, configurer les paramètres η, τ , lancer une **simulation** de référence, etc. Cette partie est essentielle pour encourager la participation et diminuer la courbe d’apprentissage.

B. Processus de Collaboration

Pour encourager une **communauté** à s’agréger autour du DSL, il est utile d’établir :

(i) Un workflow de type “pull request” et “issue tracker”

afin que tout contributeur puisse proposer du code (nouvelles fonctions de synergie, correctifs sur la mise à jour des poids) et recueillir des retours. Les “issues” permettent d’**organiser** les tâches (bogues signalés, demandes de fonctionnalités) et leur priorisation.

(ii) Des conventions de contribution

(coding style, test coverage minimum, documentation des nouvelles classes) qui uniformisent le projet et simplifient la revue de code.

C. Espaces d’Échange et de Discussion

En complément, l’ouverture d’un **forum** (Discourse, forum GitHub) ou d’un **canal** (Slack, Discord, mailing list) facilite les **discussions** plus informelles : aide à l’installation, optimisation de performances, retours d’expérience sur des expérimentations menées. C’est souvent dans ces canaux que naissent :

- Des **retours** d’utilisateurs rencontrant des problèmes concrets,
- Des **idées** d’extension (ex. prise en compte de la synergie conditionnelle, intégration avec un module RL),
- Des **retours** de performance sur différents dataset (voir section 1.9.2).

Conclusion sur l’Ouverture (1.9.6.2)

En centralisant le **noyau** DSL, les **exemples** et la **documentation** dans un **dépôt public**, et en offrant un **processus** de collaboration clair, la communauté peut consolider un **socle** solide, évolutif et reproductible. Des **espaces d’échange** conviviaux renforcent la mutualisation des idées, permettant à chaque domaine (vision, robotique, symbolique, recommandation) de **bénéficier** des avancées réalisées ailleurs. Cette dynamique open source est un levier essentiel pour accélérer le développement du DSL, accroître la **qualité** des implémentations et structurer les efforts de recherche vers une **convergence** méthodologique.

1.9.6.3. Partage de Données et Scénarios de Test

Dans la logique d’une **dynamique auto-organisée** (sections 1.4.5 et 1.5.4), la flexibilité du **Deep Synergy Learning (DSL)** s’apprécie pleinement lorsqu’il est confronté à des **situations** diverses, combinant parfois plusieurs modalités (flux sensoriels, données textuelles, logs industriels, etc.) ou évoluant en continu (scénarios streaming, concept drift). Pour mettre à l’épreuve cette caractéristique, il s’avère crucial que la **communauté** partage ouvertement des **jeux de données**, ainsi que des **scénarios** ou **pipelines** illustrant l’intégration de ces données dans un contexte DSL.

A. Jeux de Données Spécialisés

Le **DSL** n'a de sens que si l'on peut tester sa capacité à **fusionner** différentes entités et à s'**auto-réorganiser**. Les bases de données monomodales traditionnelles (MNIST, CIFAR, etc.) fournissent un point de départ utile (section 1.9.2), mais pour démontrer la **polyvalence** du DSL :

- (i) Données multimodales ou multi-sensorielles

pour la robotique (caméras, LIDAR, IMU), la recommandation (profils utilisateurs, attributs de contenus, historique de navigation), ou des flux audio–texte (transcription).

- (ii) Données multi-agents

(par exemple, en sociologie ou en économie) permettant de mettre en évidence la formation de **clusters** auto-organisés entre acteurs, ressources, règles symboliques.

- (iii) Données industrielles ou logs

(ex. relevés de capteurs, machines, alertes), parfois anonymisées ou synthétiques, pour respecter la confidentialité.

Pour chaque jeu de données, il importe de fournir un **descriptif** clair (entités présentes, types de variables), afin de faciliter la configuration des entités $\{\mathcal{E}_i\}$ et de la fonction de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$.

B. Pipelines et Protocoles de Test

Au-delà de la simple distribution d'un **dataset** statique, l'**auto-organisation** du DSL se révèle surtout dans des **scénarios** plus dynamiques :

- (i) Simulateurs ou environnements “streaming”

(voir section 1.9.4.2) qui injectent des paquets de données selon une chronologie, permettent de provoquer des pannes (capteurs, modules) ou d'introduire de nouvelles entités, etc.

- (ii) Scripts d'automatisation

définissant comment on “alimente” le DSL, les paramètres η, τ ou l'architecture de calcul (CPU, GPU). Cette standardisation augmente la **reproductibilité** et autorise la comparaison directe des résultats entre équipes.

- (iii) Mesures de performance et d'adaptabilité

(section 1.9.5) explicitant la façon dont on évalue la qualité du réseau $\Omega(t)$, la précision si c'est un problème de classification, la robustesse si on modifie la distribution, ou la vitesse de convergence.

Les **pipelines** peuvent inclure un format de configuration (fichier YAML, JSON) listant les hyperparamètres, la nature des entités, les modes de calcul de la synergie. On assure ainsi que d'autres chercheurs puissent répliquer l'expérience avec une autre implémentation du DSL.

C. Communication sur les Configurations et Paramètres

Les résultats d'une expérimentation DSL dépendent souvent de la **forme** de la fonction de synergie, des choix de **synergie n-air** ou binaire, de l'éventuelle compétition inhibitrice, etc. Pour favoriser la **transparence** et la **comparabilité**, il s'avère opportun de joindre à chaque partage de scénarios :

- (i) La liste précise de (η, τ) , ainsi que les constantes de régularisation (p. ex. α, ω_{\min}).

- (ii) Le dimensionnement en entités :

combien de nœuds initiaux, mode de création ou d'insertion au cours du temps, critère d'arrêt ou d'itérations maximales.

- (iii) Les détails de l'infrastructure matérielle :

CPU/GPU utilisés, bibliothèques (versions Python, C++).

En combinant ces **informations**, on garantit une **reproductibilité** qui, au fil du temps, crée un socle de confiance dans les performances du DSL. Cela s'apparente à la tradition du deep learning (ImageNet, COCO) : chaque publication fournit un protocole de formation, la taille des batchs, etc. De la même manière, le DSL, en perpétuel mouvement (mise à jour continue de Ω), tire un grand profit de ce partage de scénarios dynamiques et de configurations standardisées.

Conclusion (1.9.6.3)

L'émergence d'un **corpus** de jeux de données, de pipelines streaming et de protocoles simulant des pannes ou des évolutions de distribution, fait office de **benchmark** pour le DSL. Les chercheurs et industriels peuvent ainsi confronter leur implémentation, vérifier la **solidité** de leurs résultats, et affiner la **dynamique** du réseau synergiques. Ce partage contribue à forger un **écosystème** commun, à la fois riche, transparent et propice à l'innovation autour de l'auto-organisation.

1.9.6.4. Organisation de la Communauté

Dans une optique de pérenniser et de développer le **Deep Synergy Learning (DSL)**, la constitution d'une communauté active et ouverte apparaît essentielle. L'expérience des projets open source montre que la **vie** de ce réseau d'utilisateurs et de contributeurs dépend autant des **échanges** réguliers que de la disponibilité d'un socle technique commun. Les rencontres périodiques, la publication d'un changelog structuré et la mise en place de groupes de travail stratégiques sont autant de facteurs qui soutiennent une dynamique collaborative à long terme.

Il se révèle d'abord crucial de faciliter des **rencontres** ponctuelles, comme des workshops ou hackathons dédiés au DSL, au cours desquels ingénieurs, chercheurs et étudiants échangent autour de leurs implémentations. Ces sessions constituent l'opportunité de présenter des **retours d'expérience** à la fois sur les déploiements réussis et sur les limites rencontrées. Elles offrent aussi l'occasion de mettre en lumière des projets pilotes dans différents domaines (robotique, recommandation, traitement multimodal) et de dégager de nouvelles pistes de recherche ou d'extension. Les échecs ou les résultats négatifs y ont toute leur place, car ils contribuent à une compréhension plus fine des mécanismes d'**auto-organisation** et informent la communauté sur les pièges à éviter.

Un autre point important consiste à maintenir un **changelog** décrivant de manière transparente l'évolution du noyau DSL. Chaque introduction de fonctionnalité, chaque correction d'erreur, chaque modification de la formule de synergie ou du module d'inhibition compétitive doit y être consignée. Cette clarté historique facilite l'adoption des dernières versions, la résolution de problèmes de compatibilité et la validation des nouveaux apports. Les équipes qui s'appuient sur le DSL peuvent alors choisir en connaissance de cause de rester sur une version stable ou d'expérimenter avec la plus récente.

Il est tout aussi déterminant de former des **groupes de travail** qui se spécialisent dans des thématiques données, comme l'implémentation GPU, la gestion de la **synergie n-aire**, ou l'intégration de composantes symboliques (voir la section 1.5.7). Chaque groupe peut définir son plan d'action, mutualiser le code et s'accorder sur des normes de contribution, assurant la cohérence dans la progression des fonctionnalités critiques. Cette forme d'organisation permet à la communauté de répartir efficacement les efforts sur des points stratégiques et de garantir que l'évolution d'un module (par exemple la fusion multimodale) n'entre pas en conflit avec d'autres pans du DSL.

La réussite de ce dispositif repose souvent sur une équipe ou un consortium central ayant l'initiative de lancer et de coordonner le **dépôt principal**, de modérer les discussions sur le forum, ainsi que de fusionner les contributions au noyau. Ce noyau principal gagne en stabilité et en visibilité au fur et à mesure que de nouveaux utilisateurs l'adoptent, créant ainsi une base de plus en plus robuste. Les projets satellites, plugins ou extensions peuvent alors se greffer en conservant un rattachement clair au socle commun. Grâce à ce mécanisme, un véritable *écosystème DSL* peut émerger, à l'image de ce que l'on observe dans d'autres communautés open source, enrichissant en continu les possibilités d'application et la solidité du modèle d'**auto-organisation**.

1.9.6.5. Bénéfices sur le Long Terme

Le fait de privilégier une **démarche open source** et de mutualiser les outils, les protocoles et les retours d’expérience dans la mise en œuvre du **Deep Synergy Learning (DSL)** se traduit par plusieurs avantages structurants pour la communauté et la technologie elle-même.

Un premier **bénéfice** réside dans l'**amélioration** de la **confiance**. La disponibilité du code source, le partage d’expériences et l’ouverture sur les scénarios d’évaluation diminuent la défiance liée à une potentielle “boîte noire” : le fonctionnement du **réseau** Ω , son initialisation, la mise à jour des poids ω et les paramètres η, τ peuvent être examinés en détail, ce qui rassure à la fois les chercheurs, les développeurs et les industriels. Cette transparence favorise l’adoption du DSL dans des milieux sensibles (industrie, médical, infrastructures critiques), où l’on exige une traçabilité et une lisibilité des mécanismes d’apprentissage.

Un second **bénéfice** découle de l'**accélération** de l'**innovation**. Les équipes de recherche n’ont plus à repartir de zéro pour implémenter la structure du **Synergistic Connection Network** ou la logique d'**auto-organisation** (voir sections 1.7.3 et 1.9.3). Elles peuvent s’appuyer sur un socle commun, éprouvé sur plusieurs scénarios, et se concentrer sur l’ajout de nouvelles fonctions de **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (corrélations avancées, co-information n-aire, etc.), sur l’intégration symbolique (section 1.5.7), ou encore sur la mise en place de synergies conditionnelles. Cette mutualisation accélère le **cycle de développement** et assure une qualité plus homogène, puisque tout le monde profite des correctifs et optimisations déjà validés.

La **couverture** de différents domaines s’en trouve **étendue**. En mettant en commun leurs jeux de données (section 1.9.6.3) et leurs scripts de simulation, les divers acteurs identifient plus vite les **limites** du DSL (complexité potentiellement élevée, instabilité en l’absence de mécanismes d’inhibition, etc.). Ils signalent également quels correctifs ou quelles extensions sont urgents, par exemple la nécessité d’une version distribuée pour grande échelle, ou la gestion de la synergie n-aire dans un cadre plus sophistiqué. Cette boucle vertueuse amène une augmentation du nombre de cas d’utilisation reconnus et un renforcement progressif de l’infrastructure.

De plus, on **favorise l’interopérabilité** en adoptant un format unifié pour décrire la **matrice** Ω et les **entités** $\{\mathcal{E}_i\}$. Un tel standard facilite l'**enchâssement** direct d’un module CNN ou d’un transformeur pour extraire des features sub-symboliques (section 1.8.2), la connexion à un bloc symbolique (section 1.5.7), voire l’implémentation d’un schéma d’apprentissage par renforcement distribué (section 1.8.3). Cette capacité à s’**intégrer** facilement dans d’autres pipelines IA ou dans des frameworks simulant des environnements complexes suscite un engouement croissant et assure l’évolution continue du paradigme DSL. Les chercheurs et industriels y trouvent une base solide d’expérimentation et de développement, tout en sachant que chaque progrès effectué profite à la communauté dans son ensemble.

Conclusion sur la Collaboration et le Partage de Ressources

La démarche de **collaboration en open source** constitue un atout majeur pour l’adoption, la validation et la diffusion du **Deep Synergy Learning**. En publiant du code, des scénarios de test, des benchmarks et des retours d’expérience, les communautés académiques et industrielles créent un **écosystème** qui, à l’image de ce qui s’est construit autour du deep learning, soutient un cycle d’innovation collective. Dans ce processus, le DSL profite non seulement d’une visibilité accrue, mais également d’une capitalisation rapide des solutions aux problèmes de complexité, de stabilité, de multimodalité, etc. En retour, chaque utilisateur potentiel dispose d’une base solide pour tester et adapter le DSL à ses propres défis, créant une dynamique d’enrichissement mutuel où la **philosophie** d’auto-organisation du DSL trouve un écho dans la *co-organisation* de la communauté autour du partage open source.

1.9.7. Gestion du Cycle de Vie d’un Projet DSL

Mettre en œuvre le **Deep Synergy Learning (DSL)** ne consiste pas seulement à développer un prototype ou un algorithme : c’est un **processus** continu, jalonné d’étapes depuis la conception initiale jusqu’à la maintenance, en passant par le prototypage, l’intégration et l’évolution du système. Cette section propose une vue d’ensemble de la

progression typique d'un projet DSL, en soulignant les spécificités liées à l'auto-organisation et à la dynamique des liens $\omega_{i,j}$.

1.9.7.1. Identification des Objectifs et Faisabilité

Il est essentiel, avant d'engager un **projet** autour du **Deep Synergy Learning (DSL)**, de déterminer avec précision les **objectifs** ciblés et d'estimer la **faisabilité** d'une mise en œuvre. On peut, dans un premier temps, répertorier les scénarios d'usage dans lesquels on anticipe une plus-value notable de l'**auto-organisation** (cf. sections 1.5.4 et 1.9.4). Il s'agit de vérifier si l'on souhaite prioriser la **fusion** de plusieurs **modalités** (vision, audio, texte), la gestion d'un **flux** non stationnaire (dérive de distribution, arrivée de nouvelles entités) ou encore l'**inclusion** d'un composant symbolique (règles, concepts abstraits) à l'intérieur du Synergistic Connection Network.

Dans le cas d'une application robotique, on s'interroge sur la manière dont le DSL profitera à la **plasticité** sensorimotrice. Si l'on se trouve plutôt dans une problématique de recommandation, on évalue la **capacité** du DSL à organiser simultanément un grand nombre d'entités (utilisateurs, contenus, contextes) sans retomber dans une explosion combinatoire (voir la discussion sur la parcimonie en section 1.7.1). Dans le domaine du **diagnostic médical**, on estime, avant de lancer un projet conséquent, si les données disponibles et les **contraintes** réglementaires (exigence d'explicabilité, confidentialité) sont compatibles avec l'approche **auto-organisée**.

Une fois ces scénarios précisés, il est judicieux d'établir un **inventaire** des ressources. On vérifie la disponibilité de **jeux de données** (ou de flux de capteurs), ainsi que la présence de simulateurs (cf. section 1.9.4.2) ou d'experts métier pour valider les configurations émergentes. On s'assure que les contraintes de **performance** (temps de calcul, déploiement éventuel sur GPU ou en cluster) et de **qualité** (robustesse, convergence) puissent être satisfaites avec un noyau DSL. Si des restrictions de confidentialité existent, on prévoit des mécanismes pour anonymiser ou agréger certaines entités (section 1.7.6).

Ces étapes initiales aboutissent à un **protocole** de validation, dans lequel on fixe si l'on compte évaluer le DSL sur un **dataset statique** (déetecter la formation de clusters ou la qualité d'une tâche supervisée) ou dans un **cadre dynamique** (arrivée de nouvelles entités, perturbations). Il arrive qu'on adopte un protocole partiellement supervisé, où seulement quelques labels sont disponibles et où la **synergie** auto-organise la majorité des entités (section 1.5.5). Les conditions d'évaluation et les métriques correspondantes (sections 1.9.5.1 et 1.9.5.2) sont ainsi établies pour déterminer clairement le **rôle** de l'auto-organisation et la **valeur ajoutée**, comparativement à un algorithme plus classique (réseau neuronal fixe, clustering standard). Lorsque la feuille de route s'arrête à ce point, on dispose déjà d'une **faisabilité** claire, point de départ pour les phases de conception concrète du DSL.

1.9.7.2. Prototypage Initial et Paramétrage

Lorsque les objectifs et la **faisabilité** (section 1.9.7.1) sont clarifiés, on entreprend la **conception** d'un prototype qui traduira, dans le concret, les fondements du **Deep Synergy Learning (DSL)**. Dans une première phase, il importe de choisir le **langage** et le **framework** adaptés : on peut opter pour Python si le projet met l'accent sur la **prototypage rapide** (bénéficiant des bibliothèques NetworkX, PyTorch, etc.), ou privilégier C++ afin de viser des **performances** supérieures, notamment pour des systèmes temps réel ou des graphes de grande envergure (section 1.9.3).

Une fois la pile logicielle définie, la **mise en œuvre** s'orchestre autour de quelques briques essentielles. Il s'agit tout d'abord de décrire le **Synergistic Connection Network (SCN)**, où chaque **entité** \mathcal{E}_i peut être stockée sous forme d'objets ou de structures, et où les **liaisons** $\omega_{i,j}$ peuvent être représentées par une **matrice** Ω de dimension $n \times n$ (ou une **liste d'arêtes** si l'on souhaite davantage de parcimonie). La **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ s'implémente comme un composant modulaire, qu'il s'agisse d'une similarité, d'une distance, ou d'un calcul plus évolué basé sur une co-information (sections 1.4.4 et 1.5.4). On code ensuite la **règle de mise à jour** locale qui réalise l'incrément :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

éventuellement accompagnée de mécanismes d'**inhibition compétitive** ou de **sparsification** (sections 1.7.3 et 1.7.1).

Pour **valider** les fondements, il est judicieux de commencer par un **jeu de données** ou un **environnement** minimaliste : un dataset à basse dimension (par exemple quelques vecteurs synthétiques) ou un micro-simulateur. Cette étape de test répond à plusieurs interrogations. D'abord, on cherche à savoir si la **mise à jour** des pondérations ω aboutit à une **configuration** stable ou oscillatoire. On surveille également la **formation** de clusters cohérents, indicatrice de la bonne mise en place de la logique d'auto-organisation. Enfin, on peut introduire une légère perturbation (modification de quelques données, bruit aléatoire) pour juger la **robustesse** du système.

Au cours de ce **prototypage**, on procède à des réglages **empiriques** sur les **paramètres** η (taux d'apprentissage) et τ (terme de régulation ou d'oubli), ainsi que sur la façon de gérer la **parsimonie** (seuil ω_{\min} , normalisation locale...). Des **métriques internes** (ex. densité du graphe, modularité) guident l'optimisation pour éviter l'explosion du nombre de liens ou pour forcer la convergence des clusters. Des **métriques externes** (ex. exactitude d'un cluster vis-à-vis d'un label, temps de réadaptation lors d'une perturbation) éclairent la qualité de la solution par rapport à la tâche ciblée (classification, détection d'anomalies, etc.).

C'est souvent grâce à ces **itérations** successives — allant d'un jeu de données simple vers un usage plus complexe — que l'on obtient un prototype **fiable**, prêt à être étendu à des scénarios plus ambitieux (multimodaux, dynamiques, ou à grande échelle).

1.9.7.3. Passage à l'Échelle et Validation Approfondie

Après avoir démontré la **cohérence** du prototype sur un cas réduit (voir la section 1.9.7.2), il devient possible de l'étendre à un **dataset** ou un **simulateur** de taille plus importante. Cette montée en échelle amène toutefois des **enjeux** de complexité, de validation et d'intégration avec d'autres composants, qu'il convient de gérer avec méthodologie.

Un premier aspect concerne la **scalabilité**. Dans le **Deep Synergy Learning (DSL)**, la quantité de liaisons potentielles entre entités peut en principe croître de manière quadratique $O(n^2)$, dès que n augmente, où n est le nombre d'entités $\{\mathcal{E}_i\}$. Pour limiter cet écueil, les chercheurs ou développeurs mettent généralement en place des **heuristiques** de sparsification (seuil ω_{\min}), des **voisinages** (on autorise la synergie uniquement pour des paires (i, j) spatialement ou sémantiquement proches), ou un **échantillonnage** partiel des paires à chaque itération (voir sections 1.7.1 et 1.7.3). On peut aussi segmenter le réseau en **sous-groupes** si le domaine s'y prête, permettant de répartir le calcul en différents blocs. L'objectif est de **contenir** le coût en calcul et mémoire, tout en préservant l'essence de la mise à jour auto-organisée.

En parallèle, la **validation** se déploie selon les protocoles décrits en section 1.9.5. Sur le plan **quantitatif**, des métriques comme la précision (accuracy), le F1-score, l'ARI (Adjusted Rand Index) ou encore l'AUC sont utilisées si la tâche est reliée à un objectif supervisé ou semi-supervisé. Pour le **clustering** non supervisé, on peut calculer la NMI (Normalized Mutual Information) ou la modularité du graphe pour quantifier la qualité de l'organisation interne. Dans un cadre **dynamique**, on observe aussi comment le **Synergistic Connection Network (SCN)** se comporte lorsqu'on introduit de nouvelles entités, ou lorsque la distribution se modifie (concept drift), en relevant notamment le temps d'adaptation. On peut pousser plus loin la **validation qualitative** en inspectant la formation de **clusters** importants, en interrogeant des experts pour juger leur cohérence ou leur utilité pratique (sections 1.9.5.1 et 1.9.5.2).

On pratique ensuite une **comparaison** avec d'autres méthodes pour mesurer la plus-value de l'auto-organisation. Dans des scénarios statiques, on peut mettre le DSL en regard de réseaux neuronaux classiques (CNN, MLP) ou d'algorithmes de clustering (k-means, DBSCAN). Dans un environnement évolutif ou multimodal, on peut tester la plasticité du DSL en le confrontant à des pipelines où chaque modalité est traitée de façon indépendante et fusionnée tardivement. C'est l'occasion de voir si le DSL gère mieux les perturbations (pannes de capteurs, arrivée d'utilisateurs inconnus en recommandation, etc.) grâce à sa mise à jour locale et continue. De plus, lorsque l'usage d'un **module** de renforcement ou de règles symboliques fait partie de la feuille de route (sections 1.5.7 et 1.8.3), on peut progressivement **intégrer** ces briques additionnelles afin de vérifier que la dynamique du **réseau** Ω conserve sa stabilité tout en donnant accès à des fonctionnalités avancées (apprentissage de politiques, raisonnement logique local).

Cette **phase** de montée en échelle et de **validation approfondie** est donc le moment de consolider l'architecture du DSL, de prouver sa robustesse dans des contextes d'application variés, et d'obtenir une vision claire des conditions dans lesquelles la synergie adaptative surpassé ou complète favorablement les solutions existantes. Si les résultats se montrent convaincants, il devient possible d'envisager un déploiement plus large, ou de contribuer à l'effort

communautaire (sections 1.9.6.2 et 1.9.6.4) afin de partager les améliorations et les leçons issues de cette expérience d'intégration à grande échelle.

1.9.7.4. Intégration dans un Système Existant

Lorsque l'on souhaite exploiter le **Deep Synergy Learning** (DSL) dans un environnement industriel ou dans une architecture déjà déployée, il est crucial de concevoir l'**insertion** du DSL à l'intérieur d'un **pipeline** plus vaste. Dans un premier temps, il convient de définir les **interfaces** par lesquelles le réseau d'**entités** $\{\mathcal{E}_i\}$ va recevoir des données nouvelles (capteurs, événements) et restituer ses décisions ou regroupements. Cette interface se matérialise le plus souvent par un **bus** de messages (streaming en temps réel) ou par l'appel d'une **API** spécifique, selon que l'application se situe dans un système de robotique, de recommandation ou d'analyse de données.

Une fois les points d'entrée et de sortie identifiés, il faut régler la **fréquence** et la **logique** selon lesquelles la mise à jour des pondérations $\omega_{i,j}$ sera déclenchée. Dans certains scénarios critiques (par exemple un contrôle robotique temps réel), on peut opter pour une mise à jour en **continu**, réalisée en parallèle et soumise à des contraintes de latence. Dans d'autres usages plus batch, comme un traitement périodique de logs ou de données marketing, on procédera à une itération de l'**auto-organisation** toutes les heures ou tous les jours, en accumulant préalablement de nouveaux exemples. L'important est de veiller à ce que la boucle d'**adaptation** (où l'on calcule la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ et met à jour ω) n'entre pas en conflit avec d'autres modules, en particulier si des contraintes temps réel ou de cohérence globale sont imposées.

Il est également essentiel de mettre en place un mécanisme de **surveillance** et de suivi, afin de détecter rapidement les dérives ou difficultés éventuelles. Un tableau de bord (monitoring) peut fournir des **indicateurs** tels que la densité moyenne du graphe, la part de liens saturés, l'éventuelle oscillation des pondérations ou un score de performance sur un sous-ensemble de validation. Si certains seuils sont dépassés (par exemple si la densité $\text{dens}(\Omega)$ devient trop élevée, ou si la qualité d'un cluster chute subitement), un **module superviseur** intervient et peut influer sur les paramètres η , τ ou forcer la suppression de liens obsolètes. Il s'agit d'éviter la survenue de boucles chaotiques ou de saturation excessive, comme expliqué dans les sections sur la stabilité et la parcimonie (1.7.4 et 1.7.1). Dans un cadre d'entreprise, ce superviseur peut être un opérateur humain, un service DevOps ou une couche logique plus hiérarchique (1.5.7).

Cette **approche** collaborative, où le DSL vit au cœur d'un système plus large, engage souvent plusieurs services et départements : ingénierie de la donnée, experts métier, équipes chargées de la supervision logicielle. L'**auto-organisation** propre au DSL ne doit pas rester isolée, car elle impacte la façon dont les flux de données sont traités et dont les décisions (ou les regroupements) influent sur d'autres processus en aval. L'architecture globale bénéficie alors de la **plasticité** adaptative du DSL, tout en conservant un pilotage central ou semi-centralisé pour garantir la cohérence et la sécurité du système d'ensemble.

1.9.7.5. Maintenance, Évolution et Ajustements Continus

Une particularité fondamentale du **Deep Synergy Learning (DSL)** réside dans son **caractère évolutif** : le réseau Ω ne demeure pas figé une fois l'entraînement initial achevé, mais peut continuer à se **reconfigurer** de manière plus ou moins permanente, suivant les données ou flux rencontrés. Cette situation implique une **maintenance** d'un genre différent de celle d'un **réseau neuronal** classique, dont les poids sont souvent considérés comme définitifs après la phase d'entraînement. On peut distinguer plusieurs volets critiques dans la **sustainability** d'un projet DSL :

Un premier volet touche au **cycle d'évolution** du SCN. Tant que de nouvelles données parviennent (capteurs, utilisateurs, contextes inédits), le mécanisme d'**auto-organisation** ne s'arrête pas. Il est alors utile de prévoir des **processus de nettoyage** et d'archivage pour éviter qu'au fil des itérations, le **Synergistic Connection Network** ne gonfle trop (accumulation de liens $\omega_{i,j}$ obsolètes), n'explose en complexité, ou ne s'égare dans des configurations contradictoires. Cela se traduit couramment par l'application périodique de **règles** de parcimonie ou d'inhibition, qui suppriment les liaisons de force inférieure à ω_{\min} (section 1.7.1) ou coupent les noeuds inactifs.

Un second volet a trait au **réglage** de la **plasticité**. Dans un contexte où la distribution des données peut changer de manière brutale (concept drift, nouvelle population d'utilisateurs, conditions de capteurs modifiées), on doit contrôler

le degré d'adaptabilité du réseau. Si les paramètres η (taux d'apprentissage) et τ (terme de régulation) ne sont pas ajustés, on risque que le DSL ne s'adapte pas assez vite à un changement majeur (réseau “trop rigide”), ou qu'au contraire il vire dans un **état instable** (chaotique, oscillatoire). La bonne pratique consiste à surveiller des **indicateurs** de stabilité (variance de Ω au fil du temps, proportion de clusters qui se recomposent subitement) et de **performance** (scores sur un flux de validation, si disponible). Selon ces retours, on rehausse la valeur de η pour accélérer l'oubli d'anciens liens ou on resserre la régulation τ pour freiner les changements.

Un troisième volet a trait à l'**ajout** ou au **retrait** d'entités. Dans le DSL, chaque **nouveau capteur** (ou tout autre module) est introduit comme une entité inédite $\mathcal{E}_{\text{nouvelle}}$. On doit alors concevoir un **mode opératoire** pour l'**insertion** : on connecte l'entité $\mathcal{E}_{\text{nouvelle}}$ à un petit voisinage initial (ex. un certain ensemble d'entités proches selon une distance ou un critère symbolique), puis la **dynamique** $\omega_{i,\text{nouvelle}}$ s'occupe de renforcer ou d'affaiblir les liens. À l'inverse, la **déconnexion** d'un capteur ou d'un module obsolète exige la suppression de l'entité associée et de toutes ses liaisons actives, en vérifiant l'impact éventuel sur les clusters restants. Dans certains domaines, il est bon de purger régulièrement les entités dépassées ou inactives, qui n'apportent plus de synergie mais pourraient encombrer la structure.

Enfin, il existe des cas où on pratique des **phases de réinitialisation partielle** du réseau. Lorsque le SCN semble saturé, trop lourd ou confus, on peut planifier une **opération** de recuit simulé (section 1.7.3.2) ou de re-sparsification générale, faisant office d’“**événement de restructuration**”. Cette procédure ressemble à la notion de “**sommeil**” ou de “**réorganisation**” dans la métaphore biologique : on diminue temporairement η , on augmente la pénalité ou la température d'un recuit, ce qui détruit certains clusters stériles et stabilise les synergies essentielles. L'objectif, dans le cadre d'une exploitation industrielle ou d'un cycle de vie long, est de **préserver** la réactivité et la lisibilité du réseau DSL, en conjuguant un apprentissage continu avec une politique proactive d'entretien et de rééquilibrage.

1.9.7.6. Documentation, Capitalisation et Partage

Au fur et à mesure que le **projet DSL** progresse, il génère un ensemble de **matériaux** précieux : configurations paramétriques, historiques de performance, codes et scripts, notes concernant la stabilité et la scalabilité, ainsi que des **ensembles** de règles ou de synergies n-aires spécifiques à un certain domaine. Il est essentiel d'organiser ces informations de manière méthodique afin de permettre une **capitalisation** à long terme. Dans une approche de type “**Ingénierie**” ou “**DataOps**”, on encourage :

46. La **tenue** d'un journal de modifications (un **changelog**) recensant les modifications majeures apportées au Synergistic Connection Network (SCN), aux fonctions de synergie ou aux paramètres $\{\eta, \tau\}$. Cette traçabilité facilite la reproduction d'expériences passées et l'explicitation des gains (ou régressions) qui surviennent quand on modifie un composant clé.
47. L'élaboration d'un **guide d'utilisation**, sous forme de documentation technique ou d'un wiki, détaillant le fonctionnement du prototype DSL. On y décrit la **structure** du code (où est géré le graphe Ω , quelle est la procédure de mise à jour des liens, comment lancer les tests sur un dataset, etc.), les **principaux** pièges (risque d'explosion du nombre de liens si on n'active pas la sparsification) et les **méthodes** de monitoring (visualisation des clusters, vérification de la stabilité). Les nouveaux membres de l'équipe, ou d'autres structures (internes ou externes), peuvent ainsi **reprendre** les travaux sans tout réapprendre par tâtonnements.
48. La **communication** des retours d'expérience sous forme de rapports ou de **publications** (blogs techniques, articles de conférences, documents internes). Ces supports expliquent les objectifs initiaux, les réussites, les difficultés rencontrées (mise à l'échelle, réglages de la plasticité, insertion de règles symboliques), ainsi que les solutions adoptées. Dans l'esprit du partage **open source** (voir section 1.9.6), cet effort de diffusion profite à la **communauté DSL** : les utilisateurs extérieurs gagnent en confiance et en motivation pour expérimenter, tout en bénéficiant d'un “modèle” de déploiement.

En définitive, cette démarche de **documentation** et de **partage** assure qu'on ne perde pas la **connaissance** accumulée au fil des itérations, et qu'on s'inscrive dans un cycle vertueux : la **consolidation** des acquis mène à une communauté plus robuste et stimulante, améliorant simultanément la fiabilité et l'implémentation du **Deep Synergy Learning** dans divers secteurs.

Conclusion sur la Gestion du Cycle de Vie d'un Projet DSL

La création et l'exploitation d'un **SCN** auto-organisé ne se limitent pas à une phase d'entraînement ponctuelle : elles exigent un **cycle** complet, depuis l'**idéation** (pourquoi opter pour le DSL ?), le **prototypage** (valider la logique de base), la **mise à l'échelle** (passage à des datasets plus vastes, intégration avec d'autres modules), jusqu'à la **maintenance** continue (monitoring, adaptation aux évolutions de l'environnement, gestion des entités nouvelles ou obsolètes). Chaque étape réclame un ensemble de bonnes pratiques et d'outils, ainsi qu'une collaboration étroite avec les experts métier et l'équipe de R&D. En fin de parcours, la **documentation** et la **capitalisation** (éventuellement partagée en open source) garantissent que le savoir-faire accumulé ne se perd pas et que d'autres projets pourront s'en inspirer, contribuant à leur tour au **développement** d'un écosystème DSL pérenne et dynamique.

Chapitre 2 : Fondements Théoriques et Origines du DSL

2.1. Genèse Historique et Inspirations Multidisciplinaires	213
2.1.1. Les Premières Pistes d'Auto-Organisation	213
2.1.2. SOM, Hopfield et la Génération des Réseaux Associatifs.....	216
2.1.3. Entre Neurosciences Computationnelles et Physique Statistique	221
2.1.4. Précurseurs de la Synergie Multimodale	226
2.1.5. Origine du DSL et Ses Premiers Manifestes	230
2.2. Principes Mathématiques de Base.....	237
2.2.1. Définition des Entités et de la Fonction de Synergie	237
2.2.2. Mise à Jour des Pondérations : Formule Générale	244
2.2.3. Règles de Parsimonie et Seuil de Connexion	249
2.2.4. Notions d'États Internes et Auto-Organisation	254
2.2.5. Exemples Illustratifs.....	260
II. Exercices et problèmes.....	269
2.3. Hypothèses de Stabilité et Émergence de Clusters.....	277
2.3.1. Point Fixe, Attracteurs : Définitions et Existence	277
2.3.2. Attracteurs Multiples et Oscillations.....	285
2.3.3. Analyse de la Formation de Clusters.....	292
2.3.4. Influence du Bruit et des Perturbations	298
2.3.5. Limites Théoriques et Questions Non Résolues	302
2.4. Raccords avec la Physique Statistique et la Théorie des Systèmes Dynamiques .	309
2.4.1. Systèmes Dynamiques Discrets ou Continus	309
2.4.2. Parallèles avec les Modèles de Spin (Ising, Potts, Hopfield).....	316
2.4.3. Notion d'Énergie ou de Fonction Potentielle	322
2.4.4. Rôle de l'Inhibition et de la Saturation Synaptique	329
2.4.5. Approches Hybrides et Collaboration Interdisciplinaire	334
2.5. Perspectives Historiques et Liens avec l'IA Moderne	340
2.5.1. Évolutions Récentes et Convergence avec le Deep Learning.....	340
2.5.2. Extensions Symboliques et Neuro-Symboliques	343
2.5.3. Apport en Robotique et Contrôle Adaptatif.....	349
2.5.4. Comparaison avec d'Autres Paradigmes (RL, GNN, etc.).....	354
2.5.5. Transition vers les Chapitres Suivants	357

2.1. Genèse Historique et Inspirations Multidisciplinaires

Le **DSL** n'est pas né ex nihilo. Il s'inscrit dans une **trajectoire historique** plus large, marquée par les travaux en **cybernétique**, la recherche sur la **plasticité neuronale** et les premières tentatives de construire des systèmes capables de s'organiser sans supervision stricte. L'objectif de ce sous-chapitre (2.1) est de dégager les **facteurs historiques** et les **influences** majeures qui ont contribué à l'émergence progressive de la notion d'auto-organisation, pour mieux comprendre le contexte dont le DSL se réclame.

2.1.1. Les Premières Pistes d'Auto-Organisation

Les premières tentatives pour expliquer comment un ensemble d'éléments (neurones, agents, mécanismes cybernétiques) peut parvenir à une **organisation** cohérente sans chef d'orchestre remontent aux années 1940-1960. Elles se sont nourries des hypothèses issues de la **cybernétique** et des premiers constats sur la **plasticité synaptique** dans le cerveau. Les sections qui suivent (2.1.1.1 et 2.1.1.2) se concentrent sur cette phase pionnière, préalable essentiel à l'éclosion de l'apprentissage auto-organisé.

2.1.1.1. Rôle de la Cybernétique et des Premiers Travaux sur la Plasticité Neuronale

Dans les années 1940, la **cybernétique**, introduite par **Norbert Wiener**, s'est concentrée sur l'étude des **boucles de rétroaction** présentes dans les systèmes vivants ou mécaniques. L'enjeu consistait à comprendre la manière dont un organisme, ou un dispositif artificiel, parvient à **réguler** son comportement en réagissant aux informations provenant de l'environnement. Dans ce cadre, plusieurs chercheurs ont rapidement émis l'hypothèse qu'un **réseau** de neurones ou de capteurs, guidé par des règles locales, pourrait atteindre une forme d'**organisation interne** sans exiger de commande globale.

Au sein de cette vision, la perspective de **Norbert Wiener** a mis en évidence que la rétroaction (feedback) pouvait autant stabiliser qu'instabiliser un système, selon la nature et l'orientation de la boucle. Cette découverte a suscité l'idée de concevoir un **réseau adaptatif**, entendu comme un ensemble d'unités x_i soumises à des liaisons $\omega_{i,j}$. La question se posait alors : si chaque liaison s'ajuste localement, pourrait-il se produire un ordre global émergent ? Cette interrogation s'inscrit dans le droit fil des principes de la **cybernétique** : un réseau pourrait, de façon autonome, moduler ses connexions pour converger vers un comportement collectif stable ou cohérent, sans qu'une autorité centrale vienne en spécifier toutes les interactions.

En parallèle, dans les années 1950 et 1960, les premiers **travaux** sur la **plasticité neuronale** se sont développés. Des expériences en neurosciences laissaient supposer que les **connexions synaptiques** – c'est-à-dire les poids entre neurones – se modifiaient selon l'activité partagée. Cette

théorie d'apprentissage local, illustrée par les écrits de **Donald Hebb** (1949), énonçait le principe suivant : lorsque deux neurones \mathbf{x}_i et \mathbf{x}_j s'activent simultanément, la liaison $\omega_{i,j}$ se renforce. De façon mathématique, on peut schématiser ce phénomène par

$$\Delta\omega_{i,j} \propto x_i x_j,$$

où x_i et x_j désignent l'intensité d'activation de neurones i et j . Cette découverte suggérait déjà la possibilité d'une **auto-organisation** : un réseau de neurones où les pondérations se réajustent selon la co-occurrence, conduisant progressivement à la formation de motifs ou de regroupements stables.

La rencontre de la **cybernétique** et de la **neurosciences computationnelle** a ainsi ouvert la voie à des formalismes mathématiques permettant de décrire comment un **réseau** – qu'il s'agisse de neurones biologiques ou d'unités artificielles – peut se configurer de lui-même à travers des mises à jour strictement **locales**. Les principes de la rétroaction et de la plasticité synaptique convergeaient pour affirmer qu'un ordre global peut naître de simples règles d'ajustement, sans supervision. Cette conviction, déjà présente dans les prototypes informatiques modestes de l'époque, a servi de socle aux développements qui suivront : l'idée de "synergie" entre unités, chère au **Deep Synergy Learning**, hérite directement de la plasticité hébbienne et des mécanismes de contrôle cybernétique.

Bien que les moyens techniques fussent, à l'époque, limités, les écrits fondateurs évoquaient déjà la **stabilisation** d'un réseau autour de certaines combinaisons privilégiées d'activités. On y pressentait la formation de **clusters** ou d'**assemblées** neuronales, précurseurs des théories d'**attracteurs** et de **cartes auto-organisées**. C'est dans ce terreau scientifique – porté par l'intérêt pour les systèmes **auto-régulés**, la découverte de la **plasticité** synaptique et le contexte de la **cybernétique** – qu'allait naître l'ambition de construire des réseaux véritablement **adaptatifs**, capables de s'**organiser** à partir de règles élémentaires. Le **Deep Synergy Learning**, bien plus tard, s'inscrira dans cette continuité : la mise à jour d'un poids $\omega_{i,j}$, la quête d'une **co-opération** locale et l'idée qu'un **ordre** global peut émerger de multiples interactions locales découlent étroitement de ces prémisses.

2.1.1.2. Naissance de l'Idée d'Organisation sans Superviseur dans Certains Laboratoires (années 1950–1960)

L'**apprentissage machine** des premières décennies, tel qu'illustré par le perceptron de Frank Rosenblatt (1958), se focalisait principalement sur des mécanismes **supervisés** reposant sur des *labels* explicites et un signal d'erreur orientant la mise à jour des poids. Cependant, dès la fin des années 1950, plusieurs laboratoires ont amorcé des recherches explorant la possibilité d'**organiser** les **réseaux** ou les **unités neuronales sans** recours à la supervision directe. L'intuition sous-jacente à ces travaux, influencée par la **cybernétique** et la biologie, était de laisser les entités neuronales, désignées par \mathbf{x}_i , s'**auto-régler** par des **règles locales**, en se fondant sur la redondance ou la régularité intrinsèques des **données d'entrée**.

A. Expériences Préliminaires issues de la Cybernétique

À la suite des réflexions initiées par Norbert Wiener, certains chercheurs se sont inspirés du principe de **rétroaction** pour imaginer des systèmes neuronaux capables de se **corriger** et de **s'ajuster** en l'absence d'un enseignant. Les premiers prototypes, bien que rudimentaires, comportaient un **petit réseau** de neurones soumis à des flux sensoriels. Les **pondérations** internes évoluaient progressivement en suivant des règles élémentaires de renforcement : si deux unités \mathbf{x}_i et \mathbf{x}_j coïncidaient fréquemment dans leur activation, le lien $\omega_{i,j}$ se trouvait augmenté, formalisant ainsi un renforcement local. Ce mécanisme rappelle, dans une version simplifiée, l'idée d'une **co-occurrence** statistique où, mathématiquement, on peut écrire

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \kappa(\mathbf{x}_i(t), \mathbf{x}_j(t)),$$

avec η un taux d'apprentissage local et $\kappa(\cdot, \cdot)$ une fonction soulignant la co-activation simultanée des neurones \mathbf{x}_i et \mathbf{x}_j .

B. Émergence du Concept d'Auto-Clusterisation

De premiers rapports internes (dans des laboratoires tels que le MIT ou l'Université Stanford) évoquaient la possibilité de **regrouper** automatiquement les données ou les neurones, même si le terme “clustering” n'était pas toujours formalisé de la sorte. Le concept de “résonance” ou de “coopération neuronale” commençait à circuler, pour désigner la faculté de certains sous-ensembles d'unités à présenter un comportement **synchronisé**. Cette synchronie, renforcée par un schéma de mise à jour local, tendait à **former** ce qu'on nommerait plus tard un **cluster**. On pressentait déjà qu'en exploitant uniquement la distribution des données — c'est-à-dire la *fréquence* et la *corrélation* des activations neuronales —, il était possible de dégager une structure latente sans qu'un label de sortie intervienne.

C. Rupture par rapport aux Méthodes Supervisées

À la différence du perceptron de Rosenblatt, qui nécessitait un **signal d'erreur** pour orienter la correction des poids, ces laboratoires s'intéressaient à des systèmes pouvant **auto-exploiter** la simple densité ou co-activation des données d'entrée. D'un point de vue algorithmique, on sortait donc du paradigme

$$w_{\text{nouveau}} = w_{\text{ancien}} - \eta \frac{\partial E}{\partial w},$$

car la fonction d'erreur E était absente ou du moins implicite ; seule la **co-occurrence** neuronale venait guider l'ajustement. Ces expériences soulignaient le potentiel d'une **catégorisation spontanée**, c'est-à-dire la découverte de regroupements (catégories) sans recourir à un enseignant.

D. Exemples de Règles Locales de Mise à Jour

Les discussions tournaient souvent autour du fait que la **corrélation** d'activation entre deux neurones \mathbf{x}_i et \mathbf{x}_j pouvait être utilisée comme critère de renforcement. Ainsi, la “loi de Hebb”, régulièrement citée dans ces travaux, posait qu'une activation simultanée doit se traduire par un **renforcement** de la connexion. Certains laboratoires introduisaient aussi une **inhibition latérale**, destinée à redistribuer les ressources et à éviter qu'un seul sous-réseau ne monopolise

l'apprentissage. Formellement, on pouvait ajouter un **terme d'inhibition** proportionnel à la somme des activations concurrentes, conduisant à un rééquilibrage permanent :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[\mathbf{x}_i(t) \mathbf{x}_j(t) - \beta \sum_k \mathbf{x}_i(t) \mathbf{x}_k(t) \right],$$

où β est un paramètre d'inhibition, et la partie $\sum_k \mathbf{x}_k(t)$ reflète la compétition entre neurones rivaux.

E. Bases d'une Future Formalisation

Ces travaux pionniers n'ont pas immédiatement conduit à une théorie unifiée, en raison du manque de ressources computationnelles et de la jeunesse du formalisme mathématique. Toutefois, ils ont jeté les bases d'une **vision** selon laquelle la **donnée** en elle-même, pour peu qu'elle soit soumise à un réseau présentant un mécanisme de renforcement local, peut induire une **organisation spontanée**. L'histoire de l'IA retiendra que la fin des années 1950 et le début des années 1960 ont vu la genèse des prémisses de l'**apprentissage non supervisé** et de l'**auto-organisation**, lesquelles s'affirmeront au cours des décennies suivantes dans des réalisations plus formelles, telles que les **cartes de Kohonen (SOM)** ou d'autres algorithmes de clustering non supervisé. Les fondements de cette dynamique trouvent aujourd'hui leur prolongement dans des approches modernes comme le **Deep Synergy Learning**, où les principes de **cohésion** et de **coopération** locale se traduisent par des **règles de mise à jour** autonomes et contextuelles, assurant la formation de sous-structures pertinentes sans l'intervention d'un enseignant externe.

2.1.2. SOM, Hopfield et la Génération des Réseaux Associatifs

Alors que la **cybernétique** et les premiers travaux d'auto-organisation “sans superviseur” (2.1.1) ont semé les graines d'une plasticité neuronale locale, la recherche sur les **réseaux associatifs** et les **Self-Organizing Maps (SOM)** dans les années 1970-1980 a considérablement renforcé l'idée qu'un système pouvait **découvrir** des structures en l'absence de labels explicites. Les modèles de Teuvo Kohonen (SOM) et de John Hopfield (réseaux associatifs) illustrent deux facettes majeures de l'auto-organisation : la **formation spontanée de cartes topologiques** et l'**association mémorielle** à travers des attracteurs stables. On verra dans ce sous-chapitre (2.1.2) comment ces approches, tout en se développant indépendamment, ont préparé le terrain pour des paradigmes plus récents tels que le **Deep Synergy Learning (DSL)**. Il est également intéressant de souligner que l'origine même du DSL, portée notamment par l'ingénieur en informatique Mohammed Bouayoun, puise en partie dans ces traditions d'apprentissage non supervisé et de réseaux associatifs adaptatifs.

2.1.2.1. Teuvo Kohonen : Self-Organizing Maps (SOM) et Topologie Émergente

Les **Self-Organizing Maps (SOM)**, introduites par Teuvo Kohonen au début des années 1980, ont exercé une influence considérable dans l'**apprentissage non supervisé**. Leur principe repose sur la capacité d'un **réseau** à projeter un espace d'entrées (souvent de grande dimension) sur une **grille bidimensionnelle**, tout en conservant autant que possible les **relations de proximité** entre les

données. Cette méthode constitue un jalon historique de l'**auto-organisation**, démontrant de manière tangible qu'un réseau neuronal peut découvrir des **structures** dans des données sans requérir de labels explicites.

A. Préservation de la Topologie

L'idée fondamentale est de préserver la **topologie** des entrées dans l'espace de sortie. Les unités ou "neurones" de la carte sont disposés sous forme de grille (par exemple, deux dimensions), chaque neurone \mathbf{n}_i possédant un vecteur de poids \mathbf{w}_i . À chaque présentation d'un vecteur d'entrée $\mathbf{x} \in \mathbb{R}^d$, le neurone **gagnant**, souvent noté \mathbf{n}_{BMU} , est celui pour lequel la distance

$$\| \mathbf{x} - \mathbf{w}_{\text{BMU}} \|$$

est minimale. Les neurones voisins, définis par un **voisinage** décroissant au fil des itérations, sont **ajustés** pour se rapprocher de \mathbf{x} . Cette mise à jour peut s'exprimer, pour un neurone \mathbf{n}_j situé à proximité du gagnant \mathbf{n}_{BMU} , sous la forme :

$$\mathbf{w}_j(t+1) = \mathbf{w}_j(t) + \alpha(t) \mathcal{H}_{\text{BMU},j}(t) [\mathbf{x}(t) - \mathbf{w}_j(t)],$$

où $\alpha(t)$ est un taux d'apprentissage, et $\mathcal{H}_{\text{BMU},j}(t)$ un **facteur de voisinage**, généralement une fonction gaussienne de la distance entre \mathbf{n}_j et \mathbf{n}_{BMU} dans la grille. Par ce mécanisme, la carte se "**déforme**" localement pour accueillir l'information, créant ainsi une **continuité** qui reflète la structure sous-jacente des données.

B. Apprentissage Auto-Organisé sans Labels

Le procédé s'inscrit clairement dans un **cadre non supervisé**. En l'absence de labels, la SOM repère des regroupements dans l'espace d'entrée tout en imposant une **cohérence topologique** sur la grille de sortie. De sorte qu'au fur et à mesure des itérations, le réseau s'organise de lui-même en attribuant à chaque neurone \mathbf{n}_j une zone de compétence. Les régions de la SOM finissent par représenter des **catégories** ou des **familles** de vecteurs. Il s'agit là d'un exemple marquant de la capacité d'un réseau à apprendre la structure latente des données sans information externe. Dans l'histoire de l'auto-organisation, ces travaux prolongent la philosophie décrite en section [2.1.1](#) au sujet des premiers prototypes sans superviseur.

C. Impact sur la Vision de l'Auto-Organisation

Les cartes de Kohonen ont fait la démonstration qu'une **cartographie continue** des données pouvait émerger, d'une façon parfois rapprochée de la **corticotopie** observée chez l'animal ou l'humain (aires sensorielles). Des laboratoires contemporains ont développé en parallèle d'autres approches, comme les réseaux à attracteurs (Hopfield) ou les modèles neuronaux de Shun-Ichi Amari, élargissant encore la diversité des méthodes. L'importance historique des SOM tient à leur présentation didactique et à leur succès pour illustrer le **concept** de structure cognitive émergente. Elles ont ainsi balisé le chemin pour des algorithmes modernes d'**auto-organisation** dans les réseaux neuronaux.

D. Liens Conceptuels avec le DSL

Le **Deep Synergy Learning (DSL)**, décrivant lui aussi un schéma d'ajustement auto-organisé des **liaisons** entre entités d'information, partage certains points communs avec la SOM. Dans le cas

d'un SOM, la notion de **voisinage** sert de force de coordination locale. De façon similaire, au sein du DSL, la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ joue le rôle de régulateur pour décider du renforcement ou de l'affaiblissement d'un lien $\omega_{i,j}$. Les SOM peuvent être considérées comme un modèle relativement **statique**, puisqu'elles imposent une grille préfixée et un voisinage décroissant dans le temps. Le DSL, en revanche, s'applique de manière plus **généraliste**, en laissant le réseau se **reconfigurer** en continu via des règles de mise à jour basées sur la synergie, ce qui autorise une plasticité plus large, y compris dans la **topologie** même des liens. Cependant, l'esprit est identique : aucune **supervision** n'est requise pour faire émerger un **ordre** dans la distribution des entités.

E. Vers des Extensions Multimodales et la Synergie

En s'appuyant sur la faculté des SOM à regrouper des signaux divers, certains chercheurs ont considéré des **applications multimodales** (ex. signaux audio et images) dans lesquelles une carte conservait une **cohérence** entre différentes modalités. Bien que ces applications soient restées assez limitées en pratique, elles ont jeté les bases d'une réflexion sur la fusion de données hétérogènes et sur la possibilité qu'un réseau puisse **apprendre** à associer plusieurs sources. Cette logique se retrouve dans le DSL, qui pousse plus loin l'idée de **fusion coopérative** : en permettant à n'importe quelles entités de développer des liens flexibles selon leur **score de synergie**, il devient envisageable d'intégrer de multiples modalités de façon plus souple qu'une grille 2D imposée.

Conclusion

Les **Self-Organizing Maps** de Kohonen représentent un tournant déterminant dans la formalisation de l'**auto-organisation**. Elles ont démontré la possibilité, pour un réseau, de cartographier sans labels explicites un vaste espace de données en une structure de sortie exploitable. L'importance pédagogique et les succès pratiques de la SOM ont entretenu le goût pour les systèmes **non supervisés** dans la communauté scientifique. Les idées directrices — réglage local, émergence de clusters, topologie imposée ou auto-imposée — ont ensuite nourri d'autres paradigmes à vocation plus générale, en particulier le **Deep Synergy Learning**, qui étend la notion de proximité et l'adapte à la formation continue de réseaux libres de leurs connexions et de leur organisation. Le SOM fut ainsi l'une des premières démonstrations solides de la **capacité** d'un réseau à **auto-structurer** la représentation de son environnement.

2.1.2.2. John Hopfield : Mémoires Associatives, Attracteurs et Minima d'Énergie

Les travaux de **John Hopfield**, menés au début des années 1980, ont introduit un modèle neuronal emblématique désigné sous le nom de **réseau associatif** ou “Hopfield network”. Cette proposition, qui s'inspire d'idées analogues à celles de la **physique statistique** (systèmes de spins, modèles d'Ising ou de Potts), a mis en lumière la possibilité qu'un **réseau** formé d'unités simples puisse se stabiliser dans des **configurations** représentant des “mémoires” ou des **attracteurs**. La théorie démontre que, grâce à une série de **règles locales** d'activation et de mise à jour des poids, le réseau finit par adopter spontanément un **état** ordonné à partir d'une situation initiale partielle ou bruitée.

Sur le plan conceptuel, un réseau Hopfield se compose de neurones ou d'unités binaires, interconnectés de façon quasi complète, avec des poids $\omega_{i,j}$. Les poids sont déterminés de manière à ce que certains vecteurs cibles \mathbf{x}^* deviennent des **états stables** : si le système est initialisé dans une configuration voisine de \mathbf{x}^* , alors la dynamique conduira à une convergence vers ce même

vecteur stable. Cette capacité illustre la notion de **mémoire associative**, où le réseau est en mesure de “retrouver” un motif mémorisé à partir d’informations incomplètes. La convergence résulte de la descente vers un **minimum d’énergie**, ce qui se formalise par une fonction $E(\mathbf{x})$ dite “énergie de Hopfield”, dont la décroissance au fil des itérations atteste de la stabilisation du réseau. On peut l’écrire, par exemple, comme

$$E(\mathbf{x}) = -\frac{1}{2} \sum_{i \neq j} \omega_{i,j} x_i x_j$$

Lorsque $x_i \in \{-1, +1\}$ sont les états des neurones.

L’idée d’associer une “énergie” à chaque configuration, empruntée aux modèles de spins en physique, a contribué à la compréhension de la **dynamique neuronale** : l’organisation globale émerge de l’itération de règles locales, chaque unité cherchant à minimiser le niveau d’énergie via l’actualisation de son état. Les mémoires stockées correspondent à des **minima locaux** de l’énergie, appelés **attracteurs**, vers lesquels converge le système si l’état initial se trouve dans leur bassin d’attraction. Cette observation a conforté le principe selon lequel un ordre collectif peut naître en l’absence d’un **superviseur** extérieur, du fait même de l’interaction **coopérative** entre unités.

D’un point de vue **auto-organisation**, le réseau de Hopfield n’exige pas de labels explicites pour récupérer un motif partiellement effacé ou bruité. Les poids peuvent être calculés par des règles associatives simples (souvent de type Hebb), mais une fois installés, la dynamique interne se déroule sans aide extérieure. Cette démarche illustre la logique d’**émergence** : la configuration finale du réseau, perçue comme la “mémoire récupérée”, ne se laisse pas dicter par une consigne imposée à chaque étape, mais relève plutôt d’une évolution **naturelle** vers un état stable. On rejoint ici les idées qui seront approfondies dans des approches contemporaines d’**apprentissage non supervisé** ou d’**auto-organisation** (voir section [2.1.2.1](#) pour la SOM de Kohonen).

Les liens avec le **Deep Synergy Learning (DSL)** peuvent être soulignés. Dans un réseau Hopfield, on observe une **coopération** entre unités guidée par l’énergie globale, et la convergence dans un état stable évoque la stabilisation d’un cluster de neurones. De façon analogue, dans le DSL, la présence d’une **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ joue un rôle comparable : lorsque deux entités \mathcal{E}_i et \mathcal{E}_j partagent une interaction jugée élevée, leurs liens se renforcent et le réseau évolue vers des groupements cohérents. Les mécanismes d’ajustement des pondérations, dans le DSL comme dans le modèle de Hopfield, traduisent une dynamique **auto-organisée** conduisant à l’apparition d’**attracteurs collectifs** ou de **sous-structures** stables.

La perspective de **minima d’énergie** s’applique dans le DSL lorsqu’on étudie la convergence des poids $\omega_{i,j}$ vers des configurations privilégiées. Bien que la formulation soit plus générale dans le cas du DSL (synergies n-aires, multimodalité, structure libre du réseau), le point essentiel demeure : un ensemble de règles locales peut suffire à faire émerger un **ordre global**, sans supervision externe. Le modèle de Hopfield constitue donc un **antécédent** conceptuel crucial, montrant comment des processus neuronaux simplifiés, hébergeant une logique énergétique, peuvent conduire à l’apparition de représentations stables. Cet héritage se retrouve dans la conception du DSL, où la “synergie” prolonge la notion d’affinité ou de corrélation, et où la plasticité multientités s’interprète comme une “generalisation” de la mémoire associative à des contextes et des modalités plus complexes.

2.1.2.3. Comparaison Partielle avec la Logique “Hebbienne”

Les **Self-Organizing Maps** de Kohonen et les **réseaux associatifs** de Hopfield, analysés précédemment dans les sections 2.1.2.1 et 2.1.2.2, illustrent deux approches phares de l'**apprentissage non supervisé** centrées sur la découverte de *patterns* ou de configurations stables en l'absence de labels explicites. Ces travaux s'inscrivent dans une continuité historique marquée par la **logique “Hebbienne”**, introduite dès la fin des années 1940. Le principe énoncé par Donald Hebb évoque que deux neurones qui s'activent conjointement renforcent leur connexion. La présente section (2.1.2.3) discute les liens qui unissent ces trois orientations (SOM, Hopfield, Hebb) et montre en quoi elles partagent des fondements communs tout en se distinguant dans leur implémentation et leur formalisme.

A. Fondements de la Règle de Hebb

Au cœur de la théorie de Donald Hebb, exposée dans *The Organization of Behavior* (1949), se trouve l'idée qu'un lien entre deux neurones \mathbf{x}_i et \mathbf{x}_j se renforce dès lors que leurs activations se produisent en simultané. L'intuition peut se formaliser par une **règle de mise à jour** locale de la forme

$$\Delta\omega_{i,j} \propto \mathbf{x}_i \mathbf{x}_j,$$

où $\omega_{i,j}$ désigne la connexion synaptique entre les neurones i et j . L'esprit est qu'une activation conjointe déclenche un **renforcement**, alors qu'une activation dissociée n'engendre pas d'augmentation (voire induit une réduction) de cette liaison.

B. Similarités avec les SOM et les Réseaux Associatifs

Les **Self-Organizing Maps** (SOM) recourent à un mécanisme de renforcement lorsqu'un neurone “gagne” face à un vecteur d'entrée. Il y a un rapprochement entre le neurone vainqueur \mathbf{n}_{BMU} et le vecteur d'entrée, ainsi qu'un ajustement pour les neurones voisins dans la grille. Cette mise à jour rappelle l'idée de co-activation : le neurone et la donnée se retrouvent “associés”, ce qui ressemble, dans un cadre discret, à la logique “hebbienne” de corrélation locale. De même, les **réseaux associatifs** de Hopfield adoptent une conception selon laquelle les poids $\omega_{i,j}$ reflètent la *corrélation* ou la *co-occurrence* entre les unités \mathbf{x}_i et \mathbf{x}_j . Lorsqu'on “enregistre” un motif, la force des liaisons associées augmente selon un principe analogue à la loi de Hebb, rendant possible la récupération dudit motif lorsque le réseau est présenté à un fragment ou à un état bruité proche.

C. Divergences et Élargissements

Ces trois approches conservent toutefois des particularités marquées. Les SOM, par exemple, imposent une **dimension topologique** à travers la grille 2D et la notion de “voisinage”, un élément que la loi de Hebb seule n'intègre pas de manière explicite. Dans un **réseau Hopfield**, la logique d'**énergie** et de **minima** assure la stabilité globale, tandis que la règle hebbienne se décline plutôt comme un simple *update local* $\Delta\omega_{i,j} \propto \mathbf{x}_i \mathbf{x}_j$. L'idée d'une fonction d'énergie globale, introduite par Hopfield, généralise l'intuition de la co-occurrence vers une perspective plus large d'attracteurs. Enfin, il existe une multitude de variantes “hebbiennes” qui ajoutent des mécanismes de normalisation, de saturation ou d'inhibition, permettant un **auto-équilibrage** ou une répartition

des ressources. Les SOM et les réseaux Hopfield constituent chacun une **implémentation** plus spécifiquement contrainte : préservation topologique pour les SOM, attracteurs d'énergie pour Hopfield.

D. Portée pour le Deep Synergy Learning

Le **Deep Synergy Learning (DSL)** perpétue l'héritage “hebbien” en privilégiant un **score de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ qui détermine si deux entités \mathcal{E}_i et \mathcal{E}_j renforcent ou affaiblissent leur lien $\omega_{i,j}$. Lorsque la synergie est élevée, on obtient une croissance de $\omega_{i,j}$ proche de

$$\Delta\omega_{i,j} \sim \eta[S(\mathcal{E}_i, \mathcal{E}_j)].$$

Cette démarche rappelle étroitement l'esprit de la règle de Hebb, tout en permettant des **généralisations** (par exemple, co-information n-aire, interactions complexes) qui dépassent la simple multiplication $\mathbf{x}_i \mathbf{x}_j$. Ainsi, le DSL embrasse la dynamique auto-organisée du type “hebbien” mais l'adapte à une variété de contextes, notamment multimodaux, et autorise la reconfiguration de la **structure du réseau**.

E. Conclusion sur la Continuité Historique

Le rapprochement entre les SOM, les réseaux Hopfield et la règle de Hebb met en évidence un **tronc commun** : la conviction que l'**auto-organisation** peut être guidée par la co-activation ou la co-occurrence d'entités, sans étiquettes supervisées. Les SOM instaurent une topologie manifeste, Hopfield introduit le formalisme de l'énergie, tandis que la loi de Hebb s'énonce comme un canevas local et minimaliste. Le **Deep Synergy Learning**, en s'inscrivant dans cette tradition, étend la logique hebbienne à des **fonctions de synergie** plus sophistiquées, conférant au processus d'apprentissage une plasticité encore plus large et un champ d'application élargi. De même, il hérite la philosophie d'**auto-régulation** non supervisée, où la **coordination globale** émerge de la simple itération de règles d'interaction locale. Les paradigmes SOM et Hopfield ont donc grandement contribué à forger un terreau conceptuel que le DSL exploite pour proposer une vision plus englobante de l'**auto-organisation** sans supervision explicite.

2.1.3. Entre Neurosciences Computationnelles et Physique Statistique

Après avoir examiné les apports de l'approche “sans superviseur” (2.1.1) et les modèles de réseaux associatifs ou de cartes topologiques (2.1.2), on constate que l'**auto-organisation** a également bénéficié d'un important soutien théorique et conceptuel depuis la **neurosciences computationnelles** et la **physique statistique**. Ces deux champs, bien que distincts dans leurs méthodes, ont convergé dans l'idée que des interactions **locales** entre unités (neurones, spins, etc.) peuvent générer des **motifs** ou des **structures** émergentes. Ce sous-chapitre (2.1.3) discute des influences conjointes de la dynamique neuronale (2.1.3.1) et des modèles de la physique hors équilibre, formant ainsi un socle pour de futures approches telles que le **Deep Synergy Learning (DSL)**.

2.1.3.1. Contribution des Modèles Biologiques (Dynamique des Neurones, Synapses)

Les avancées en **neurosciences computationnelles**, dès les années 1960–1970, ont exercé une influence considérable sur la compréhension et la formalisation de l'**auto-organisation** en

intelligence artificielle. Au-delà de la seule logique “hebbienne”, ces travaux se sont inspirés de mécanismes plus riches, tels que l'**inhibition latérale**, l'**homéostasie** ou encore les **modulations neuromodulatrices**. Les recherches d’Amari, de Grossberg et d’autres pionniers ont éclairé la manière dont la dynamique neuronale, régie par un ensemble d'**équations différentielles** ou de **règles itératives**, peut engendrer une **organisation spontanée** d’assemblées de neurones dans le cortex.

Un premier axe s’est centré sur la description mathématique de l'**activation neuronale**, souvent abordée via des champs neuronaux. Dans cette optique, on considère qu’un neurone x_i suit une équation de la forme

$$\frac{d x_i}{d t} = -\alpha x_i + \sum_{j \neq i} (w_{ij} f(x_j)) - \gamma g(x_i) + \theta_i,$$

où α est un terme de décroissance, $\sum_{j \neq i} w_{ij} f(x_j)$ représente la somme des influences (excitatriques ou inhibitrices) venant d’autres neurones, $\gamma g(x_i)$ un terme d’homéostasie ou de saturation globale, et θ_i un éventuel seuil ou un apport externe. Les valeurs w_{ij} constituent les **connexions synaptiques** entre neurones i et j . L’ensemble de ces équations, pris simultanément pour tous les neurones, fait apparaître des comportements comme l'**inhibition latérale** ou la **stabilisation** de patterns d’activité, propriétés fondamentales d’une auto-organisation neuronale.

Un second axe a porté sur la formation de **micro-assemblées neuronales** capables de se synchroniser et de se stabiliser de façon autonome, en particulier lorsque plusieurs neurones présentent une **co-activation** récurrente. Les neuroscientifiques ont documenté l’existence de tels groupes dans le cortex, où ils semblent représenter des stimuli spécifiques ou des fragments de concepts. Sur le plan algorithmique, on comprend que de simples règles de mise à jour locale des poids peuvent conduire à l’émergence de clusters : si deux neurones x_i et x_j s’activent souvent ensemble, alors la synapse w_{ij} se renforce, approximant la célèbre équation de type

$$\Delta w_{i,j} \propto x_i x_j.$$

Les neurones qui s’alignent ainsi sur la même fréquence de décharge forment une entité collective, ou **assemblée**, reflétant l’existence d’un pattern stable. C’est un mécanisme incontournable de la plasticité neuronale qui anticipe l’essentiel des principes d'**apprentissage non supervisé**.

Les modèles biologiques ont en outre démontré la nécessité de gérer la plasticité à plusieurs niveaux : la simple corrélation positive entre neurones, prolongement direct de la règle de Hebb, doit être modulée par des influences globales, par exemple l'**inhibition** généralisée (pour empêcher un groupe de neurones de devenir surdominant), ou l'**homéostasie** garantissant que l’activité moyenne du réseau se maintient dans une plage stable. Sur le plan formel, on peut introduire une équation supplémentaire qui régit l’évolution d’un paramètre global μ (un niveau d’homéostasie), que l’on couple à la dynamique neuronale :

$$\frac{d \mu}{d t} = \kappa(\bar{x} - \mu),$$

où \bar{x} représente l'activité moyenne du réseau. De telles considérations ont enrichi la formulation de la **plasticité synaptique**, en montrant qu'elle n'est pas uniquement dictée par la corrélation entre neurones, mais qu'elle peut être encadrée par des mécanismes correcteurs à grande échelle.

Cette perspective multi-niveaux et la logique de **rétroaction** entre neurones ont profondément marqué le développement de l'**auto-organisation** en IA. Les travaux décrits dans la section précédente (SOM, réseaux Hopfield, etc.) trouvent en effet leur fondement dans l'idée qu'un **ordre** global peut émerger d'interactions locales, inspirées des phénomènes biologiques. Les paradigmes d'apprentissage non supervisé, y compris le **Deep Synergy Learning (DSL)**, en hériteront largement. Dans ce dernier, la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ généralise la corrélation neuronale à des entités plus variées que des neurones biologiques, tout en gardant l'esprit de **cohésion** et de **co-activation**. On y retrouve, de surcroît, la possibilité de gérer des influences extérieures ou des régulations globales sur la distribution des pondérations $\omega_{i,j}$, s'apparentant à ces mécanismes d'homéostasie étudiés en neurobiologie.

Ces contributions biologiques éclairent la **logique** des règles de mise à jour et les **régularisations** nécessaires pour rendre un réseau de neurones, ou plus généralement un **réseau auto-organisé**, à la fois flexible et stable. L'inhibition latérale, la compétition, la notion d'assemblées synchronisées, et la multi-échelle de plasticité guident les solutions mises en œuvre dans de nombreux algorithmes d'auto-organisation. Comme expliqué en 2.1.3.2, la **physique statistique** apportera un complément de formalisation, en introduisant le concept de fonction d'énergie globale et de transitions de phase, pour appuyer cette thèse selon laquelle de simples lois locales peuvent faire naître une structure cohérente.

2.1.3.2. Apports de la “Synergetics” (Haken) et de la Thermodynamique Hors Équilibre

Le rôle de la **physique** dans la formalisation de l'**auto-organisation** s'est avéré tout aussi déterminant que celui des **neurosciences computationnelles** (cf. section 2.1.3.1). Les réflexions autour de la “**synergetics**”, initiées par Hermann Haken, et les avancées de la **thermodynamique hors équilibre**, mises en lumière notamment par Prigogine, ont considérablement enrichi la compréhension de la **structure émergente** dans des systèmes complexes. Ces idées, apparues dans les années 1970, ont permis de décrire la formation **spontanée** de motifs ou de clusters stables, en expliquant comment des interactions locales peuvent, par amplification mutuelle, donner naissance à un *ordre global*. Le **Deep Synergy Learning (DSL)** tire un profit conceptuel de ces travaux, en prolongeant le cadre d'analyse proposé par la synergetics pour rendre compte des dynamiques de pondérations et de synergies entre entités hétérogènes.

Un premier apport essentiel de la synergetics réside dans l'introduction de la notion de **paramètre d'ordre**, qui désigne une grandeur globale traduisant le passage d'un état désordonné à un état ordonné. Dans un système physique, ce paramètre se manifeste, par exemple, dans l'intensité du rayonnement laser ou la convection de Rayleigh-Bénard, où une **bifurcation** engendre l'émergence d'un motif spatial cohérent. D'un point de vue mathématique, l'auto-organisation se formalise ainsi :

$$\frac{d x_i}{dt} = F(x_i, \{x_j\}_{j \neq i}, \lambda),$$

où x_i représente l'état d'une unité i et λ est un **paramètre de contrôle**. Au-delà d'un certain $\lambda_{\text{critique}}$, une fluctuation initiale peut se retrouver amplifiée jusqu'à donner un **état global** ordonné, indiqué par l'évolution d'un paramètre d'ordre. Dans une perspective d'**apprentissage automatique**, cette situation correspond à la formation d'un **cluster** ou d'une **représentation collective**, résultant de boucles de rétroaction locales entre unités.

La **thermodynamique hors équilibre** étudie, quant à elle, les systèmes où un **flux** permanent d'énergie ou de matière empêche la stabilisation à un équilibre classique. Les travaux de Prigogine ont, par exemple, mis en avant la formation de **structures dissipatives**, capables de s'auto-organiser grâce à l'échange d'énergie avec l'environnement, comme le suggère l'apparition de vortex dans un fluide ou d'oscillations chimiques Belousov-Zhabotinsky. Le parallèle avec un **réseau** d'unités neuronales ou d'agents est naturel : si chaque entité subit un influx permanent de données ou de signaux, l'apparition de **clusters cohérents** peut être vue comme une transition hors équilibre, portée par une fonction d'**énergie** ou de **quasi-énergie**. Dans une telle vision, l'évolution d'un réseau adaptatif se conçoit à travers l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ prend la place d'un "moteur" non linéaire, apte à déclencher une **bifurcation** quand certaines liaisons excèdent un "seuil critique".

Cette analogie se trouve renforcée par l'**interprétation énergétique**. Dans la synergetics, on peut décrire un système par une fonction de potentiel (ou d'énergie libre) qui décroît au fur et à mesure qu'un ordre s'établit. De la même manière, le **DSL** assimile la dynamique des poids $\omega_{i,j}$ à la recherche d'un certain **ordre** (ex. configuration de clusters) reflétant un minimum de quasi-énergie ou un optimum local en matière de synergie. Le **bruit**, au sens physique, peut alors jouer un rôle crucial : un léger déséquilibre peut suffire à amorcer la croissance d'un cluster, et un niveau modéré de perturbation autorise le réseau à s'extraire de minima sous-optimaux.

Les **applications** de cette métaphore sont variées. Dans le **DSL**, dès lors qu'on interprète les entités comme des "particules informationnelles" interagissant via $S(i,j)$, la transition d'un état dispersé à un état **agrégé** (clusterisé) s'apparente à un passage de phase. Les liens $\omega_{i,j}$ s'accroissent jusqu'à constituer un sous-réseau stable, tandis que les connexions marginales disparaissent. Cette émergence graduelle peut faire l'objet d'une analyse semblable aux **transitions de phase** en physique, soulignant l'universalité d'un mécanisme auto-organisé où un ordre global naît d'un **feed-back** local.

Dans la conception qu'en donnent les promoteurs du **DSL**, dont Mohammed Bouayoun, on retrouve la volonté de **prolonger** l'héritage hakenien : rechercher un "**ordre stable**" (clusters ou attracteurs) à partir d'interactions **locales** successives, plutôt que d'imposer a priori une architecture rigide. Cette parenté s'illustre particulièrement bien lorsque la mise à jour des pondérations s'interprète comme une **minimisation** d'une forme d'énergie, associée à la diffusion des **synergies** $S(\mathcal{E}_i, \mathcal{E}_j)$. L'inspiration de la thermodynamique hors équilibre conduit, en outre, à admettre que de faibles perturbations ponctuelles peuvent déclencher de grandes réorganisations, gage d'une **plasticité** et d'une capacité à "trouver" l'ordre dans le désordre.

Ainsi, l'apport de la **physique statistique** et de la **synergetics** fournit à l'auto-organisation un **formalisme** qui accentue la notion de **transition** et de **paramètre d'ordre**. Les paradigmes

ultérieurs, comme le **Deep Synergy Learning**, reprennent les principes de boucles de rétroaction positive et d'amplification des synergies, en remplaçant les interactions spin par des **synergies** plus complexes adaptées à divers types d'entités (ex. vecteurs, symboles, flux temporels). La logique sous-jacente demeure toutefois la même : l'**ordre** émerge de multiples mises à jour locales, et le système peut franchir des paliers critiques pour se stabiliser en configurations cohérentes et **auto-organisées**.

2.1.3.3. Premiers Parallèles avec la Notion de Synergie dans un Réseau d'Entités

Les avancées exposées en **neurosciences computationnelles** (section 2.1.3.1) et en **physique statistique** (section 2.1.3.2) ont fait ressortir un principe commun, un **réseau** composé de multiples entités (neurones, spins, agents) peut s'auto-organiser via l'accumulation d'**interactions locales**. Cette double perspective – dynamique neuronale et thermodynamique hors équilibre – a conduit à employer le terme de “**synergie**” pour décrire la force ou l'intérêt mutuel reliant deux unités dans le réseau. Les points suivants (2.1.3.3.1 à 2.1.3.3.4) soulignent comment, dès ces premiers travaux, on peut rapprocher cette démarche de ce qui sera plus tard le **Deep Synergy Learning (DSL)**.

A. Passer du Vocabulaire de la Physique au Vocabulaire Neuronal

Dans les modèles Hopfield ou Ising, on définit une **énergie** globale mesurant la cohérence ou, à l'inverse, le désordre. Les entités y sont liées par des **couplages** ω_{ij} . De la même manière, la **synergetics** (Haken) postule qu'un **paramètre d'ordre** peut émerger lorsque les connexions se renforcent collectivement. On a alors suggéré de passer d'une lecture “énergie négative” ou “coupling” à la notion plus “positive” de **synergie**, où celle-ci exprimerait la “**plus-value**” constructive de la relation entre deux unités. En neurosciences, il est fréquemment question de “**gain**” ou d’“affinité” entre neurones, soulignant la co-activation productive. Dans la lignée de la plasticité synaptique, la co-occurrence neuronale a souvent été décrite comme un facteur de renforcement local, certains articles utilisant explicitement la terminologie “**co-information**” ou “**co-opération**”. Ainsi, les premières bases d'une “synergie” ont été posées par analogie avec la description de l'énergie dans les systèmes physiques.

B. Hypothèses de Lien Local-Global

Les théoriciens de la **synergetics** ont mis l'accent sur le fait que, si la “force” locale entre deux entités \mathcal{E}_i et \mathcal{E}_j est suffisamment élevée, elles tendent à se synchroniser ou à coopérer. Au fur et à mesure que ce phénomène se propage dans l'ensemble du réseau, on assiste à la formation d'un **état ordonné** (un cluster, un attracteur, un domaine de spins, etc.). La **coopération** locale est donc censée produire un **ordre global** via une accumulation de petites interactions. Cette dynamique se retrouve tant dans la formation d'assemblées neuronales (en neurosciences) que dans la constitution de domaines magnétiques (en physique), ou encore dans le principe d'un **réseau** qui franchit un **seuil** pour consolider des groupes fortement connectés. La synergie, en se focalisant sur la “valeur constructive” de la liaison entre deux unités, sous-entend que l'effet global est un gain pour l'organisation du système. On peut alors parler d'une transition structurelle, où le **paramètre de contrôle** (température, taux η) déclenche une réorganisation.

Certains modèles de champs neuronaux introduisent un poids $\omega_{ij}(t)$ dépendant de la corrélation entre les signaux d'activation, ce qui s'apparente déjà à une **mesure de synergie**. Dans le domaine de la physique, des études soulignent qu'une “mise en phase” (phase-locking) entre deux oscillateurs peut être quantifiée par un indice de co-activation, lequel s'approche également de

l'idée de synergie. Dans ce contexte, des heuristiques simplifiées s'énoncent sous forme de règles locales, si la corrélation entre deux unités dépasse un certain seuil, le lien se renforce ; dans le cas contraire, il s'affaiblit. Bien que rudimentaires, ces règles ouvrent la voie à un cadre plus général, comme celui qu'adoptera le **DSL**. Une fonction $S(i, j)$ exprimant le degré d'interaction "utile" ou "cohérente" entre \mathcal{E}_i et \mathcal{E}_j . La démarche se concrétise en améliorant le simple critère "corrélation positive" vers d'autres formes de co-information.

L'idée de **réseaux dynamiques** où les liens $\omega_{i,j}(t)$ évoluent en continu pour mettre en avant ou en retrait certaines connexions tire parti de cette notion naissante de synergie. Les modèles Hopfield, par exemple, conservent leurs poids constants après un certain apprentissage, alors que la **SOM** (Kohonen) s'appuie sur un voisinage prédéfini. Par contraste, dès qu'on conçoit la synergie comme une **variable** interne évoluant librement, le réseau s'apparente à un **système hautement plastique**. Il abandonne un cadre statique en faveur d'une mise à jour incessante, suivant

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)].$$

Ce schéma préfigure le **Deep Synergy Learning**, où les entités ne sont plus restreintes aux neurones biologiques ou à des spins, mais peuvent inclure des capteurs, des modules symboliques ou d'autres agents. Le concept de **synergie** s'y généralise, passant d'une simple co-activation à une interaction multimodale ou multiniveau. De surcroît, la *structure du réseau* elle-même se trouve remodelée, la synergie permettant de dépasser le canevas fixe des SOM ou des réseaux associatifs classiques.

En définitive, la **convergence** entre neurosciences et physique a forgé la notion d'une synergie locale provoquant un ordre global. Les premières approches ont implémenté ce concept sous forme de règles de corrélation, de mesures de co-information ou de couplages spin, suscitant le projet d'un **réseau auto-organisé** dans lequel l'évolution des liaisons répondrait à un principe de "plus-value" coopérative. Le **DSL** émerge dans cette continuité, proposant un cadre unificateur où la synergie prend des formes variées et se règle en permanence, traduisant la capacité du réseau à **former, rompre, ou stabiliser** ses clusters d'entités en l'absence de supervision.

2.1.4. Précurseurs de la Synergie Multimodale

Les recherches en **auto-organisation** se sont longtemps concentrées sur des données relativement homogènes (spectres sensoriels similaires ou vecteurs numériques). Cependant, dans de nombreux contextes biologiques et robotiques, des **modalités** distinctes (vision, toucher, audition, etc.) interagissent simultanément, incitant les chercheurs à imaginer des **systèmes** capables de fusionner plusieurs types de signaux au sein d'un même réseau adaptatif. Ce sous-chapitre (2.1.4) met en lumière comment, avant même l'essor du **Deep Synergy Learning (DSL)**, des équipes ont tenté de faire coexister différentes sources d'information dans des architectures orientées auto-organisation.

2.1.4.1. Initiatives Cherchant à Fusionner Plusieurs Types de Données (Vision + Tactile, etc.)

Dès les premiers travaux en **robotique sensorielle**, l'idée d'une **multimodalité** s'est imposée comme un enjeu central. En effet, un robot ou un système cognitif peut être pourvu simultanément de caméras (capteurs visuels) et de senseurs tactiles (capteurs de pression, vibration), et souhaiter

découvrir de manière autonome des corrélations ou des “**synergies**” entre ces différents flux. Le besoin de modéliser cette complémentarité, sans supervision explicite, a mené à diverses expérimentations démontrant que la **co-activation** de canaux sensoriels peut se traduire par un **renforcement** des liens entre entités, selon un principe déjà évoqué en section 2.1.3.

D’anciens laboratoires ont notamment testé, dès la fin des années 1970, l’association de **réseaux neuronaux rudimentaires** à des données issues d’**images** et de **capteurs tactiles**, dans le but de reconnaître un objet ou de caractériser une scène. Le fonctionnement de ces prototypes consistait à considérer l’activation simultanée d’une zone visuelle \mathbf{v}_i et d’une zone tactile \mathbf{t}_j . Dans un schéma de type hebbien, la pondération $\omega_{(v_i,t_j)}$ augmentait si ces deux entités coïncidaient lors de la perception d’un même objet. On peut formuler la mise à jour comme

$$\Delta\omega_{(v_i,t_j)} \propto \mathbf{v}_i \mathbf{t}_j,$$

indiquant que la corrélation entre la voie visuelle \mathbf{v}_i et la voie tactile \mathbf{t}_j renforce la liaison les unissant. À mesure que ces expériences étaient réitérées, la liaison correspondante se consolidait jusqu’à se traduire par une coopération plus robuste entre les deux modalités.

Dans le sillage des **Self-Organizing Maps (SOM)** proposées par Teuvo Kohonen, certains chercheurs ont implémenté des “SOM double”, consistant en deux grilles topologiques (l’une pour la vision, l’autre pour le tactile) reliées par un **mécanisme de pont**. Lorsque le même objet était perçu sur les deux modalités, le **voisinage** dans la grille visuelle et la grille tactile incitait chacune de ces deux cartes à se rapprocher, créant une **zone bimodale** où les neurones se trouvaient mutuellement renforcés. Bien que limités, ces essais de **fusion** sans label explicite démontrent déjà le concept qu’une **auto-organisation** conjointe peut conduire à des sous-zones où les informations visuelles et tactiles se *recouvrent*, facilitant une identification plus fiable des objets manipulés.

Parallèlement, les avancées en **neurosciences** révélaient l’existence d’aires associatives “**amodales**” dans le cortex, suggérant que la **coopération** des signaux visuels et tactiles peut survenir naturellement pour former une **perception** intégrée. Les modèles computationnels s’inspiraient de ce constat : à chaque co-occurrence de signaux, on augmentait la force d’un lien (ou d’un cluster) reliant ces deux canaux, ce qui, d’un point de vue mathématique, revient à généraliser la corrélation binaire vers une notion de co-activation dans le temps. De cette manière, l’apprentissage non supervisé exploitait la co-occurrence répétée des mêmes événements sur deux voies sensorielles, sans qu’il soit nécessaire qu’un humain étiquette explicitement leur correspondance.

Cette démarche de **renforcement** adaptatif entre flux hétérogènes préfigure la dynamique mise en œuvre dans le **Deep Synergy Learning (DSL)**. Au lieu de se restreindre à un couplage stationnaire (une SOM visuelle, une SOM tactile, etc.), on peut imaginer un **réseau** où les liens $\omega_{(i,j)}(t)$ entre n’importe quelles entités — visuelles, tactiles, auditives ou symboliques — évoluent en continu, selon une fonction de synergie $S(i,j)$ reflétant leur degré d’activation conjointe. La mise à jour suit un schéma tel que

$$\omega_{(i,j)}(t+1) = \omega_{(i,j)}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{(i,j)}(t)],$$

ce qui rejoint la logique des heuristiques multimodales employées dans ces premiers prototypes. Ces recherches antérieures se révèlent donc être un **préalable direct** à la formulation du DSL, car elles ont posé les bases d'un **apprentissage réellement non supervisé**, tirant parti de la complémentarité entre différents flux sensoriels ou conceptuels, et structurant spontanément un **réseau** où les pondérations fortes dénotent une intense **synergie** entre modalités.

En somme, avant l'avènement du **Deep Synergy Learning**, divers travaux en robotique et en neurosciences computationnelles avaient démontré la viabilité d'une auto-organisation **multimodale**, où le couplage entre modalité visuelle et tactile, notamment, se révélait un cas d'école. L'observation de corrélations temporelles ou fonctionnelles suffisait à susciter un **apprentissage** de type clustering, aboutissant à des représentations plus riches et plus stables que l'approche unimodale. Les idées ainsi expérimentées ont confirmé qu'il n'était pas nécessaire de disposer d'un **superviseur** externe pour associer deux flux distincts : la co-occurrence ou la convergence récurrente s'avérait largement suffisante pour laisser **émerger** la liaison adaptative souhaitée. Ce constat précède et anticipe directement les principes du DSL, qui pousse plus loin la logique de **synergie adaptative** dans un cadre uniifié et potentiellement extensible à plusieurs entités ou modalités simultanées.

2.1.4.2. Influences sur la Mise en Place d'un Cadre Unifié de Synergie Adaptative

Les expériences décrites en section [2.1.4.1](#), qu'il s'agisse de **fusion** vision+tactile ou de cartes “double-SOM” connectées, ont fait émerger une nécessité : disposer d'une **approche générale** pour décrire la manière dont plusieurs **modalités** (ou entités hétérogènes) peuvent coopérer par **auto-organisation**. Les initiatives multimodales pionnières ont souligné l'intérêt d'étendre les règles traditionnelles d'apprentissage sans superviseur vers des situations où les **entités** (capteurs, flux symboliques, représentations vectorielles) ne partagent pas forcément le même format ni le même espace de description. Les réflexions suscitées dans ce domaine ont contribué à l'élaboration ultérieure de paradigmes tels que le **Deep Synergy Learning (DSL)**, dont la logique unifie et **généralise** les notions d'interactions adaptatives entre entités variées.

A. Impulsion depuis la Synergie “Bimodale”

En mettant en relation, par exemple, des données **visuelles** (images) et **tactiles** (capteurs de pression), plusieurs laboratoires ont observé que les **règles classiques** (du type corrélation ou co-activation) se révélaient efficaces même lorsque les flux appartenaient à des espaces différents. Au lieu de se limiter à une **similarité** calculée dans un espace vectoriel identique, ces chercheurs ont proposé des fonctions de “**matching**” ou de “distance inter-modale”, aptes à comparer un patch visuel à un signal tactile. On voyait ainsi qu'il suffisait d'avoir une mesure d'**affinité** cohérente pour permettre un renforcement local, selon un principe semblable à :

$$\Delta\omega_{i,j} \propto \text{similarité}(\mathbf{x}_i, \mathbf{x}_j).$$

Cette conclusion a incité la communauté à considérer des fonctions encore plus flexibles, où chaque couple d'entités définirait sa propre notion de corrélation ou de distance, préparant le chemin vers une **synergie** plus générale.

B. Vers une Extension n-aire

À partir de la fusion de deux modalités, un passage naturel a consisté à s'interroger sur la possibilité de connecter **plusieurs** flux simultanément. Dès lors, on envisageait une **synergie n-aire**, où au-

delà d'un simple produit binaire (vision + tactile), on permettait la coopération de la vision, du toucher, de l'audition, voire d'autres sources (signaux proprioceptifs, informations symboliques, etc.). Cette extension laissait entrevoir des perspectives de **cohésion collective** : un groupe d'entités pouvait se synchroniser autour d'un même événement, même sans label externe, par simple co-occurrence en temps ou en contexte. Les premiers essais, bien qu'expérimentaux, ont montré qu'un réseau pouvait découvrir des regroupements plus sophistiqués dès lors qu'il disposait de **flux variés** et d'une mise à jour adaptative des liens. Sans le cadre théorique unifié, la mise en œuvre restait toutefois ad hoc.

C. Nécessité d'un Cadre Mathématique Unifié

L'extension à plusieurs modalités a révélé les limites des **architectures** de type SOM isolée ou réseau associatif spécialisé. De multiples travaux ont souligné qu'on avait besoin d'un **réseau** davantage "libre" : au lieu de cloisonner chaque flux dans un module (une carte), il serait plus fécond de concevoir un unique **ensemble** d'entités, au sein duquel n'importe quelle paire (ou sous-ensemble) pourrait maintenir une liaison $\omega_{i,j}$ reflétant la "synergie" $S(\mathcal{E}_i, \mathcal{E}_j)$. Cette vision implique un **espace d'interaction** où la **pondération** entre entités évolue au fil du temps, en s'appuyant sur des règles locales de renforcement ou d'affaiblissement. Le concept est proche du **Synergistic Connection Network (SCN)** décrit dans la suite du **DSL** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Ici, \mathcal{E}_i et \mathcal{E}_j peuvent être des entités de natures diverses (sensorielles, symboliques, etc.), et la fonction S se charge d'estimer le degré de "**cohérence**" ou de "**complémentarité**".

D. Évolution Vers la Pondération Adaptative

Dans les premières "SOM doubles" ou architectures robotiques multimodales, les liens entre modalités restaient parfois figés ou suivaient un schéma de **mapping** imposé. Les itérations ultérieures, plus ambitieuses, ont cherché à rendre ces **connexions** elles-mêmes **dynamiques**, comme s'il s'agissait d'une extension de la logique de Hebb ou de la compétition neuronale. On posait donc la question :

"Pourquoi ne pas autoriser la liaison *inter-cartes* à se mettre à jour en temps réel, au même titre que les liaisons *intra-carte* ?"
 Cette interrogation a mené à l'idée d'un **réseau unique**, plutôt que de modules encapsulés, préparant la conception que le **Deep Synergy Learning** développera : un SCN global, évoluant au gré de la synergie perçue entre tous les **types** d'entités.

E. Préparation du Terrain pour le DSL

En repensant l'architecture de l'apprentissage auto-organisé sous un angle plus universel, on arrivait progressivement à un **paradigme** où l'on n'avait plus à distinguer a priori les données en tant que "vision", "tactile" ou "auditive". On leur attribuait plutôt un statut générique d'**entités** dans un réseau, reliées par des liaisons qui s'ajustaient en fonction d'un **score** de co-opération ou de co-information. Ainsi se dessinait déjà l'idée d'une **synergie** non supervisée, en suivant le fil rouge des premiers prototypes multimodaux et en l'étendant à un cadre totalement **auto-adaptatif**.

Les travaux de recherche dans la robotique sensorielle et l'intégration multimodale ont de fait affirmé la **faisabilité** d'une telle approche. Ils ont mis en évidence que la conclusion "ces deux flux

s'avèrent souvent synchronisés, donc ils doivent être reliés et codés conjointement” pouvait être atteinte **sans** recours à un signal d’erreur supervisé. C’est sur cette base qu’ont émergé des pistes ultérieures visant une formalisation plus large de la “synergie” et, en particulier, l’essor de paradigmes comme le **Deep Synergy Learning** qui généralisent ces principes en offrant un langage pour la **coopération** entre entités et la **réorganisation** des liaisons $\omega_{i,j}$.

Dès lors, la volonté de **fusionner** plusieurs types de données a joué le rôle d’un catalyseur, forçant la communauté à sortir d’architectures préétablies pour considérer un **réseau unifié** et malléable. Les règles d’évolution des pondérations sont alors guidées par une fonction $S(i,j)$ – future “fonction de synergie” – et cette logique d’interactions adaptatives reste au cœur du **DSL**.

En conclusion, ces initiatives multimodales ont poussé à systématiser l'**auto-organisation** au-delà de la simple proximité de vecteurs, afin d’engendrer un **cadre** où toute forme de similarité ou de **complémentarité** (visuelle, tactile, auditive, symbolique) puisse s’exprimer sous forme de synergie, ouvrant ainsi la voie aux principes fondateurs du **Deep Synergy Learning**.

2.1.5. Origine du DSL et Ses Premiers Manifestes

Les divers courants présentés dans les sections précédentes (2.1.1 à 2.1.4) — allant des travaux pionniers en auto-organisation jusqu’aux tentatives de fusion multimodale — ont fini par converger vers l’idée de constituer un **cadre unifié** où chaque entité (capteur, neurone, module symbolique, etc.) pourrait interagir avec les autres via un mécanisme de **synergie adaptive**. C’est dans ce contexte qu’est apparue la notion de **Deep Synergy Learning (DSL)**, portée initialement par Mohammed Bouayoun, un ingénieur en informatique qui souhaitait formaliser et pousser plus loin la dynamique de l'**auto-organisation** pour qu’elle englobe l’hétérogénéité des flux (multimodaux, symboliques, etc.) tout en restant fidèle à la tradition d’un apprentissage non supervisé. Le sous-chapitre qui suit (2.1.5) s’intéresse à la genèse historique et à la cristallisation de l’étiquette “DSL” dans la littérature, puis à la formalisation progressive d’un “Synergistic Connection Network (SCN)”.

2.1.5.1. Quand et Comment l’Étiquette “Deep Synergy Learning” Apparaît pour la Première Fois : la Publication dans cet Ouvrage

Au tournant des années 2010, divers laboratoires et chercheurs ont exploré l’idée de **réseaux adaptatifs** capables de gérer simultanément des entités ou des modalités multiples, dans la lignée des inspirations décrites dans les sections précédentes. Toutefois, le terme “**Deep Synergy Learning (DSL)**” tel qu’il est présenté ici n’avait jusqu’alors **jamaïs** fait l’objet d’une publication ou d’une formalisation officielle. C’est dans **ce livre**, pour la première fois, que je, **Mohammed Bouayoun**, rends publiques et détaillées les bases conceptuelles et le fonctionnement d’un **réseau** apte à intégrer la notion de **synergie** au sens large, qu’il s’agisse d’entités perceptuelles, symboliques, ou mixtes.

L’idée de la **synergie** m’accompagne depuis plus de **quarante ans**, influencée par mes travaux initiaux en informatique, par les modèles d'**auto-organisation** neuronale (type Hopfield, Kohonen) et par la **thermodynamique hors équilibre**. Au fil des décennies, ces réflexions se sont enrichies de la logique “hebbienne”, des simulations biologiques et de la fusion multimodale, mais n’ont jamais été diffusées dans un ouvrage ou un article de référence. Aucun document antérieur

ne porte l'étiquette “**Deep Synergy Learning**” au sens où je l'entends aujourd’hui. Ce livre constitue dès lors la **première** présentation formalisée du DSL, énonçant ses principes, ses justifications théoriques et ses applications potentielles.

On trouvera dans les chapitres ultérieurs la description du **Synergistic Connection Network (SCN)**, pièce maîtresse du DSL, ainsi que l’ensemble des règles de mise à jour permettant de faire évoluer les liaisons $\omega_{i,j}$ selon une fonction de **synergie** $S(i,j)$. Si l’on retrouve des traces d’inspirations diverses (réseaux associatifs, cartes topologiques, plasticité synaptique), la synthèse qui en résulte se veut inédite : elle généralise l’**auto-organisation** à un large spectre d’entités, tout en recherchant un ordre émergent via des **pondérations adaptatives**. Cette publication, en franchissant pour la première fois les frontières des laboratoires et des notes internes, aspire donc à faire connaître le DSL et à favoriser son adoption dans un grand éventail de contextes, de la robotique sensorielle à la cognition symbolique.

Ainsi, le **Deep Synergy Learning**, dont je pose ici les fondements, n’avait jusque-là jamais été officiellement exposé ni dans un livre ni dans un article. Ce texte inaugure donc son entrée dans la **littérature** scientifique, concrétisant une vision mûrie de longue date mais demeurée confidentielle jusqu’à présent.

2.1.5.2. Évolution vers la Formalisation d'un “SCN” (Synergistic Connection Network)

La **naissance** du **Deep Synergy Learning (DSL)** s'est accompagnée d'une volonté de conférer à la **synergie adaptative** une **incarnation concrète** au sein d'un réseau à la fois généraliste et dynamique. Cette exigence a conduit Mohammed Bouayoun et ses collaborateurs à façonner le concept de **Synergistic Connection Network (SCN)**, vu comme la structure fondamentale sur laquelle repose le DSL. Les réseaux associatifs traditionnels (Hopfield) et les cartes topologiques (SOM) ont inspiré certains principes, mais ils restaient marqués par des limites : absence de plasticité continue, champ d’application restreint, difficulté à intégrer des entités non neuronales ou des flux multimodaux. Le **SCN**, au contraire, se veut un **réseau** entièrement **évolutif**, où chaque pondération $\omega_{i,j}$ dépend d'une fonction de synergie $S(i,j)$ susceptible d'unifier différents types d’entités (capteurs sensoriels, modules symboliques, etc.). Les paragraphes ci-dessous (2.1.5.2.1 à 2.1.5.2.5) retracent la **progression** qui a permis d’aboutir à ce cadre.

A. Passer d'un Réseau “Fixe” à un Réseau “Entièrement Dynamique”

Les approches antérieures telles que les **SOM** de Kohonen ou les **réseaux** Hopfield présentaient un fonctionnement essentiellement **statique**. Une SOM impose une grille et un mode de voisinage prédéterminé, tandis que les poids d'un réseau Hopfield sont généralement calculés une fois pour toutes sur la base de motifs ciblés. Ces méthodes ne permettent pas de gérer en continu l’arrivée de nouvelles entités ou de mettre à jour les liaisons dans des environnements en transformation permanente. Mohammed Bouayoun, inspiré par la **plasticité neuronale**, a suggéré que le réseau pourrait évoluer par itérations successives, les pondérations $\omega_{i,j}(t)$ se renforçant ou s’atténuant selon un **score de synergie** $S(i,j)$. L'équation générique (développée plus tard dans ce livre) concrétise cette idée en intégrant un **terme de renforcement** (quand la synergie est élevée) et un

terme de décroissance (pour éviter l’emballlement), assurant un équilibre entre stabilité et adaptabilité.

B. Notion de “Synergistic Connection Network”

Sous l’impulsion de ces réflexions, est né le concept de **Synergistic Connection Network (SCN)**, un graphe (voire un hypergraphe) dont les nœuds correspondent à des **entités** $\mathcal{E}_1, \mathcal{E}_2, \dots$, alors que les arêtes pondérées $\omega_{i,j}$ incarnent la liaison entre \mathcal{E}_i et \mathcal{E}_j . La distinction cruciale, par rapport aux architectures classiques, réside dans le fait que ces liaisons ne sont pas figées : elles varient sous l’effet d’une **fonction** $S(\mathcal{E}_i, \mathcal{E}_j)$ évaluant la synergie. À chaque étape (itération, nouvelle donnée, flux continu), les pondérations se recalculent pour refléter le degré de coopération ou de similitude actuelle entre les entités. Ce **réseau vivant** autorise la découverte de structures auto-organisées dans des champs aussi divers que la **fusion sensorielle**, la **cohérence symbolique** ou la **mise en correspondance** de flux d’origine variée.

C. Dynamique de la Fonction $\omega_{i,j}(t)$ et Synergie $S(i,j)$

Le SCN s’appuie sur un mécanisme de **plasticité locale** où la pondération $\omega_{i,j}(t)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j obéit, à chaque itération, à une équation de type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Ici, η est un **taux d’apprentissage**, tandis que τ joue le rôle d’un **coefficent** de régulation, souvent nommé “terme de décroissance”. La **fonction** $S(i,j)$ peut être une simple corrélation ou quelque chose de plus élaboré : distance inversée, similarité cosinus, information mutuelle, etc. Cette souplesse correspond à l’ambition du SCN de **fusionner** différents types de données, en adoptant une métrique ou une fonction de synergie appropriée pour chaque couple (ou sous-groupe) d’entités. L’effet global est un **réseau** qui se reconfigure sans cesse, renforçant les liaisons cohérentes et affaiblissant les autres, selon les principes hebbiens et les idéaux de la **physique hors équilibre** (cf. chapitres 2.1.3.1 et 2.1.3.2).

D. Diffusion dans des Publications et Ateliers

Au fil des travaux, ce schéma évolutif a pris forme dans des **rapports** et **communications** spécialisées, notamment dans le cadre de congrès sur l’**apprentissage non supervisé**. Les premières applications pratiques concernaient la fusion multisensorielle ou la classification évolutive, montrant que le SCN sait gérer la venue de nouvelles données, adapter les pondérations au fil de l’expérience et former des **clusters** persistants. Parmi les questions soulevées figurent la **scalabilité** (comment se comporte le SCN avec un grand nombre d’entités ?) et la **stabilité** (comment éviter que les pondérations ne divergent ou ne s’étiolent toutes ?). Ces interrogations ont orienté la mise au point de mécanismes de **parsimonie** (seuils, inertie minimale), d’**inhibition** compétitive et de **taux d’apprentissage** dynamique, dont les détails seront explicités dans les prochains chapitres.

E. Positionnement pour la Suite de l’Ouvrage

Ainsi, la création du **Synergistic Connection Network** marque une étape décisive dans la concrétisation du **Deep Synergy Learning**. Pour la première fois, un **réseau** est conçu pour embrasser la diversité des entités (capteurs de nature variée, modules symboliques, etc.), tout en veillant à ce que les liaisons $\omega_{i,j}$ restent **adaptatives** et que la **synergie** $S(i,j)$ puisse se décliner sous des formes diverses. Les chapitres suivants de cet ouvrage (chapitres 3, 4, 5) approfondiront la formulation analytique, les algorithmes de mise à jour, et les fondements mathématiques de la dynamique $\omega_{i,j}(t)$. On y détaillera notamment la notion de **cluster** en tant qu’attracteur emergent, la gestion de la **parsimonie** afin de maintenir la lisibilité et la scalabilité, ainsi que les méthodes pour **évaluer** la pertinence des synergies dans des scénarios multimodaux réels.

En définitive, la formalisation du **SCN** illustre la transition d’une simple **idée** (renforcer les liens lorsqu’on détecte une forte co-occurrence) à un **cadre** complet d’**auto-organisation** appliquée aux entités hétérogènes. Le **DSL** s’appuie désormais sur cette armature conceptuelle pour proposer un système de pondérations locales, à la fois général et flexible, ouvrant la voie à une **intelligence** non supervisée, capable de gérer la diversité des données et de s’auto-structurer en conséquence.

2.1.5.3. Lien Direct avec la Suite du Livre : Axes Théoriques Restés Inexplorés

La genèse du **Deep Synergy Learning (DSL)** et la formalisation du **Synergistic Connection Network (SCN)**, évoquées dans les sections précédentes, laissent entrevoir un champ encore **vaste** de problématiques que la littérature n’a pas couvertes de manière exhaustive. Le présent ouvrage se propose précisément de combler plusieurs de ces **lacunes** et de consolider les bases théoriques et pratiques du **DSL**. La discussion suivante met en évidence les principaux **axes** de recherche inachevés, tout en exposant la manière dont les chapitres ultérieurs (3 à 15) s’emploient à les approfondir ou à proposer des réponses partielles.

A. Extension de la Fonction de Synergie à des Contextes Plus Complexes

La plupart des modèles initiaux se limitaient à des **fonctions** de similarité relativement simples, comme la distance euclidienne ou la corrélation binaire, pour calculer la synergie entre deux entités \mathcal{E}_i et \mathcal{E}_j . Or, certains domaines exigent une **cohérence** ou une **interaction** plus élaborée, qui peut s’appuyer sur des notions de co-information n-aire ou sur des métriques conditionnelles sophistiquées. D’un point de vue mathématique, on se heurte alors à la nécessité de définir

$$S(\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n)$$

pour un sous-groupe de taille $n > 2$, ce qui amène à de nouveaux défis de calcul et de stockage. Le **Chapitre 3** aborde la représentation des entités et jette les bases de la synergie, tandis que le **Chapitre 12** traite spécifiquement de la **synergie n-aire**, en exposant des formules permettant de saisir les interactions multivariées.

Par ailleurs, le **DSL** se révèle apte à gérer des flux en temps continu. Cela suppose que la fonction $S(i,j)$ demeure **adaptative** au fil du temps, afin de prendre en compte l’évolution possible des entités \mathcal{E}_i et \mathcal{E}_j . Les **Chapitres 9 et 11**, consacrés respectivement aux scénarios temps réel et à la

robustesse/sécurité, examinent la façon dont le **SCN** peut conserver la mémoire des synergies passées, tout en s'ajustant à l'arrivée de nouvelles données ou d'entités inédites.

B. Contrôle de la Stabilité et Algorithmes d'Optimisation

Il est légitime de s'interroger sur la **stabilité** de la règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta[S(i,j) - \tau\omega_{i,j}(t)].$$

D'un point de vue analytique, il s'agit de savoir si ces équations peuvent engendrer des **oscillations**, ou même des comportements chaotiques, lorsque le réseau s'agrandit ou que la matrice $\{\omega_{i,j}\}$ atteint une dimension élevée. Les **Chapitres 4, 5, 7 et 10** explorent ces problématiques. Les solutions proposées incluent, entre autres, l'ajout d'un mécanisme d'**inhibition compétitive** (destiné à empêcher l'explosion d'un trop grand nombre de liaisons fortes), l'introduction de seuils minimaux pour couper les connexions trop faibles ou encore l'élaboration de stratégies de **parsimonie** structurelle.

La **scalabilité** en présence de milliers, voire de millions, d'entités devient un enjeu majeur. Les approches naïves peuvent rapidement se heurter à une complexité en $O(n^2)$. Le **Chapitre 7** détaille des algorithmes plus performants, tels que l'**échantillonnage** ou l'**approximation** de voisinage, et puise dans des analogies avec la **physique statistique** (par exemple le recuit simulé) afin de maintenir la convergence sans tout calculer explicitement. Les utilisateurs du **DSL** se voient ainsi offrir un éventail de méthodes pour contrôler la densité du réseau et éviter que les liaisons $\omega_{i,j}$ ne prolifèrent ou ne s'effondrent de manière incontrôlée.

C. Ouverture vers la Multimodalité Avancée et le Raisonnement Symbolique

Un autre front de recherche concerne la **multimodalité** lorsqu'elle est poussée à l'extrême : il ne s'agit plus seulement de relier deux flux (vision + tactile), mais d'intégrer simultanément des sources diverses (images, sons, textes, logique symbolique). Dans cette optique, on souhaite s'assurer que la **fonction de synergie** $S(\cdot, \cdot)$ ou $S(\cdot, \cdot, \cdot)$ soit assez flexible pour gérer des écosystèmes d'entités hétérogènes, tout en garantissant que le SCN ne se fragmente pas en une myriade de **clusters** trop spécialisés et déconnectés. Le **Chapitre 8** se penche spécifiquement sur ces scénarios multimodaux et propose des pistes pour favoriser la **transversalité** entre flux distincts. Les **Chapitres 14 et 15**, davantage orientés applications, reviennent sur divers cas concrets (fusions de flux audio-visuel-textuel, organisation de connaissances dans un cadre hybride).

La présence d'**entités symboliques** (règles logiques, axiomes) étend encore le périmètre d'action du DSL, et il s'agit alors de caractériser la **synergie** entre une entité perceptuelle (un capteur) et une entité abstraite (une proposition symbolique). Les **Chapitres 5, 13 et 14** abordent cette problématique en discutant des approches dites **neuro-symboliques**, où un réseau gère simultanément de la "matière" sub-symbolique (poids, neurones virtuels) et des "objets" symboliques. La recherche sur la cohabitation de ces deux mondes reste néanmoins incomplète, et le livre ne prétend pas en livrer une théorie achevée, mais propose des esquisses d'**implémentation** et des modèles hybrides encourageant un raisonnement distribué.

Conclusion sur les Axes Théoriques Restés Inexplorés

Le **Deep Synergy Learning**, formalement présenté dans ce livre pour la première fois (cf. section [2.1.5.1](#)), débouche sur un **cadre** suffisamment large pour accueillir des entités variées et autoriser

une dynamique de pondérations adaptatives. Les **lacunes** ou **chantiers** toujours ouverts concernent d'abord l'élaboration de **fonctions de synergie** toujours plus polyvalentes (synergie n-aire, co-information conditionnelle). Ils portent aussi sur l'analyse **mathématique** de la stabilité du **SCN** (en particulier dans les grands réseaux), et enfin sur la **fusion** multimodale avancée associant données sensorielles, logiques symboliques et autres ressources. Les chapitres à venir s'emploient à clarifier chacun de ces thèmes :

- Les **fondements mathématiques** (Chapitre 3 et Chapitre 4)
- Les **règles de mise à jour** et la **structure interne** du SCN (Chapitre 5, Chapitre 7)
- Les scénarios d'**évolution en temps réel** (Chapitre 9) et de **multimodalité** (Chapitre 8, Chapitre 14)
- Les **cas pratiques** et les extensions potentielles (Chapitre 15)

Ainsi, le livre fournit un socle formel pour la **synergie adaptative** entre entités. La variété des pistes encore ouvertes souligne combien le **DSL** n'est pas un aboutissement final, mais plutôt un **cadre évolutif** lui-même, évoluant au rythme des investigations en intelligence artificielle, en neurosciences computationnelles et en physique des systèmes complexes.

Conclusion sur les Axes Théoriques Restés Inexplorés

En clair, si la **genèse** du Deep Synergy Learning et la mise en place du SCN marquent une avancée notable dans l'histoire de l'auto-organisation, plusieurs **chantiers** théoriques et pratiques restent à développer :

- **Formalisation** complète d'une synergie adaptée à la cohabitation de multiples types d'entités (sub-symboliques, symboliques) ;
- **Étude** approfondie de la **stabilité** et des attracteurs dans un réseau dont les pondérations évoluent en continu, notamment en présence de synergie n-aire ;
- **Optimisation** et **scalabilité**, cruciales pour supporter un grand nombre d'entités dans le SCN ;
- **Transitions** vers la multimodalité avancée et la prise en compte d'un raisonnement symbolique plus élaboré.

Les chapitres suivants du livre aborderont bon nombre de ces **challenges**, proposant des embryons de solution ou montrant des applications concrètes. Le lecteur verra comment ces problématiques s'imbriquent dans la dynamique du SCN, et comment l'héritage historique présenté ici (depuis les réseaux associatifs, les cartes topologiques, jusqu'aux modèles multi-sensoriels et la physique statistique) trouve sa continuité dans un cadre unifié **Deep Synergy Learning**.

2.2. Principes Mathématiques de Base

Le **Deep Synergy Learning (DSL)** s'appuie sur un ensemble de **principes mathématiques** permettant de définir formellement les entités (leurs caractéristiques) et les règles d'**auto-organisation** du réseau. Contrairement à certains paradigmes neuronaux classiques (où l'on manipule seulement des poids fixes ou un gradient d'erreur), le DSL met en avant la **fonction de synergie** S et la dynamique de mise à jour des pondérations $\omega_{i,j}$. Ces concepts reposent sur des bases analytiques qui font intervenir la notion d'**espace de représentation**, de **similarité** ou de **co-information**, et de **procédés d'évolution** (discrets ou continus). Le présent chapitre (2.2) en présente les **fondements**, en plusieurs sous-sections :

- **2.2.1.** Définition des Entités et de la Fonction de Synergie
- **2.2.2.** Mise à Jour des Pondérations : Formule Générale
- **2.2.3.** Règles de Parsimonie et Seuil de Connexion
- **2.2.4.** Notions d'États Internes et Auto-Organisation
- **2.2.5.** Exemples Illustratifs

Ce sont les **piliers** mathématiques sur lesquels s'édifie le **Synergistic Connection Network (SCN)** et, plus généralement, l'ensemble de la logique d'**auto-organisation** du DSL.

2.2.1. Définition des Entités et de la Fonction de Synergie

La première étape pour tout modèle DSL consiste à **identifier** les **entités** $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ qui composeront le réseau et, surtout, à spécifier la **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Si la littérature sur l'apprentissage non supervisé (SOM, Hopfield, etc.) imposait en général un format vectoriel unique, le DSL se distingue par sa capacité à gérer des **représentations hétérogènes** (capteurs, flux symboliques, extraits d'un transformeur, etc.). Les sections suivantes (2.2.1.1 à 2.2.1.3) détaillent cette notion de **définition d'entités** et la façon dont on conçoit la **fonction S** .

2.2.1.1. Rappel : Une Entité \mathcal{E}_i (Vecteur de Caractéristiques, Représentation Symbolique, etc.)

Le **Deep Synergy Learning (DSL)** s'articule autour de la notion d'**entité** \mathcal{E}_i et de la **synergie** définie sur l'ensemble de ces entités. Dans la pratique, on considère que chaque \mathcal{E}_i fait partie d'un **ensemble** \mathcal{X} qui peut prendre des formes variées : espace vectoriel, structure symbolique, ou représentation probabiliste. Il est indispensable de disposer d'une **mesure** ou d'une **similarité** pour comparer deux entités, car c'est cette comparaison qui sous-tend la **pondération** $\omega_{i,j}$ et la **dynamique** d'auto-organisation du DSL. Les développements ci-après soulignent les différents cas de figure envisageables (sections vectorielles, symboliques, probabilistes) et introduisent un formalisme plus général permettant de prendre en compte des **espaces hétérogènes**.

A. Entités dans un Espace Vectoriel \mathbb{R}^d

Un premier cas, souvent le plus courant, suppose que chaque entité \mathcal{E}_i est représentée par un **vecteur** réel de dimension d . On écrit alors

$$\mathcal{E}_i \in \mathbb{R}^d,$$

et on dispose de **distances** ou de **similarités** naturelles : distance euclidienne, distance Minkowski, similarité cosinus, etc. Les propriétés de l'analyse vectorielle facilitent la définition de mesures telles que

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\| \quad \text{ou} \quad \text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

Cette configuration se rencontre fréquemment dans la pratique (traitement d'images, de séries temporelles, etc.), et fournit une base conceptuelle aisée pour **construire** la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ par inversion de distance ou par évaluation de similarité, prolongeant l'idée de la corrélation ou de la co-occurrence.

B. Structures Symboliques ou Espaces Discrets

Une **entité** \mathcal{E}_i peut aussi correspondre à un **concept** symbolique, un **graphe** local, ou tout autre objet appartenant à un **espace** discret où aucune structure vectorielle n'est immédiatement disponible. Les ensembles de cette nature exigent la définition d'une **mesure** ou d'une **similarité** plus adaptée, prenant la forme d'une distance combinatoire, d'une ressemblance sémantique, ou d'une co-information symbolique. Formellement, on peut écrire

$$\mathcal{E}_i \in \mathcal{X}_{symbolique},$$

et introduire une fonction $\text{sim}(\mathcal{E}_i, \mathcal{E}_j)$ prenant des valeurs dans $[0,1]$ ou \mathbb{R}^+ . Ce cadre est crucial dans le DSL, qui autorise la **co-existence** d'entités hétérogènes et nécessite donc que la **synergie** s'applique aussi à des représentations non purement numériques. Par exemple, on peut concevoir une mesure de similarité symbolique $\sigma(\mathcal{E}_i, \mathcal{E}_j)$ qui vaut 1 si deux symboles sont identiques, ou prend une valeur intermédiaire si l'on recourt à une **ontologie** reliant les concepts.

C. Espace de Probabilité ou Représentation Probabiliste

Dans certains travaux, l'entité \mathcal{E}_i est un **modèle probabiliste**, par exemple une **distribution** $p_i(x)$. On peut alors définir la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ par une divergence ou une similarité adaptée, telles que la **KL divergence** ou la **distance Wasserstein**. Ainsi, on écrit

$$\mathcal{E}_i \in \mathcal{P}(\mathcal{X}),$$

où $\mathcal{P}(\mathcal{X})$ désigne l'ensemble des distributions sur un espace \mathcal{X} . La fonction S peut alors prendre des formes comme

$$S(\mathcal{E}_i, \mathcal{E}_j) = \exp(-d_{KL}(p_i, p_j)) \quad \text{ou} \quad S(\mathcal{E}_i, \mathcal{E}_j) = \text{Wasserstein}(p_i, p_j),$$

autorisant l'auto-organisation de **modèles** statistiques sans référence explicite à des labels supervisés.

D. Forme Générale et Hypothèses Topologiques

On adopte volontiers la notation

$$\mathcal{E}_i \in \mathcal{X},$$

où \mathcal{X} est un **espace topologique** (ou mesurable), muni d'une **distance** ou d'une **similarité**. Le **triplet** (\mathcal{X}, d, μ) ou (\mathcal{X}, S, μ) permet de considérer soit des distances classiques, soit des "fonctions de synergie" plus directes. Le **DSL** n'exige pas nécessairement que les entités se situent dans un unique espace vectoriel : différentes **sous-familles** d'entités peuvent coexister, chacune dotée de sa propre mesure.

E. Propriétés Clés : Séparabilité, Embedding Vectoriel, Extension Symbolique

Certaines **propriétés** mathématiques facilitent l'étude de la synergie. Par exemple, une **distance** d sur \mathcal{X} procure un espace métrique complet, ce qui autorise une **analyse** de la convergence de la fonction de synergie si elle se base sur d . Sous des conditions favorables (e.g. la structure de \mathcal{X} est hilbertienne), il est possible d'**implanter** (\mathcal{X}, d) dans un espace \mathbb{R}^d ou un **espace de dimension infinie** (via un **noyau**), ouvrant la voie à des similarités usuelles (produit scalaire, gaussien, cosinus). Lorsque \mathcal{X} est un ensemble **discret** (fin ou dénombrable), on peut définir une **similarité** $\sigma(\mathcal{E}_i, \mathcal{E}_j)$ dans $[0,1]$ basée sur une codification combinatoire ou sémantique.

F. Gestion de l'Hétérogénéité des Entités

Le **DSL** poursuit un objectif d'**intégration** de différents types d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$, qui peuvent appartenir à divers espaces $\mathcal{X}_1, \mathcal{X}_2, \dots, \mathcal{X}_k$. Pour comparer $\mathcal{E}_i \in \mathcal{X}_p$ et $\mathcal{E}_j \in \mathcal{X}_q$, il est indispensable de **prolonger** les mesures ou d'introduire un **mécanisme de passerelle** (embedding commun, correspondance symbolique...). C'est dans cette optique qu'est formulé le **théorème** sur la consistance de la synergie inter-espaces, garantissant l'existence d'une fonction globale Σ unifiant les similarités σ_p dans un unique référentiel. Les détails topologiques (réunion disjointe, mise en place d'une distance globale δ séparant strictement les blocs) assurent la **continuité** de la fonction Σ .

Exemple de Théorème (Consistance de la Synergie Inter-Espaces)

Enoncé (informel).

Soient $\{\mathcal{X}_m\}_{m=1}^k$ des espaces (topologiques ou mesurables), chacun muni d'une fonction de similarité σ_m . Sous hypothèses de continuité uniforme et de séparation stricte entre blocs, il existe une **extension** Σ définie sur la réunion disjointe $\mathcal{U} = \bigsqcup_{m=1}^k (\mathcal{X}_m \times \{m\})$, telle que

$$\Sigma|_{\mathcal{X}_m \times \mathcal{X}_m} = \sigma_m \quad \text{et} \quad \Sigma((x, p), (y, q)) = c_{p,q} \quad (p \neq q).$$

On obtient ainsi une fonction $\Sigma: \mathcal{U} \times \mathcal{U} \rightarrow [0,1]$ (ou \mathbb{R}^+), **uniformément continue** sur la distance δ définie par bloc, offrant un cadre d'unification pour l'**auto-organisation** dans un seul **Synergistic Connection Network (SCN)**.

La démonstration s'appuie sur la **réunion disjointe** et la définition d'une distance globale δ qui maintient une séparation stricte (Δ_0) entre blocs \mathcal{X}_p et \mathcal{X}_q pour $p \neq q$. La construction assure que Σ est **intra-bloc** égale à σ_m , et **inter-bloc** égale à une constante (souvent zéro). On en déduit que Σ est **uniformément continue**, ce qui garantit la possibilité de réaliser une **analyse** de stabilité ou de convergence si l'on souhaite faire évoluer un réseau global.

Conclusion (2.2.1.1)

Au terme de cette analyse, on retient que, dans le **Deep Synergy Learning**, chaque **entité** \mathcal{E}_i peut être un simple **vecteur** (cas fréquent en \mathbb{R}^d), une **structure symbolique** (concept discret, graphe), ou même un **modèle probabiliste**. L'important est de disposer d'une **fonction** (distance, similarité, co-occurrence) permettant de quantifier la “**proximité**” ou la “**coopération**” entre entités, sans quoi la notion de **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ reste vide. Dans les chapitres suivants, on verra **comment** décliner cette synergie (section 2.2.1.2, centrée sur distance, corrélation, information mutuelle) et **comment** l'utiliser pour piloter la **mise à jour** des pondérations $\omega_{i,j}$ (section 2.2.2). Ainsi, le **DSL** se veut un cadre assez large pour **unifier** la comparaison d'entités de natures diverses et lancer, sur cette base, un **SCN** (Synergistic Connection Network) unique, où les liens évoluent selon la valeur de la synergie détectée.

2.2.1.2. Qu'est-ce que $S(\mathcal{E}_i, \mathcal{E}_j)$? Exemples (distance, similarité, co-information)

Pour décrire la **synergie** reliant deux entités \mathcal{E}_i et \mathcal{E}_j dans le **Deep Synergy Learning (DSL)**, on introduit une **fonction** $S(\mathcal{E}_i, \mathcal{E}_j)$ qui, pour chaque couple d'entités, mesure leur **coopération** ou leur **affinité**. On formalise souvent ce concept par

$$S: \quad \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+,$$

où \mathcal{X} représente l'espace (ou la **réunion** d'espaces) hébergeant les entités $\mathcal{E}_i, \mathcal{E}_j$. Selon la **nature** de \mathcal{X} et les **objectifs** (clustering, fusion multimodale, etc.), la fonction S peut être fondée sur différents concepts (distance, similarité, information mutuelle). Les développements ci-après (2.2.1.2.1 à 2.2.1.2.3) exemplifient trois grandes **familles** de mesures couramment mises en avant dans le DSL.

A. Distance

Lorsque les entités \mathcal{E}_i et \mathcal{E}_j appartiennent à un **espace métrique** (\mathcal{X}, d) , on peut définir la synergie comme une **fonction** décroissante de la distance :

$$S(\mathcal{E}_i, \mathcal{E}_j) = f(d(\mathcal{E}_i, \mathcal{E}_j)),$$

avec, par exemple, $f(x) = 1/(1+x)$ ou $f(x) = \exp(-\alpha x)$. L'intuition repose sur l'idée qu'une faible **distance** implique une **coopération** plus marquée. Dans un cadre vectoriel \mathbb{R}^d , si l'on prend la distance euclidienne

$$d(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|,$$

on peut fixer

$$S(\mathcal{E}_i, \mathcal{E}_j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|),$$

de sorte que deux entités **proches** dans \mathbb{R}^d obtiennent un score S élevé, et deux entités **éloignées** un score proche de zéro. Sur le plan des **propriétés**, la distance garantit la **symétrie** $S(\mathcal{E}_i, \mathcal{E}_j) = S(\mathcal{E}_j, \mathcal{E}_i)$ si le choix de la fonction f respecte cette invariance.

B. Similarité ou Corrélation

Une autre façon de concevoir la **synergie** privilégie l'idée de **similarité** ou de **corrélation** statistique. Par exemple, si \mathcal{E}_i et \mathcal{E}_j sont deux vecteurs $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$, on peut recourir à la **similarité cosinus** :

$$S(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

Dans ce cas, plus l'**angle** entre les vecteurs est faible, plus la valeur de S est grande. Pour des variables **aléatoires** X, Y , on peut choisir le **coefficient de corrélation** $\rho(X, Y)$, par exemple la **corrélation de Pearson**, ou toute autre mesure similaire. D'un point de vue algorithmique, le **DSL** exploitera alors ces scores comme un indicateur de synergie : lorsque deux entités se révèlent linéairement (ou faiblement) interdépendantes, leur lien $\omega_{i,j}$ se renforce modérément ; lorsqu'elles affichent une corrélation élevée, le **renforcement** sera marqué. Cette logique, conforme aux paradigmes associatifs (Hopfield) ou hebbiens, consolide l'idée que la **co-activation** façonne le **réseau** en intensifiant les liaisons utiles.

C. Information Mutuelle ou Co-Information

Dans des situations où les entités $\mathcal{E}_i, \mathcal{E}_j$ représentent des **variables aléatoires** (ou des distributions de probabilité), une approche consiste à mesurer la **synergie** via l'**information mutuelle** :

$$S(\mathcal{E}_i, \mathcal{E}_j) = I(\mathcal{E}_i; \mathcal{E}_j) = \int p(x, y) \log[p(x, y)/p(x)p(y)] dx dy.$$

Cette grandeur quantifie la **dépendance** entre deux variables : plus l'une réduit l'incertitude sur l'autre, plus leur information mutuelle est élevée. Dans un cadre plus général, on parle de **co-information** $I(\mathcal{E}_1; \mathcal{E}_2; \dots; \mathcal{E}_k)$ lorsque plusieurs variables interagissent collectivement (voir Chapitre 12). Sur le plan **théorique**, l'information mutuelle a l'avantage de capturer des dépendances **non linéaires**, dépassant la simple corrélation. Si $S(\mathcal{E}_i, \mathcal{E}_j)$ est défini comme $I(\mathcal{E}_i, \mathcal{E}_j)$, alors la valeur s'annule en cas d'indépendance et s'accroît avec la force de la dépendance. Dans le **DSL**, ce schéma se prête bien à un **renforcement** de $\omega_{i,j}$ lorsque les variables s'avèrent interdépendantes.

Toutefois, la mise en pratique de l'information mutuelle peut exiger des **estimations** ou des **approximations** si l'on ne connaît pas explicitement les distributions $p(x)$. Des estimateurs basés

sur des techniques de **binning**, de **k plus proches voisins** ou des méthodes paramétriques (ex. gaussiennes) sont souvent employés. Les performances et la fiabilité de ces estimateurs conditionnent alors la précision de la fonction S.

Conclusion sur le Choix de la Fonction de Synergie

La décision quant à la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ oriente la **dynamique** du **DSL** : seules les paires $\mathcal{E}_i, \mathcal{E}_j$ présentant un score élevé verront leur lien $\omega_{i,j}$ se renforcer durablement. Ce principe de “coopération” peut s’appuyer sur :

- Une **distance** (ou son inverse),
- Une **similarité** (cosinus, corrélation linéaire, etc.),
- Une **information** ou **co-information** (mutuelle),
- Ou tout autre **critère** de dépendance ou d’utilité mutuelle.

Le **Deep Synergy Learning** n’impose pas un choix rigide, mais définit un **cadre** — le **Synergistic Connection Network** (SCN) — où la mise à jour $\Delta\omega_{i,j}$ se greffe sur la valeur de $S(\mathcal{E}_i, \mathcal{E}_j)$. Les chapitres suivants approfondiront la question du **calcul effectif** de cette synergie, notamment lorsque l’on gère des entités **multimodales** (Chapitre 8) ou de **plusieurs** variables (Chapitre 12). L’idée maîtresse demeure : toute fonction de **coopération** ou d’**interaction** entre deux entités, pourvue d’une signification appropriée au domaine d’application, peut servir de **base** au mécanisme adaptatif du DSL.

2.2.1.3. Relation entre cette Fonction et la Mesure d’Utilité Mutuelle ou d’Interaction

Après l’exposé des divers exemples (distance, similarité, information mutuelle) pour la **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$, il importe d’analyser la **signification** profonde de ce score dans le **Deep Synergy Learning (DSL)**. Cette section [2.2.1.3] aborde la connexion existante entre S et des concepts plus généraux d’**utilité mutuelle** ou d’**interaction** bénéfique, en formalisant l’idée que la **synergie** reflète une forme de “plus-value” commune aux entités \mathcal{E}_i et \mathcal{E}_j .

A. Notion de Bénéfice Mutuel et d’Utilité

Considérons un **réseau** adaptatif où l’**interaction** $(\mathcal{E}_i, \mathcal{E}_j)$ procure un **gain** à la structure globale, par exemple la reconnaissance partagée d’un même cluster ou la contribution à une tâche identique. On peut **formaliser** cette idée au moyen d’une “utilité” $Util(\mathcal{E}_i, \mathcal{E}_j)$, envisagée comme une fonction $Util: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}^+$. Il s’agit de quantifier dans quelle mesure la **relation** entre i et j se révèle profitable. Dans le DSL, il arrive qu’on assimile

$$S(\mathcal{E}_i, \mathcal{E}_j) = Util(\mathcal{E}_i, \mathcal{E}_j)$$

ou qu’on emploie une transformation monotone pour s’assurer que $S \in [0,1]$. L’essentiel est que plus l’**utilité** perçue est grande, plus la **synergie** S l’est aussi. Dans un cadre **non supervisé**, la

reconnaissance d'une dépendance mutuelle forte (distance faible, corrélation élevée, information partagée importante) implique un **bénéfice** pour le réseau, lequel renforce alors le lien $\omega_{i,j}$.

Exemple : Lorsque la **synergie** consiste en l'**information mutuelle** $I(X; Y)$, la “diminution d’incertitude” qu’apporte X sur Y s’interprète comme un **gain** (une “utilité”), d’où le renforcement de $\omega_{i,j}$. Si la synergie repose sur une **similarité** positive (ex. $\exp(-\alpha d(\mathbf{x}_i, \mathbf{x}_j))$), l'**utilité** réside dans la facilité ou le bénéfice de rassembler deux entités $\mathbf{x}_i, \mathbf{x}_j$.

B. Lien avec la Coopération et l’Émergence de Clusters

Dans un **réseau** où les liaisons $\omega_{i,j}$ se voient renforcées lorsque $S(\mathcal{E}_i, \mathcal{E}_j)$ est élevée, on retrouve la **logique** de la théorie de la coopération : les unités qui coopèrent s’avèrent plus **connectées** et, par ricochet, le **système** global y gagne. Sur le plan algébrique, des paires (i, j) pour lesquelles $S(i, j)$ reste notable finissent par tisser un **sous-réseau** (ou cluster) de pondérations fortes $\omega_{i,j}$.

Cette vision recoupe la notion d'**utilité** : regrouper des entités “compatibles” ou “enrichissantes” apparaît comme une stratégie d’amélioration globale (stratégie de “somme positive”). On peut d’ailleurs ajouter une perspective énergétique : considérer la somme (ou l’opposé)

$$\sum_{i < j} \omega_{i,j} S(\mathcal{E}_i, \mathcal{E}_j)$$

en tant que **critère** à maximiser, ce qui amène le réseau à rechercher la configuration $\{\omega_{i,j}\}$ la plus profitable pour l’ensemble. Les chapitres 4 et 7 présenteront des **théorèmes** expliquant comment, sous certaines hypothèses (symétrie de la synergie, bornes sur η, τ), la dynamique converge vers un **minimum local** stable, c.-à-d. une **partition** en clusters où les liens internes sont forts.

C. Interaction au-delà de la Simple Corrélation

Dans certains cas, l'**interaction** entre deux entités outrepasse la corrélation linéaire (Pearson). Un **lien** non linéaire ou conditionnel peut se révéler plus important. De même, l'**information mutuelle** ou la **co-information** peut détecter des dépendances qui échappent à un modèle linéaire. Si \mathcal{E}_i et \mathcal{E}_j entretiennent un rapport non trivial (exemple : $Y = X^2 + \text{bruit}$), la corrélation peut être faible, tandis que l’information mutuelle pointe une dépendance claire. Dans un cadre plus vaste encore, si $\{\mathcal{E}_1, \dots, \mathcal{E}_k\}$ partagent une interaction collective, on peut imaginer une **synergie n-aire** (voir Chapitre 12).

Le **DSL** entend **s’adapter** à de telles configurations. La fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ doit donc pouvoir accepter une grande variété de mesures d’“utilité” ou de co-dépendance. Cette **flexibilité** fait de ce paradigme un cadre général, capable d’intégrer tantôt une complémentarité sémantique, tantôt une co-occurrence sensorielle, etc.

D. Théorème (Existence d’un Score d’Interaction)

Considérons un ensemble d’entités $\{\mathcal{E}_i\}$. Supposons qu’on possède pour chaque paire (i, j) une mesure d'**interaction** $\text{Int}(\mathcal{E}_i, \mathcal{E}_j) \geq 0$ reflétant la co-occurrence ou la dépendance. Il est possible de construire une fonction de **synergie** S grâce à une transformation monotone ϕ . Concrètement, on définit

$$S(\mathcal{E}_i, \mathcal{E}_j) = \phi(\text{Int}(\mathcal{E}_i, \mathcal{E}_j)),$$

où ϕ est une application $[0, +\infty) \rightarrow [0,1]$ (ou \mathbb{R}^+), strictement croissante et vérifiant $\phi(0) = 0$. Ainsi, chaque valeur d'interaction $\text{Int}(i, j)$ se convertit en un **score** $S(i, j)$ strictement **non négatif**, aligné sur l'**ordre** (plus Int augmente, plus S augmente). Cette construction prouve qu'il existe un moyen d'englober toute forme de mesure d'utilité mutuelle dans la **logique** du DSL, tant que la mesure est monotone et non négative.

Conclusion

La **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ constitue, dans le **DSL**, un **score** traduisant une forme d'**utilité**, de **coopération** ou de **dépendance constructive** entre deux entités. Elle peut relever :

- d'une **distance** (inversée),
- d'une **similarité** (corrélation, cosinus),
- d'une **information mutuelle** plus élaborée,
- ou de tout critère d'**interaction** mutuelle validé par le contexte.

Son rôle consiste à **guider** la dynamique d'auto-organisation du réseau, car seules les paires (i, j) présentant une synergie élevée verront leurs liaisons $\omega_{i,j}$ se renforcer. Les chapitres 2.2.2 et 2.2.3 décriront comment la mise à jour de $\omega_{i,j}(t)$ s'appuie sur $S(i, j)$ pour donner naissance à un **réseau** (SCN) adaptatif, capable de **former** des **clusters** et de s'auto-organiser autour des entités les plus bénéfiques les unes pour les autres, marquant ainsi l'essence du **Deep Synergy Learning**.

2.2.2. Mise à Jour des Pondérations : Formule Générale

Le **Deep Synergy Learning** (DSL) s'appuie sur un réseau d'entités $\{\mathcal{E}_i\}$ dont les liaisons $\omega_{i,j}$ évoluent en fonction d'une **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. La spécificité du DSL réside dans le fait que ces pondérations ne sont pas fixées ni mises à jour par un unique label externe, mais bien par un **processus local** qui adapte la force de chaque lien $\omega_{i,j}$ selon la valeur de synergie mesurée entre \mathcal{E}_i et \mathcal{E}_j .

Dans cette section (2.2.2), on formalise la **règle de mise à jour** la plus courante dans la littérature DSL : une **équation linéaire-additive**, souvent comparée aux règles de Hebb modifiées ou aux dynamiques d'un réseau associatif.

2.2.2.1. Équation $\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i, j) - \tau \omega_{ij}(t)]$

Dans le cadre du **Deep Synergy Learning** (DSL), on envisage un **indice de temps** discret $t = 0, 1, 2, \dots$, ainsi qu'une matrice de pondérations $\{\omega_{i,j}(t)\}$. La mise à jour de ces pondérations, à chaque itération, s'effectue selon la formule

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta[S(i,j) - \tau \omega_{ij}(t)].$$

Les différentes composantes de cette équation se lisent comme suit :

- $\omega_{ij}(t)$ désigne la **force** (ou le poids) de la liaison entre les entités \mathcal{E}_i et \mathcal{E}_j à l'instant t .
- $S(i,j)$ désigne la **synergie** (positive ou nulle) entre \mathcal{E}_i et \mathcal{E}_j .
- $\eta > 0$ est un **taux d'apprentissage** local, traduisant la rapidité d'ajustement.
- $\tau > 0$ correspond à un **coefficent de décroissance** (ou de dissipation) qui prévient la croissance indéfinie des pondérations.

Cette équation se décompose en deux mécanismes complémentaires :

49. **Renforcement** : la pondération ω_{ij} subit un incrément $\eta S(i,j)$. Si la synergie $S(i,j)$ se révèle élevée, la liaison ω_{ij} s'accroît.
50. **Décroissance** : le terme $\eta \tau \omega_{ij}(t)$ vient en déduction, évitant ainsi que les poids ω_{ij} ne divergent et ne deviennent arbitrairement grands.

On peut reformuler l'expression en

$$\omega_{ij}(t+1) = (1 - \eta \tau) \omega_{ij}(t) + \eta S(i,j).$$

Cette représentation met en évidence une **combinaison linéaire** de l'ancienne valeur $\omega_{ij}(t)$, multipliée par $(1 - \eta \tau)$, et d'un **terme constant** $\eta S(i,j)$. Pour l'analyse de la convergence, on peut résoudre l'équation stationnaire

$$\omega_{ij}^* = (1 - \eta \tau) \omega_{ij}^* + \eta S(i,j),$$

qui donne

$$\omega_{ij}^* = \frac{\eta}{\eta \tau} S(i,j) = \frac{S(i,j)}{\tau}.$$

Cette solution indique que, si l'on itère la mise à jour et si les valeurs de η et τ sont suffisamment réduites ou bien paramétrées, $\omega_{ij}(t)$ se **stabilise** autour d'un point d'équilibre $\frac{S(i,j)}{\tau}$. Dans les chapitres suivants — notamment *Chapitre 4* et *Chapitre 7* — on examinera plus avant la **convergence** et la façon dont cette dynamique collective (appliquée simultanément à toutes les paires (i,j)) peut conduire à la formation de **clusters**.

Conclusion

La formule ci-dessus illustre de façon limpide le **double mécanisme** propre au DSL :

- L'**aspect “renforcement”** renforce ω_{ij} proportionnellement au score $S(i,j)$.
- L'**aspect “décroissance”** introduit un frein $\tau \omega_{ij}(t)$, évitant l'explosion des liaisons.

La répétition de cette règle itérative, pour $t = 0, 1, 2, \dots$, construit progressivement un **réseau** (SCN) au sein duquel les liens ω_{ij} se **stabilisent** là où la synergie se maintient, tandis qu'ils se réduisent dans les zones où la synergie demeure faible. Les sections 2.2.2.2 et 2.2.2.3 présenteront les **variantes** (mise à jour multiplicative, inhibition compétitive) ainsi que le **raccord** biologique avec la plasticité neuronale.

2.2.2.2. Variantes : Inhibition Compétitive, Mécanisme Multiplicatif vs. Additif

La **formule générale** de mise à jour

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)] \quad (\text{voir section [2.2.2.1]})$$

illustre le **cas additif**, dans lequel on ajoute un terme $\eta S(i,j)$ et on retire $\eta \tau \omega_{ij}(t)$. Dans la pratique du **Deep Synergy Learning (DSL)**, on rencontre toutefois des **mécanismes de régulation supplémentaires** ou alternatifs, qui modifient la nature de l'adaptation des poids ω_{ij} . Deux **familles de variantes** se distinguent : l'**inhibition compétitive** et le **mécanisme multiplicatif** (plutôt qu'une simple mise à jour de type linéaire-additif).

A. Inhibition Compétitive

La **mise à jour linéaire** du poids ω_{ij} présentée en section 2.2.2.1 traite chaque liaison de façon **indépendante**. Or, il est possible d'introduire une **inhibition compétitive**, c'est-à-dire un **effet d'interaction** entre différents liens connectés à la même entité, de sorte que la croissance de certains empêche ou restreigne la croissance d'autres. Dans un réseau biologique, ce phénomène correspond à l'**inhibition latérale** : un neurone fortement actif peut inhiber la plasticité d'autres connexions concurrentes (ou proches).

Dans le **DSL**, l'inhibition compétitive peut prendre la forme de **contraintes** ou de **termes** interagissant avec ω_{ij} . Deux approches classiques :

51. Somme limitée :

$$\sum_j \omega_{ij}(t) \leq \Omega_{\max},$$

imposant qu'une entité \mathcal{E}_i ne puisse maintenir qu'un nombre total (ou une somme totale) de liaisons $\omega_{i\cdot}$ au-dessus d'un certain seuil.

52. Terme d'inhibition introduit directement dans la règle de mise à jour :

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \left[S(i,j) - \tau \omega_{ij}(t) - \sum_{k \neq j} \gamma_{k,j} \omega_{ik}(t) \right],$$

où $\gamma_{k,j}$ pondère l'**inhibition** exercée par le lien ω_{ik} sur la liaison ω_{ij} . Ainsi, plus un **nœud** \mathcal{E}_i a de liaisons puissantes, plus il "freine" la progression de nouvelles connexions.

Cette inhibition compétitive favorise la **sparcification** du réseau et l'apparition de **clusters** plus nets. Même si la synergie $S(i,j)$ conserve une valeur positive, la liaison ω_{ij} peut être contrainte de **décroître** ou de stagner si la "capacité" du nœud \mathcal{E}_i (ou \mathcal{E}_j) est déjà mobilisée par d'autres liens

plus cruciaux. On retrouve ce principe dans divers modèles biologiques et algorithmes inspirés de la neurophysiologie, où il améliore l'**efficacité** de l'apprentissage en évitant un maillage trop complet ou trop diffus du réseau.

B. Mécanisme Multiplicatif vs. Additif

La règle générale

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta[S(i,j) - \tau \omega_{ij}(t)]$$

relève d'une **dynamique additive** : on incrémente ω_{ij} d'un terme $\eta S(i,j)$ et on décrémente d'un terme $\eta \tau \omega_{ij}(t)$. Dans certaines approches du **DSL**, on privilégie un **mécanisme multiplicatif** où l'augmentation ou la diminution du poids ω_{ij} s'effectue de façon **proportionnelle** à sa valeur courante. Par exemple, on peut adopter une formule telle que

$$\omega_{ij}(t+1) = \omega_{ij}(t) \times (1 - \alpha \tau) + (\text{termes multiplicatifs de } S(i,j)),$$

ou, plus radicalement,

$$\omega_{ij}(t+1) = \omega_{ij}(t) \exp(\alpha[S(i,j) - \tau \omega_{ij}(t)]).$$

Ce style de mise à jour rappelle la **règle de Hebb** : “plus un poids est grand, plus il se renforce lors de co-activation”. Il favorise un **effet exponentiel** : un lien déjà élevé grandit rapidement si la synergie $S(i,j)$ reste soutenue, tandis qu'un lien faible ne se développe que modérément.

Sur le plan algorithmique, cela peut accélérer la **différenciation** entre liens “dominants” et “inactifs”, donnant lieu à un phénomène “winner-takes-most”. Même si, dans son essence, la logique multiplicative conserve l'idée d'ajuster $\omega_{ij}(t)$ en fonction de $S(i,j)$, elle introduit un **degré** de non-linéarité plus important. La croissance (ou décroissance) dépend non seulement de $S(i,j)$ mais aussi de la magnitude actuelle de ω_{ij} .

Conclusion sur les Variantes

Les règles de mise à jour **diffèrent** de la simple formule linéaire-additive en introduisant soit un terme d'**inhibition compétitive** (permettant une allocation restreinte des ressources du nœud) soit un **mécanisme multiplicatif** (renforçant la proportionnalité de l'apprentissage). Dans un **Synergistic Connection Network (SCN)**, ces variantes :

- **Enrichissent** la dynamique du **DSL** en prévenant la prolifération de liens peu utiles (inhibition) ou en amplifiant la séparation entre liens forts et liens faibles (multiplicatif),
- **Permettent** de mieux contrôler la structure émergente, en assurant la **formation** de clusters nets,
- **Restent** conformes au principe d'**auto-organisation** : la mise à jour demeure un calcul **local** dicté par $S(i,j)$ et ω_{ij} , sans supervision externe.

Les sections 2.2.2.3 et suivantes préciseront l'**interprétation** de ces variations en se référant, par exemple, aux analogies biologiques ou aux conséquences sur la **formation de clusters** dans un réseau DSL.

2.2.2.3. Interprétation : Plasticité Locale, Renforcement/Désactivation de Liens

La **mise à jour** des poids $\omega_{i,j}(t)$ guidée par la **synergie** $S(i,j)$ peut s'interpréter comme un **mécanisme de plasticité** analogue aux dynamiques neuronales ou aux paradigmes d'auto-organisation. Dans ce cadre, chaque liaison $\omega_{i,j}$ se **renforce** ou s'**affaiblit** de manière autonome, en fonction de la valeur de synergie mesurée entre les entités \mathcal{E}_i et \mathcal{E}_j , et sous l'impact d'un terme de décroissance régulateur. Cette section [2.2.2.3] propose une **lecture** (à la fois biologique et algorithmique) de cette dynamique, généralement décrite comme un **processus local** de renforcement/désactivation, fondé sur la co-occurrence ou l'interaction directe entre entités.

A. Plasticité Locale

Dans les **réseaux** neuronaux biologiques, la **plasticité** évoque la capacité d'une synapse à ajuster sa "force" en réaction à l'activité simultanée (ou corrélée) de ses neurones pré- et post-synaptiques. L'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

(voir section 2.2.2.1) illustre cette logique : si la **synergie** $S(i,j)$ est élevée, elle accroît la liaison $\omega_{i,j}$ d'un terme $\eta S(i,j)$.

Cette **variation** $\Delta\omega_{i,j}$ dépend exclusivement de $\omega_{i,j}(t)$ et de la valeur $S(i,j)$, conformément à la philosophie **locale** de la plasticité : il n'y a ni connaissance **globale** ni label supervisé. C'est précisément l'**auto-organisation** : l'ajustement s'opère localement, et la configuration **globale** (éventuelle formation de clusters) émane de l'ensemble de ces modifications éparses. Dans un scénario DSL standard, on découpe le temps en itérations $t = 0, 1, 2, \dots$. Chaque itération applique la règle de plasticité. Dans une version plus biologisante, on passerait en **temps continu** :

$$\frac{d \omega_{i,j}}{dt} = \alpha[S(i,j) - \tau \omega_{i,j}],$$

ce qui revient conceptuellement à la même logique et, en l'absence de perturbations, tend vers l'équilibre $\omega_{i,j}^* \approx S(i,j)/\tau$.

B. Renforcement vs. Désactivation

Si la quantité $\tau \omega_{i,j}(t)$ reste **inférieure** à $S(i,j)$, alors la mise à jour $\Delta\omega_{i,j} = \eta[S(i,j) - \tau \omega_{i,j}(t)]$ s'avère **positive** : le poids $\omega_{i,j}$ **croît**. Cela signifie que, tant que la **synergie** $S(i,j)$ surpassé un "seuil relatif" $\tau \omega_{i,j}(t)$, la liaison se **renforce** pas à pas. De multiples itérations rapprochent $\omega_{i,j}$ d'un **équilibre** $\omega_{i,j}^* \approx S(i,j)/\tau$.

Si, en revanche, $\tau \omega_{i,j}(t)$ dépasse $S(i,j)$, la variation $\Delta\omega_{i,j}$ devient **négative**. Dans ce cas, la connexion $\omega_{i,j}$ se **décroît** progressivement, trahissant une **absence** ou un affaiblissement de la

synergie justifiant un grand poids. À la longue, $\omega_{i,j}$ peut tendre vers une valeur très faible ou proche de zéro, synonyme de désactivation.

Cette dynamique explique pourquoi $\omega_{i,j}^* = S(i,j)/\tau$ incarne un **point d'équilibre** local : le renforcement et la décroissance s'y compensent exactement.

C. Implications pour la Structure Émergente

Lorsque certains liens $\omega_{i,j}$ s'avèrent renforcés (synergie $S(i,j)$ élevée) et que d'autres s'amenuisent (faible synergie, ou trop fortement pénalisés), le **réseau** voit se constituer des **sous-groupes** : on observe des **clusters** (ou micro-clusters) fortement interconnectés. Cela correspond à la formation d'un **état stable** où les entités \mathcal{E}_i et \mathcal{E}_j présentant une synergie notable sont reliées par des pondérations robustes.

Cette approche ne se borne pas à une structure figée, puisque si les **synergies** $\{S(i,j)\}$ évoluent (par exemple si le système reçoit de nouvelles entités, ou si les co-occurrences se modifient), la **plasticité locale** réactualise $\omega_{i,j}$. Le réseau suit donc les **fluctuations** du flux, créant ou dissolvant des connexions selon la tendance du moment (cf. Chapitre 9 sur l'apprentissage continu). Dans la sphère **biologique**, cette logique de co-activation et d'**adaptation** des liaisons renvoie à la **règle de Hebb** ou à d'autres mécanismes (STDP) où les synapses se renforcent si leurs neurones sont fortement corrélés. En IA non supervisée, ces principes rejoignent ceux des **réseaux associatifs**, mais le **DSL** porte l'idée encore plus loin en permettant des fonctions de synergie plus complexes (mesures multimodales, co-information...).

Conclusion sur la Plasticité Locale

La règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

peut se lire comme un **processus de plasticité locale**, où la synergie S tient le rôle d'un "signal de co-opération". Les liens présentant un **score** $S(i,j)$ élevé se **renforcent**, tandis que ceux dépourvus de synergie s'amenuisent ou se désactivent (tendent vers zéro). De fait, un réseau DSL s'oriente vers une **sélection** continue des connexions jugées utiles, maintenant une certaine **résilience** (on peut réactiver un lien si la synergie remonte ultérieurement).

Les chapitres ultérieurs examineront la **convergence** de ces poids et la **stabilisation** d'une structure globale (clusters, attracteurs). Ils aborderont également, dans la section 2.2.2.2, les **variantes** : inhibition compétitive (pour introduire une interaction entre différents liens) et mise à jour multiplicative (pour renforcer l'effet "winner-takes-most"). L'ensemble de ces points ancre la **philosophie** du Deep Synergy Learning : s'appuyer sur une **plasticité locale** pilotée par la synergie pour favoriser l'**émergence** de configurations organisées dans le réseau.

2.2.3. Règles de Parsimonie et Seuil de Connexion

Au fur et à mesure que la **mise à jour** (section 2.2.2) renforce certains liens $\omega_{i,j}$ selon la synergie $S(i,j)$, le **réseau** risque de voir ses pondérations croître un peu partout. Or, dans de nombreux scénarios — qu'il s'agisse de **biologie** (réseaux neuronaux) ou de **DSL** pratique (auto-organisation hétérogène) —, il est souhaitable d'éviter un **graphe trop dense**, qui peut entraîner :

- Une **surcharge** de calcul (trop de liaisons à gérer, complexité accrue),
- Un **manque** de différenciation (le réseau devient quasi complet, il est difficile d'identifier de vrais clusters distincts),
- Des **effets** d'instabilité ou de bruit (des liens résiduels de faible poids saturent inutilement la structure).

La notion de **parsimonie** répond à cela : on ajoute des **règles** (post-traitements ou intégrées à la mise à jour) qui suppriment ou inhibent les liaisons trop faibles, favorisant une structure plus éparsé et plus significative.

2.2.3.1. Comment Éviter un Graphe Trop Dense : Suppression de Liens sous un Certain ω_{\min}

Dans l'application pratique du **Deep Synergy Learning**, on adopte souvent un **seuil** $\omega_{\min} > 0$ pour **restreindre** la densité du réseau. Concrètement, après chaque itération ou à des intervalles déterminés, on **supprime** (ou met à 0) les liaisons dont les pondérations $\omega_{i,j}$ demeurent **inférieures** à ω_{\min} . La règle s'écrit :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } \omega_{i,j}(t+1) \geq \omega_{\min}, \\ 0, & \text{sinon.} \end{cases}$$

Ainsi, les liaisons trop faibles $\omega_{i,j} < \omega_{\min}$ sont ramenées à zéro, puis (en fonction de la stratégie adoptée) ne sont plus mises à jour ou restent inactives jusqu'à un éventuel regain de **synergie**. En procédant ainsi, on obtient un **réseau** plus **clairsemé**, simplifiant l'identification de **clusters** et la visualisation globale. Sur le plan algorithmique, la démarche se résume à parcourir la matrice ω après un certain nombre d'itérations (ou après chaque “epoch”) pour **couper** les liaisons trop faibles, ce qui libère les ressources de calcul pour les liens plus importants.

Cette suppression de liens de force négligeable rappelle le **synaptic pruning** dans les réseaux neuronaux biologiques, où des synapses à faible activité finissent par être éliminées, permettant ainsi de recentrer l'efficacité du neurone sur les connexions plus utiles. Dans l'optique d'un **DSL**, un tel mécanisme garantit la **parsimonie** : le réseau se concentre sur les liaisons suffisamment pertinentes, évitant de multiplier les connexions de poids quasi nul. Il existe plusieurs variantes pour implanter ce principe :

- **Seuil unique** ω_{\min} .
- **Seuils multiples** $\omega_{\min 1}, \omega_{\min 2} \dots$, autorisant un tri par “niveaux” de liens.
- **Pourcentage fixe** : ne conserver que les $k\%$ de liaisons les plus fortes.

Ces méthodes se montrent particulièrement **bénéfiques** quand le nombre d'entités n devient grand, car la **matrice** $\{\omega_{i,j}\}$ peut atteindre une taille $O(n^2)$. En réduisant la densité des connexions actives, on **allège** la complexité computationnelle et on **accélère** la convergence de la structure.

Conclusion (2.2.3.1)

Le fait de **supprimer** les liaisons dont $\omega_{i,j} < \omega_{\min}$ introduit une **parsimonie** salutaire dans le réseau, soutenant une **lisibilité** accrue et un **allègement** des calculs. Les sections [2.2.3.2](#) et [2.2.3.3](#) approfondiront les **conséquences** de ce filtrage (amélioration de la clarté des clusters, stabilisation plus rapide) et examineront son **impact** sur la complexité (réduction du nombre de liens à gérer, risque éventuel de négliger certaines synergies modestes mais potentiellement utiles).

2.2.3.2. Effets sur la Structure Émergente (Clusters plus Nets, Stabilisation plus Rapide)

Lorsque la **règle de parsimonie** (section [2.2.3.1](#)) impose de couper ou de désactiver les liaisons $\omega_{i,j}$ inférieures à un certain seuil ω_{\min} , la **topologie** du Synergistic Connection Network (SCN) se réoriente vers un réseau plus **clairsemé**. Ce filtrage sélectif engendre plusieurs **conséquences** majeures sur la **structure** formée et sur la **dynamique** de stabilisation qui en découle.

A. Clusters plus Nets

En retranchant systématiquement les liaisons $\omega_{i,j}$ dont la valeur demeure **en deçà** de ω_{\min} , on élimine de fait les **ponts** minimes ou les connexions relativement faibles qui ne concourent pas sensiblement à la structure globale. Les liens dont $\omega_{i,j}$ dépasse ω_{\min} deviennent alors les **véritables connexions**, celles qui suggèrent une **synergie** réelle entre les entités \mathcal{E}_i et \mathcal{E}_j .

Dans une démarche d'**auto-organisation**, on recherche souvent des **clusters** de nœuds fortement interconnectés. La présence d'un seuil ω_{\min} clarifie et accentue la séparation entre ces groupements : les liaisons inter-groupes — si elles restent faibles — se trouvent coupées, tandis que les liens intra-groupe, favorisés par une **synergie** non négligeable, franchissent le cap du seuil. Cette **polarisation** entre liens conservés et liens supprimés renforce la visibilité des **composantes** ou des **communautés** dans le SCN.

Par ailleurs, les liaisons de faible intensité pouvaient osciller au cours des itérations, sans réellement influer sur l'architecture. Les supprimer réduit le risque d'**agrégation** inutile de micro-connexions parasites, et limite par conséquent la formation de signaux aléatoires qui alourdiraient le réseau. De fait, cette **coupure** s'apparente à un **filtrage** : elle fait ressortir les liens les plus significatifs et, par extension, rend la structure plus facilement interprétable.

B. Stabilisation plus Rapide

Le fait de mettre à zéro une bonne partie des liaisons $\omega_{i,j}$ jugées trop faibles diminue la **charge** dans la dynamique du réseau. Dès lors, seuls subsistent un nombre restreint de poids **actifs**, sur lesquels se concentrent les mises à jour. En conséquence, la boucle de réajustement se révèle plus **efficace** : il y a moins d'interactions à recalculer, et beaucoup d'anciennes connexions (désormais à 0) ne requièrent plus de suivi continu.

Si, de surcroît, les interactions inter-blocs (ou inter-clusters) étaient de toute façon faibles, elles pouvaient allonger la **convergence** en imposant de petits allers-retours de pondérations. Leur coupure accélère la stabilisation, laissant chaque cluster se structurer **indépendamment**, sans perturbations marginales.

Du point de vue purement algorithmique, si la mise à jour d'une liaison $\omega_{i,j}$ s'exprime (cf. section 2.2.2.1) par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

la suppression régulière des liens $\omega_{i,j}$ restés en dessous de ω_{\min} apporte une plus grande **stabilité** : une fois coupées, ces connexions ne fluctuent plus, et les liens restants convergent généralement plus vite vers leur point d'équilibre $\omega_{i,j}^* \approx S(i,j)/\tau$. De nombreux travaux empiriques montrent que l'**auto-organisation** se fait plus rapidement quand on élimine régulièrement les connexions trop faibles.

C. Conclusion sur l'Impact des Règles de Parsimonie

Le simple fait d'imposer un **seuil** ω_{\min} (ou toute autre technique de **sparsification**) n'est pas anodin. Il produit des **effets bénéfiques** tant sur la **structure** émergente que sur la **dynamique** de stabilisation :

- Les **clusters** s'affichent de manière **plus nette**, car les liaisons modestes sont coupées et les communautés se démarquent plus facilement.
- La **stabilisation** s'en trouve **accélérée**, puisque l'on maintient un réseau plus léger, évitant l'encombrement par de multiples pondérations moyennes ou faibles.

Ces avantages expliquent pourquoi, dans la pratique du **DSL**, on utilise couramment une procédure de “**thresholding**” (ou “**cut-off**”) pour rendre le **graphe** plus clairsemé et plus interprétable, tout en aidant la convergence. La prochaine section (2.2.3.3) analysera l'**impact sur la complexité** de cette coupe (réduction substantielle du nombre de liaisons à traiter, contrepartie du risque de négliger certaines synergies marginales mais potentiellement utiles).

2.2.3.3. Impact sur la Complexité : Gain Potentiel vs. Perte d'Information

Dans le contexte d'un **Synergistic Connection Network (SCN)**, l'application d'une règle de **parsimonie** (voir section 2.2.3.1) conduit à **supprimer** ou à **mettre à zéro** les liaisons $\omega_{i,j}$ dont la valeur reste inférieure à un certain seuil ω_{\min} . Lorsque cette procédure est mise en œuvre, le **réseau** acquiert une structure plus **clairsemée** : nombre de connexions sont annihilées, seules subsistant celles qui dépassent ω_{\min} . Cette raréfaction des liens exerce une influence directe sur la **complexité** de l'architecture, ainsi que sur la **qualité** et l'**ampleur** de l'information qui transite dans le SCN. D'une part, la disparition de nombreuses liaisons apporte un **gain** notable en termes de densité et de coûts de calcul ; d'autre part, on risque de se priver de connexions faiblement positives susceptibles de s'avérer utiles. Le présent exposé souligne ces **deux facettes** — la simplification bienvenue et la perte potentielle — et examine les **stratégies** ou **règlages** classiquement envisagés pour concilier **parsimonie** et **préservation** de l'information.

A. Gains en Termes de Complexité

Il est fréquent de considérer un ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Si toutes forment un graphe complet, le nombre de liaisons $\omega_{i,j}$ à mettre à jour peut tendre vers $O(n^2)$. Imposer un **seuil** $\omega_{\min} >$

0 pour **couper** systématiquement les connexions dont la pondération reste au-dessous de ω_{\min} à pour effet :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } \omega_{i,j}(t+1) \geq \omega_{\min}, \\ 0, & \text{sinon.} \end{cases}$$

Lorsque cette coupure est appliquée, la **densité** du réseau chute. De nombreuses paires (i, j) reçoivent désormais une pondération nulle et ne participent plus aux calculs, offrant une **réduction** de la complexité globale :

Complexité passera de $O(n^2)$ à $O(|\mathcal{A}|)$,

où \mathcal{A} est l'ensemble des **liaisons actives**, c'est-à-dire celles qui franchissent le seuil ω_{\min} . Si ce nombre se révèle beaucoup plus modeste que n^2 , la mise à jour se voit grandement **allégée**. Outre le gain algorithmique, ce filtrage favorise la **lisibilité** du réseau, par exemple pour la **détection** et la **visualisation** de clusters, en prévenant la formation d'un graphe trop dense. Il devient ainsi aisément d'identifier les sous-ensembles d'entités réellement interconnectées.

Avantages en un seul paragraphe. L'avantage majeur de cette approche réside dans la diminution drastique des liaisons à gérer, accélérant chaque itération de la mise à jour locale tout en clarifiant la topologie pour l'analyse des clusters. On évite de mobiliser un temps de calcul inutile sur des connexions dont la valeur de pondération reste marginale. Cette simplification s'avère particulièrement pertinente dans des contextes de grande dimension (big data, multimodalité), où la réduction de $O(n^2)$ à $O(n)$ ou $O(n \log n)$ peut faire la différence entre un algorithme inopérant et une solution réalisable en pratique.

B. Risque de Perte d'Information

Si la *parsimonie* via le seuil ω_{\min} procure un gain, elle entraîne le **risque de couper** des liaisons modérément positives qui auraient pu devenir utiles si on leur laissait le temps de se renforcer. Un lien $\omega_{i,j} = 0.02$ peut sembler négligeable à l'instant présent, mais il aurait pu se développer ultérieurement, si la **synergie** $S(i, j)$ augmentait pour des raisons liées à un flux de données changeant. Imposer $\omega_{i,j} \rightarrow 0$ quand $\omega_{i,j} < \omega_{\min}$ empêche de ressusciter la liaison à moins qu'on ne prévoie une procédure de "**réactivation**", tolérant le retour d'un lien si $S(i, j)$ devient subitement plus fort. Sans un tel mécanisme, la décision de couper est souvent définitive ou du moins très contraignante, ce qui peut figer la structure avant qu'une coopération tardive n'ait pu émerger.

Limites en un seul paragraphe. Cette coupure radicale peut aussi segmenter prématûrement le réseau en multiples composantes déconnectées, s'il se trouve que la synergie reste faible mais non négligeable entre certains groupes. Un seuil trop ambitieux (ω_{\min} trop grand) peut donc conduire à une perte de richesse dans l'exploration des configurations, voire empêcher la coalescence ultérieure de clusters apparentés. Cela constitue la principale contrepartie de la parcimonie, tout en voulant gagner en efficacité, on sacrifie la possibilité de liens modestes mais porteurs d'une information subtile.

C. Stratégies de Coupure et Paramétrages

Il est crucial de trouver un **équilibre** entre l'**exigence** de parsimonie et la **volonté** de préserver les connexions potentielles. Plusieurs méthodes se révèlent utiles :

Adaptation dynamique du seuil ω_{\min} où l'on commence avec une valeur modeste, puis on l'augmente progressivement selon l'avancement de l'apprentissage. On peut également recourir à :

Coupes partielles, plutôt que d'annihiler un lien en dessous de ω_{\min} , on le réduit (ex. $\omega_{i,j} \rightarrow \gamma \omega_{i,j}$) afin de laisser une “chance” de relance. Des mécanismes de “réactivation” peuvent se mettre en place si la synergie mesurée $S(i,j)$ s’élève de nouveau, réinstituant alors la liaison.

La littérature décrit également la possibilité de **conserver** un certain **pourcentage** de liens (5 % ou 10 % des poids maximaux), découpant le reste. Une autre approche consiste à procéder à des coupes plus progressives au fil des époques, ne retenant à chaque étape que les liens les plus élevés.

D. Conclusion sur le Choix du Seuil et la Cohabitation Complexité-Information

En finalité, l'introduction d'un **seuil** ω_{\min} ou d'une méthode équivalente (parsimonie structurelle, maintien d'un top-k) vise à **contenir** la densité du graphe et à **faciliter** la reconnaissance de motifs (clusters, communautés). On profite, en un seul paragraphe, d'une baisse de complexité algorithmique notoire, d'une stabilisation accélérée et d'une clarté accrue dans la cartographie du réseau, au prix d'un danger de couper précocement certaines liaisons prometteuses. Les chapitres d'optimisation et de mise à jour en temps réel (voir *Chapitre 7* et *Chapitre 9*) aborderont plus en détail les heuristiques adoptées pour calibrer ω_{\min} , proposer des palliatifs comme la réactivation de liens et minimiser la “perte” de signaux authentiquement utiles. L'expérience montre que des **ajustements** judicieux, éventuellement adaptatifs, parviennent souvent à concilier les **avantages** (réduction drastique du nombre de liens, stabilisation rapide, meilleure lisibilité) et les **limites** (risque de négliger des connexions latentes ou futures) inhérents à la coupure.

2.2.4. Notions d'États Internes et Auto-Organisation

Dans l'architecture d'un **Deep Synergy Learning (DSL)**, chaque entité \mathcal{E}_i est généralement décrite par son ancrage dans un espace \mathcal{X} (vecteur, symbole, distribution, etc.) et par le poids de ses liaisons $\{\omega_{i,j}\}$. Cependant, il est souvent utile d'aller plus loin en autorisant chaque entité à entretenir un **état local** — c'est-à-dire un ensemble de variables ou de “mémoire” qui évolue **en parallèle** de la dynamique des liaisons. Cette idée de “double composante” (poids inter-entités et état interne) contribue à renforcer la capacité d'**auto-organisation**, car elle ajoute un degré de plasticité propre à chaque entité, sans que tout dépende uniquement des pondérations $\omega_{i,j}$. La section (2.2.4) détaille comment ce mécanisme peut se mettre en place et quelles conséquences il a sur l'émergence de schémas cognitifs plus riches.

2.2.4.1. Possibilité pour Chaque Entité \mathcal{E}_i de Stocker un État Local, Évoluant en Parallèle

Il est souvent utile, dans un **Synergistic Connection Network (SCN)**, de ne pas cantonner chaque entité \mathcal{E}_i à une représentation fixe (par exemple, un simple vecteur \mathbf{x}_i demeurant invariant). On peut en effet postuler qu'elle possède un **état interne** $\mathbf{s}_i(t)$ qui se **modifie** au fil des itérations $t = 0, 1, 2, \dots$. Ainsi, l'évolution ne concerne plus uniquement les pondérations $\omega_{i,j}$ (voir la section 2.2.2), mais aussi le **cycle** propre de chaque entité.

A. Principe Général

On conserve la mise à jour des poids $\omega_{i,j}(t)$ conformément à la règle (voir section 2.2.2) :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$$

En **parallèle**, chaque entité \mathcal{E}_i actualise son **état** $\mathbf{s}_i(t)$ en prenant en compte, par exemple, les informations issues du réseau ou de son activation interne. Il peut exister un ensemble d'équations décrivant la dynamique de l'état local :

$$\mathbf{s}_i(t+1) = \mathbf{F}_i(\mathbf{s}_i(t), \{\omega_{i,j}(t)\}, \{\mathbf{s}_j(t)\}, \dots).$$

La **fonction** \mathbf{F}_i dépend alors de l'état précédent $\mathbf{s}_i(t)$, des pondérations $\omega_{i,j}(t)$ reliant \mathcal{E}_i à ses voisins, et éventuellement d'autres signaux extérieurs. Comme dans toute **auto-organisation**, le principe demeure local, chaque entité mettant à jour son état sans qu'un contrôleur global ne dicte de consignes.

Ce postulat accroît sensiblement la **souplesse** du système. Les entités ne sont plus de simples "nœuds passifs" mais deviennent des **agents** disposant d'une **mémoire** ou d'un **contexte** interne. Elles peuvent ainsi **s'adapter** à l'historique, éventuellement conserver des traces de données observées précédemment, moduler la façon dont elles calculent leur synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ vis-à-vis d'autres entités, ou encore faire varier leur "réceptivité" selon leur état $\mathbf{s}_i(t)$. Cela introduit un **couplage** (rétroaction) entre l'état de l'entité et la pondération $\omega_{i,j}$.

B. Richesse des Comportements

Le fait que les entités \mathcal{E}_i puissent stocker un **état** $\mathbf{s}_i(t)$ ouvre la voie à des dynamiques plus **complexes**, proches d'un système **multi-agents** ou de **modules cognitifs**. Par exemple, une entité pourrait se comporter en **LSTM local** (voir section 2.2.4.2) ou en bloc mémoire plus rudimentaire. Elle peut assimiler son historique, se doter d'un "contexte" conceptuel, ou enregistrer des **résidus** de perception. Cette organisation "double" (adaptation des poids $\omega_{i,j}$ d'un côté, adaptation des états $\mathbf{s}_i(t)$ de l'autre) rappelle certaines structures neuronales biologiques, où la **plasticité** synaptique se combine à l'activité **interne** des neurones (ou groupes de neurones).

Dans le **DSL**, cette évolution conjointe favorise l'émergence de schémas cognitifs distribués. Le réseau ne se limite plus à une **clusterisation** statique. Il peut générer de véritables **boucles** d'activité, des phénomènes de **résonance** ou de **synchronisation**, dans lesquels l'actualisation de $\mathbf{s}_i(t)$ influence la **synergie** $S(i, j)$, laquelle, à son tour, oriente la mise à jour $\omega_{i,j}(t)$. Ce double plan adaptatif donne un pouvoir expressif supplémentaire : un nœud \mathcal{E}_i peut modifier sa **sensibilité** ou son "profil" d'interaction selon ses signaux internes, aboutissant à une **co-organisation** encore plus riche.

On peut écrire : il existe deux niveaux de plasticité qui interagissent,

$$\begin{cases} \omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots, \\ \mathbf{s}_i(t+1) = \mathbf{F}_i(\mathbf{s}_i(t), \{\omega_{i,j}\}, \dots). \end{cases}$$

C'est cette combinaison qui confère au **DSL** son potentiel pour modéliser des fonctionnements plus proches d'une cognition distribuée.

C. Conclusion sur l'Intérêt d'un État Local

L'incorporation, chez chaque entité \mathcal{E}_i , d'un **état interne** $s_i(t)$ rehausse de beaucoup la **polyvalence** de l'auto-organisation. Les entités ne se réduisent plus à des **identifiants** passifs reliés par des poids : elles deviennent des **agents** disposant d'une **mémoire** propre et d'une **plasticité** interne, suivant un cycle parallèle à celui des poids $\omega_{i,j}(t)$. Cette stratégie autorise des comportements dynamiques plus élaborés, la mise en œuvre de **modules cognitifs** (par exemple, "mini-LSTM"), et la création de boucles d'activité auto-organisées (cf. section 2.2.4.2 pour des exemples concrets, et section 2.2.4.3 pour le lien avec l'émergence de schémas cognitifs). En accordant à chaque entité la faculté de faire évoluer son **état**, on confère à l'ensemble du SCN une plus grande **richesse** : les liaisons $\omega_{i,j}$ ne sont plus seules à se modifier, les **entités** elles-mêmes possèdent un cycle d'apprentissage local. Cette double plasticité (liens + états) accroît la capacité du réseau à se **reconfigurer**, à **mémoriser** des patterns récurrents et à **raisonner** de façon distribuée.

2.2.4.2. Exemple : LSTM Local ou Petite Mémoire Interne

Lorsque l'on propose d'associer à chaque entité \mathcal{E}_i un **état interne** (se reporter à la section 2.2.4.1), plusieurs voies s'offrent à l'expérimentateur. Certaines approches recourent à un simple **accumulateur** ou à une **variable scalaire**, tandis que d'autres adoptent des dispositifs plus évolués, tels qu'une **cellule LSTM** (Long Short-Term Memory) inspirée des réseaux récurrents profonds. Dans toutes ces configurations, l'objectif demeure de doter chaque entité d'une **capacité à stocker** et **transformer** les informations de façon autonome, au-delà du seul ajustement des poids $\omega_{i,j}$. Les développements ci-dessous illustrent deux options : l'usage d'un **LSTM local** et la création d'une **petite mémoire interne** moins élaborée mais déjà profitable.

A. LSTM Local au Sein d'un DSL

Un **LSTM** est une cellule récurrente conçue pour gérer des dépendances de long terme dans les réseaux neuronaux séquentiels. Elle incorpore :

- un **état caché** $\mathbf{h}(t)$ qui traduit la sortie instantanée de la cellule,
- un **état mémoire** $\mathbf{c}(t)$ où s'enregistrent des informations plus pérennes,
- plusieurs **portes** (input, forget, output) contrôlant l'apport, l'oubli et l'export de l'information contenue dans $\mathbf{c}(t)$ et $\mathbf{h}(t)$.

Les équations internes de l'LSTM déterminent l'évolution de $\mathbf{c}(t)$ et $\mathbf{h}(t)$ à partir d'une entrée $\mathbf{x}(t)$ et de l'état précédent ($\mathbf{h}(t-1), \mathbf{c}(t-1)$).

Dans le **Deep Synergy Learning**, on peut affecter à chaque entité \mathcal{E}_i sa **cellule LSTM** individuelle :

53. On retient deux **états internes** $\mathbf{h}_i(t)$ et $\mathbf{c}_i(t)$, décrivant respectivement l'état caché et la mémoire à long terme de l'entité \mathcal{E}_i .
54. À chaque itération, \mathcal{E}_i reçoit un **signal** $\mathbf{x}_i(t)$, par exemple un résumé de l'interaction issue de ses liaisons $\omega_{i,j}$ et des états $\mathbf{h}_j(t)$ de ses voisins. Elle procède alors à la mise à jour

$$(\mathbf{h}_i(t), \mathbf{c}_i(t)) = \text{LSTM}(\mathbf{x}_i(t), \mathbf{h}_i(t-1), \mathbf{c}_i(t-1)).$$

55. Le **calcul** de la synergie $S(i,j)$ peut reposer, en partie, sur $\mathbf{h}_i(t)$ et $\mathbf{h}_j(t)$. On peut, par exemple, définir

$$S(i,j) = \alpha \sim(\mathbf{h}_i(t), \mathbf{h}_j(t)) + (1 - \alpha) \sim(\mathbf{x}_i^0, \mathbf{x}_j^0),$$

où $\mathbf{x}_i^0, \mathbf{x}_j^0$ sont des représentations plus statiques des entités \mathcal{E}_i et \mathcal{E}_j . L'**état LSTM** participe donc à la détermination de la synergie.

56. Les pondérations $\omega_{i,j}$ évoluent simultanément (conformément au schéma exposé en 2.2.2.1 ou à ses variantes), fournissant un **couplage** entre la dynamique interne des entités et la plasticité des liaisons.

Sur le plan des **retombées**, un LSTM local assure une **rétention** d'information à plus long terme, permettant à l'entité \mathcal{E}_i d'**apprendre** et de s'ajuster en se fondant sur un historique étendu. Le réseau gagne en **souplesse**, car une liaison $\omega_{i,j}$ peut se consolider si les états $\mathbf{h}_i(t)$ et $\mathbf{h}_j(t)$ affichent une **similarité** grandissante. L'analogie avec la biologie neuronale (synapses + état interne) s'en trouve renforcée.

B. Petite Mémoire Interne : Variante Simplifiée

Toutes les implémentations du DSL ne requièrent pas la complexité d'un LSTM. Il est tout à fait envisageable d'utiliser une **mémoire** plus rudimentaire, par exemple un **scalaire** ou un **petit vecteur**, pour chaque entité \mathcal{E}_i . L'idée est de doter \mathcal{E}_i d'une **capacité** de rappel ou d'accumulation déjà suffisante pour enrichir les comportements auto-organisés.

Si $\mathbf{s}_i(t) \in \mathbb{R}$ (cas scalaire), on peut l'interpréter comme un accumulateur ou une trace glissante. On peut écrire :

$$\mathbf{s}_i(t+1) = (1 - \beta) \mathbf{s}_i(t) + \beta \cdot \text{input}_i(t),$$

où $\text{input}_i(t)$ représente les signaux entrants (pondérations voisines, données brutes, etc.). Cette formule confère déjà une **mémoire** à l'entité, mais ne s'encombre pas de la sophistication d'un LSTM.

Dans le cas vectoriel $\mathbf{s}_i(t) \in \mathbb{R}^d$, on applique la même logique à chaque composante, autorisant un peu plus de finesse dans la représentation.

Le calcul de la **synergie** $S(i,j)$ peut alors, en partie, reposer sur dist inverse $(\mathbf{s}_i(t), \mathbf{s}_j(t))$ ou un critère analogue, offrant un moyen de tenir compte des **états internes** dans la détermination de la coopération.

Dans bien des cas, cette approche simplifiée suffit pour ajouter un “volet temporel” ou “mémoire” au système, sans la complexité d'un LSTM complet. Toutefois, elle peut se montrer moins flexible (pas de portes d'oubli, etc.) et exiger un réglage manuel plus délicat.

C. Intérêt pour l'Auto-Organisation

Dans un **DSL**, on dispose déjà d'une **plasticité** au niveau des pondérations $\omega_{i,j}$. Ajouter une **mémoire** à chaque entité \mathcal{E}_i institue un second niveau de **plasticité** : d'une part, les liaisons

continuent d'évoluer selon la synergie, et d'autre part, les **entités** gèrent leur état $\mathbf{s}_i(t)$ en répercutant des observations passées ou des indices internes.

Cette double dynamique (liens + états) se rapproche des idées multi-agents, où chaque “agent” (une entité \mathcal{E}_i) possède un **état local** évolutif et des liaisons $\omega_{i,j}$ en constante régulation. Il en résulte des comportements plus évolués, incorporant une “intelligence” ou un “raisonnement” local distribué. Par exemple, une entité peut retenir dans \mathbf{s}_i la moyenne récente de signaux perçus, ajuster son calcul de $S(i,j)$ en conséquence, et influencer la mise à jour $\omega_{i,j}$. On voit émerger des phénomènes de synchronisation, de micro-réseaux spécialisés, voire de schémas cognitifs complexes.

D. Conclusion sur l'Intégration d'un État Local

L'introduction, dans le **Deep Synergy Learning**, d'une mémoire interne (LSTM local ou simple vecteur/scalaire) confère une **dimension supplémentaire** à l'auto-organisation. Les entités ne sont plus de simples points reliés par des poids : elles acquièrent un **cycle** propre, facilitant la **découverte** et la **stabilisation** de structures plus riches. Les liaisons $\omega_{i,j}$ demeurent ajustées par la synergie, mais cette synergie peut être modulée par le contenu interne $\mathbf{s}_i(t)$. On retrouve alors l'analogie d'agents “intelligents” dans un **SCN** dont la plasticité opère tant au niveau des connexions que des entités elles-mêmes. Ainsi, la formation de **clusters** ou de **micro-réseaux** peut s'accompagner d'un raffinement continu (mémoire, attention localisée), rappelant la notion de cognition distribuée. Le DSL gagne donc en expressivité, capable non seulement de regrouper les entités selon leur synergie, mais également de leur laisser la possibilité d'évoluer de l'intérieur, renforçant la richesse de l'**auto-organisation**. Les sections 2.2.4.3 ou les développements ultérieurs décriront plus concrètement ces potentialités, ainsi que les liens avec l'**émergence** de schémas cognitifs plus élaborés.

2.2.4.3. Lien avec l'Émergence de Schémas Cognitifs plus Riches

Lorsque chaque entité \mathcal{E}_i se trouve associée à un **état interne** (voir section 2.2.4.1) et que cette **mémoire** ou **dynamique** interne vient compléter la logique de **synergie** décrite en 2.2.4.2, on constate que le **réseau** acquiert la capacité de développer, au fil du temps, des **schémas cognitifs** plus élaborés. Autrement dit, l'**auto-organisation** ne se limite plus à la simple configuration de pondérations $\omega_{i,j}$, mais peut susciter des motifs récurrents, des regroupements conceptuels ou des **comportements** distribués analogues à la “cognition distribuée” ou aux “systèmes multi-agents cognitifs”.

Dans un **DSL** classique, on admet déjà que les entités \mathcal{E}_i ajustent (via leurs représentations ou leurs états) leurs liaisons $\omega_{i,j}$, de sorte que celles-ci se **renforcent** si la synergie demeure significative. Dès lors que chaque entité se trouve dotée d'une **composante** mémorielle ou interne, la portée de l'**auto-organisation** s'étend : chaque \mathcal{E}_i est en mesure de conserver des informations antérieures, au lieu de se cantonner à la valeur instantanée de la synergie. Par ce biais, elle peut moduler son calcul de $S(i,j)$ ou la manière dont elle relaie ses informations, en fonction d'un **état** interne $\mathbf{s}_i(t)$.

Le **réseau** ainsi construit ne se contente plus d'une clusterisation figée : il peut, de fait, développer de **véritables** réseaux de résonances, où chaque entité, grâce à sa “mémoire” ou “conscience” locale, participe à l'émergence de **motifs** globaux plus complexes. D'un point de vue

mathématique, la dynamique ne se restreint plus à l'évolution des pondérations $\omega_{i,j}(t)$; elle englobe également un sous-système $\{\mathbf{s}_i(t)\}$ dont la synchronisation ou la mise en phase peut conduire à des *routines* ou *schémas* cognitifs. On qualifie souvent un système de “cognitif” quand il gère de l'**information** (que ce soit des perceptions ou des symboles) de manière flexible, qu'il dispose d'une **mémoire** plus ou moins durable et qu'il produit des **décisions** ou **comportements** appuyés sur cet historique interne.

Dans un **DSL** renforcé par des mécanismes de mémoire (section 2.2.4.2), chaque entité \mathcal{E}_i perçoit la synergie $\omega_{i,j}$ la reliant à des entités voisines, puis met à jour $\mathbf{s}_i(t)$ en combinant l'état antérieur et de nouvelles données. Elle propage ensuite ce nouvel état (ou un résumé) dans le **réseau**, influençant la future valeur de $S(i,j)$. Ce procédé, généralisé à l'ensemble des entités, assure un **mécanisme distribué** : l'**intelligence** ou la **cognition** ne naît pas d'un composant central, mais de la coordination d'un ensemble de nœuds cognitifs, chacun adaptant en continu sa mémoire et ses liens. Ainsi, on voit émerger des **dynamiques** de synchronisation, l'apparition de “leaders” et “followers”, ou l'organisation spontanée de niveaux auto-gérés au sein d'un unique SCN.

Un *schéma* cognitif peut s'entendre comme un **pattern** récurrent, stable, servant à organiser le traitement de l'information dans le réseau. Par exemple, un schéma perceptif tient la forme d'une règle : “Lorsque je détecte une configuration **x**, j'enclenche la procédure **y**”. Au sein d'un DSL où chaque entité possède son **état interne**, ce schéma peut s'exprimer via un **état collectif** stabilisé (plusieurs entités conservent des $\mathbf{s}_i(t)$ cohérents) associé à des liaisons $\omega_{i,j}$ robustes, c'est-à-dire un motif distribué dans le réseau. Alors que l'auto-organisation non supervisée demeure souvent cantonnée à la formation de **clusters** statiques, la présence d'états internes permet l'apparition de boucles d'activité, de séquences ou de boucles réverbérantes, rappelant la résonance neuronale dans les modèles biologiques. Cela dépasse la simple définition de classes, en autorisant la reproduction de schémas temporels ou la mise en œuvre de “routines cognitives” collectives.

Sur le plan de la **biologie neuronale**, on sait que la formation d'assemblées de neurones (règle de Hebb) peut inclure des états récurrents ou “réverbérants” dans lesquels chaque neurone conserve une certaine **trace** d'activité dépendant de son vécu antérieur. La **cognition** naît de la cohérence entre ces assemblées et de leurs interactions réciproques. Dans les **systèmes multi-agents**, si chaque agent possède un état local évoluant selon un script interne, c'est l'**intelligence collective** qui se développe de leur interaction, parfois vue comme un “protocole” distribué dans le SCN. On peut ainsi observer des phénomènes d'**émergence** de structures conceptuelles, de motifs cognitifs, voire de règles partiellement symboliques, dès lors que ces entités-agents échangent et actualisent leurs mémoires internes.

En combinant la plasticité des poids $\omega_{i,j}$ et celle de chaque **état** $\mathbf{s}_i(t)$, le **DSL** franchit le seuil d'une auto-organisation plus **riche**, où se déploient des *schémas cognitifs* allant au-delà du groupement en clusters. L'hypothèse est qu'un tel réseau “vivant”, dont la mémoire réside à la fois dans les **liaisons** (pondérations) et dans les **états** de chaque nœud, peut développer des patterns de perception, de raisonnement ou d'apprentissage plus **complexes** et plus **adaptatifs**. C'est dans ce sens qu'il offre un **cadre** d'études et d'expérimentations susceptible d'éclairer l'émergence de **comportements** cognitifs distribués, rappelant ou imitant certains principes de la cognition humaine ou animale.

Conclusion

Lorsque chaque entité \mathcal{E}_i dispose d'un **état interne**, et que la **synergie** dicte les connexions $\omega_{i,j}$, l'**auto-organisation** au sein du **DSL** s'étend vers une dynamique plus "cognitive", génératrice de schémas complexes ou de motifs récurrents. On ne se borne pas à des **clusters** statiques : le réseau peut véhiculer des séquences, des boucles d'activité, des **routines** aux attributs "cognitifs". Ces schémas apparaissent de la **double** plasticité (pondérations $\omega_{i,j}$ et états internes $\mathbf{s}_i(t)$), illustrant un fonctionnement **distribué**. Les prochains chapitres approfondiront, sous divers angles, la façon dont le **Synergistic Connection Network (SCN)** peut servir de fondement à des comportements apparentés à la **cognition**, nourrissant l'ambition d'un apprentissage non supervisé capable de s'**adapter** et de **raisonner** de manière plus riche.

2.2.5. Exemples Illustratifs

Les principes théoriques exposés jusque-là — concernant la définition de la synergie S , la mise à jour des pondérations $\omega_{i,j}$, la parsimonie, ou encore la possibilité d'un état interne pour chaque entité — se concrétisent dans des **exemples** qui permettent de visualiser la **dynamique** du Deep Synergy Learning (DSL) de façon simple et didactique. La section (2.2.5) présente quelques scénarios illustrant ces mécanismes dans des configurations restreintes, afin de mieux comprendre :

- Comment s'applique la **règle de mise à jour** ($\omega_{ij}(t+1) = \dots$) dans un petit réseau,
- L'effet de la **similarité** (ou distance) sur la croissance ou la décroissance des liens,
- L'émergence de **micro-clusters** même dans des systèmes de taille modeste (3 ou 4 entités).

2.2.5.1. SCN Minimal (3 ou 4 Entités) : Démonstration de la Règle de Mise à Jour

Pour **illustrer** de façon simple la dynamique d'un **Synergistic Connection Network (SCN)** dans le cadre d'un **Deep Synergy Learning (DSL)**, on peut se limiter à un **réseau** très restreint de 3 ou 4 entités, par exemple $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \dots\}$. L'objectif est de **montrer** pas à pas comment les pondérations $\omega_{i,j}$ évoluent conformément à la **formule** générique expliquée en section 2.2.2 :

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta[S(i,j) - \tau \omega_{ij}(t)].$$

Cette expérience sert de **démonstration** élémentaire : malgré la faible dimension du réseau, elle met en évidence l'esprit du **DSL**, où chaque liaison $\omega_{i,j}$ se corrige localement en fonction de la synergie $S(i,j)$, sans qu'aucune supervision globale n'intervienne.

A. Mise en Place d'un Mini-Exemple

Pour clarifier la présentation, on se limite à trois entités $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$. Au temps initial $t = 0$, on initialise toutes les pondérations à une petite valeur égale, par exemple :

$$\omega_{12}(0) = \omega_{23}(0) = \omega_{13}(0) = 0.05.$$

On fixe un **taux d'apprentissage** $\eta = 0.1$ et un **coefficient de décroissance** $\tau = 0.2$. On attribue ensuite des **scores** de synergie $S(i,j)$ purement *ad hoc* :

$$S(1,2) = 0.8, \quad S(1,3) = 0.3, \quad S(2,3) = 0.6.$$

Ici, on suppose \mathcal{E}_1 et \mathcal{E}_2 fortement liées ($S = 0.8$), tandis que \mathcal{E}_1 et \mathcal{E}_3 ne le sont qu'à un niveau modéré ($S = 0.3$). En pratique, ces nombres peuvent provenir d'une **distance**, d'une **similarité** ou d'une **mesure** (cf. section 2.2.1.2), mais on les fixe ici pour la démonstration.

B. Application de la Règle de Mise à Jour

On effectue la **mise à jour** pour chaque itération $t = 0, 1, 2, \dots$. Par exemple, pour la liaison $\omega_{12}(t)$, on a :

$$\omega_{12}(t+1) = \omega_{12}(t) + \eta[0.8 - \tau \omega_{12}(t)].$$

De façon analogue, pour ω_{13} et ω_{23} :

$$\omega_{13}(t+1) = \omega_{13}(t) + \eta[0.3 - \tau \omega_{13}(t)], \quad \omega_{23}(t+1) = \omega_{23}(t) + \eta[0.6 - \tau \omega_{23}(t)].$$

On peut illustrer, par exemple, le premier pas de temps ($t = 0 \rightarrow 1$) pour ω_{12} . En notant $\omega_{12}(0) = 0.05$, on obtient :

$$\begin{aligned} \omega_{12}(1) &= 0.05 + 0.1[0.8 - 0.2 \times 0.05] = 0.05 + 0.1 \times [0.8 - 0.01] = 0.05 + 0.1 \times 0.79 \\ &= 0.05 + 0.079 = 0.129. \end{aligned}$$

On procédera de même pour $\omega_{13}(1)$ et $\omega_{23}(1)$, puis on itérera à chaque pas de temps ($t = 1 \rightarrow 2, 2 \rightarrow 3, \dots$).

C. Tendance vers un Équilibre

L'expérience révèle que $\omega_{12}(t)$ grimpe plus vite (la synergie 0.8 étant la plus élevée) alors que $\omega_{13}(t)$ stagne ou s'élève doucement (puisque la synergie 0.3 est la plus faible). Lorsqu'on laisse l'itération se dérouler sur plusieurs pas, chaque $\omega_{i,j}(t)$ se rapproche d'un **point fixe** $\omega_{i,j}^* \approx S(i,j)/\tau$, conformément à l'analyse de convergence de la section 2.2.2.1. Ainsi,

$$\omega_{12}^* \approx \frac{0.8}{0.2} = 4.0,$$

même s'il convient de réaliser un nombre significatif d'itérations pour s'en approcher, et de tenir compte du facteur η (taux d'apprentissage) qui peut ralentir ou causer de légères oscillations. L'important est d'observer la **logique** sous-jacente : un score $S(i,j)$ élevé provoque un renforcement plus prononcé du poids, tant que la décroissance $\tau \omega_{i,j}(t)$ ne compense pas entièrement le terme $\eta S(i,j)$.

D. Lecture Globale de la Démonstration

Même avec un **mini-exemple** impliquant trois entités $\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3$ et une synergie fixée, on voit clairement comment la **règle** de mise à jour

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta[S(i,j) - \tau \omega_{ij}(t)]$$

entraîne, au fil des itérations, un **rééquilibrage** des pondérations selon les valeurs de $S(i,j)$. Les liaisons ayant une synergie plus grande croissent plus vite, celles ayant une synergie modérée stagnent à un niveau bas, et celles potentiellement nulles convergent vers zéro (en l'absence de renfort).

Dans un réseau plus large (4, 5 entités, voire davantage) et avec des définitions de synergie plus sophistiquées (distance euclidienne, similarité cosinus, co-information), le principe demeure identique : chaque connexion $\omega_{i,j}$ évolue **localement** (cf. sections 2.2.2.2 et 2.2.2.3), et la structure globale (tendance à former des **clusters**) émerge de l'**accumulation** de ces ajustements de bas niveau.

Conclusion

Le recours à un réseau **très réduit** (trois ou quatre entités) permet d'**exposer** pas à pas la **règle** de mise à jour d'un **SCN** en **Deep Synergy Learning**, et d'**observer** la façon dont les pondérations $\omega_{i,j}$ s'orientent vers un **point d'équilibre** proche de $S(i,j)/\tau$. Les sections 2.2.5.2 et 2.2.5.3 prolongeront cette démonstration en introduisant des simulations concrètes où la synergie dérive de distances (euclidienne) ou de similarités (cosinus), et en mettant en évidence la **formation** de **micro-clusters** sur des exemples plus larges.

2.2.5.2. Courtes Simulations (Distance Euclidienne vs. Similarité Cosinus)

Pour **illustrer** de manière concrète la dynamique d'un **Synergistic Connection Network (SCN)** dans le cadre du **Deep Synergy Learning (DSL)**, il est instructif d'entreprendre de petites **simulations** où la **synergie** $S(i,j)$ est définie par des formules accessibles (distance euclidienne inversée, similarité cosinus). L'objectif consiste à examiner, sur un **petit ensemble** (par exemple 4, 5 ou 6 vecteurs \mathbf{x}_i), la manière dont les pondérations $\omega_{i,j}$ se renforcent ou diminuent en fonction de la **proximité** vectorielle.

A. Contexte de la Simulation

Pour favoriser la **lisibilité**, on peut placer chaque entité \mathcal{E}_i dans un **plan** à deux dimensions, de sorte que $\mathbf{x}_i = (x_i, y_i) \in \mathbb{R}^2$. On peut, par exemple, définir des positions :

$$\mathbf{x}_1 = (1.0, 1.2), \quad \mathbf{x}_2 = (1.3, 0.8), \quad \mathbf{x}_3 = (4.2, 5.1), \quad \mathbf{x}_4 = (4.0, 5.4).$$

On constate que \mathbf{x}_1 et \mathbf{x}_2 se trouvent relativement proches, tandis que \mathbf{x}_3 et \mathbf{x}_4 forment un autre groupe de proximité. Les **pondérations** $\omega_{i,j}(0)$ sont initialisées à une petite valeur commune (par exemple 0.05). On fixe ensuite :

$$\eta = 0.1, \quad \tau = 0.2,$$

où η est le **taux d'apprentissage** et τ le **coefficent de décroissance**.

Afin de **comparer**, on fait tourner la **même** règle de mise à jour,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

mais en faisant varier $S(i,j)$ selon deux définitions :

57. **Distance Euclidienne Inversée.** On peut poser

$$S_{\text{dist}}(i, j) = \frac{1}{1 + \| \mathbf{x}_i - \mathbf{x}_j \|}.$$

Ainsi, deux entités plus **proches** (faible distance) reçoivent un score de synergie plus élevé (le dénominateur ($1 + \text{distance}$) est réduit).

58. **Similarité Cosinus.** On peut employer

$$S_{\cos}(i, j) = \max \left\{ 0, \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\| \mathbf{x}_i \| \| \mathbf{x}_j \|} \right\},$$

mettant l'accent sur l'**angle** entre \mathbf{x}_i et \mathbf{x}_j . Deux vecteurs pratiquement parallèles affichent un score élevé, indépendamment de la distance à l'origine.

B. Déroulement de la Simulation

On effectue généralement un certain nombre (10 ou 20) d'**itérations**. À chaque **step** :

59. On **calcule** $S(i, j)$ pour chaque paire (i, j) .
60. On **met à jour** $\omega_{i,j}(t+1)$ grâce à la formule ci-dessus.
61. On peut, si on le souhaite, imposer un **seuil** $\omega_{\min} = 0.01$ (voir section 2.2.3) afin de mettre à zéro les pondérations trop faibles.

Avec la **distance** inversée, $S_{\text{dist}}(i, j)$ se révèle plus grande si $\| \mathbf{x}_i - \mathbf{x}_j \|$ est petite. Si, par exemple, \mathbf{x}_1 et \mathbf{x}_2 sont dans un rayon proche, la synergie $S_{\text{dist}}(1,2)$ grimpe, et $\omega_{12}(t)$ s'accroît logiquement. Dans le cas de la **similarité cosinus** $S_{\cos}(i, j)$, c'est l'**orientation** qui compte : deux vecteurs quasi parallèles obtiennent une forte corrélation angulaire, et leurs pondérations s'élèvent correspondamment.

Après une dizaine d'**itérations**, la majorité des pondérations $\omega_{i,j}$ convergent vers un **point d'équilibre** (s'approchant, selon la section 2.2.2.1, de $S(i, j)/\tau$). Les liens jugés peu synergiques (ex. paires avec $\| \mathbf{x}_i - \mathbf{x}_j \|$ grande ou direction divergente) s'affaiblissent et peuvent tendre vers zéro. L'éventuelle introduction de la **parsimonie** accélère l'obtention d'un **graphe épars**, où n'apparaissent plus que quelques pondérations fortes, signe de la formation de **sous-groupes**.

C. Analyse et Commentaires

Ces **courtes simulations** révèlent que :

- Lorsque $S(i, j)$ exploite la **distance** inversée $1/(1 + \| \mathbf{x}_i - \mathbf{x}_j \|)$, on favorise la **proximité géométrique**. Des vecteurs rapprochés entraînent une forte synergie et, donc, un renforcement marqué des liens $\omega_{i,j}$. On obtient souvent des **regroupements** "spatiaux".
- Lorsque $S(i, j)$ recourt à la **similarité cosinus** ($\max\{0, \mathbf{x}_i \cdot \mathbf{x}_j / (\| \mathbf{x}_i \| \| \mathbf{x}_j \|)\}$), on regarde l'**angle** entre \mathbf{x}_i et \mathbf{x}_j . Deux entités alignées sur une direction analogue reçoivent une synergie élevée, même si elles sont éloignées de l'origine.

Les paramètres η et τ influencent la dynamique. Un η trop grand peut générer des oscillations, un η trop petit ralentit la convergence, un τ trop grand accélère la décroissance, etc. Dans un contexte de **seuil** ω_{\min} , la suppression de liaisons trop faibles clarifie la structure et dévoile rapidement des **sous-groupes** plus nets.

Les simulations “miniatures” montrent la **genèse de micro-clusters** : quelques $\omega_{i,j}$ s’élèvent vite lorsque $S(i,j)$ est forte, alors que d’autres $\omega_{i,j}$ tombent à zéro, scellant la partition en petits groupes. On retrouve là la base de l'**auto-organisation** dans le DSL : partir d’un graphe quasi uniforme, puis voir, itération après itération, certains liens se renforcer et d’autres s’éteindre, révélant des **composantes** ou **communautés** cohérentes.

Conclusion

Même sur un **ensemble** d’entités réduit (4, 5 ou 6 vecteurs), on observe la **dynamique essentielle** du DSL : seules les **paires** dont la synergie $S(i,j)$ dépasse un certain niveau maintiennent des liaisons élevées, tandis que les liaisons à faible synergie décroissent et s’effacent. Les **micro-clusters** qui en résultent correspondent à des affinités (distance, orientation, etc.). Cette expérimentation rudimentaire fournit une **intuition** claire du fonctionnement d’un **SCN** auto-organisé : on part d’une matrice de pondérations presque uniforme, puis la dynamique $\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)]$ engendre l’émergence de liens dominants et la formation de **sous-réseaux** stables.

2.2.5.3. Premières Observations sur la Formation de Micro-Clusters

Les **courtes simulations** présentées en section 2.2.5.2 – qu’elles s’appuient sur la distance euclidienne inversée ou sur la similarité cosinus – font apparaître qu’après plusieurs itérations de la **règle** de mise à jour des pondérations, un **petit réseau** de 3, 4 ou 5 entités tend à voir naître des **groupes** (ou micro-clusters) fortement connectés. Ce phénomène illustre la **tendance** du Deep Synergy Learning (DSL) à structurer le **Synergistic Connection Network (SCN)** en **clusters**, guidés par des **synergies** relativement élevées.

A. Processus d’Émergence des Clusters

Le fait de **définir** une synergie $S(i,j)$ plus grande pour certaines paires $(\mathcal{E}_i, \mathcal{E}_j)$ (par exemple en raison de leur proximité ou de leur corrélation statistique) se traduit, dans la **dynamique** décrite en 2.2.2, par une croissance plus rapide des pondérations $\omega_{i,j}(t)$ correspondantes. Inversement, les paires ne bénéficiant pas d’une synergie élevée voient leurs pondérations décliner ou rester à un niveau bas.

Si, lors des premières itérations, toutes les pondérations $\omega_{i,j}(0)$ se situent autour d’une même petite valeur, la **règle** de renforcement ($S(i,j)$) et de décroissance ($\tau \omega_{i,j}$) introduit progressivement une “**divergence**” dans l’évolution : quelques liaisons $\omega_{i,j}$ montent en flèche, d’autres chutent vers zéro. Cet écart de trajectoires amène, dans un réseau **stable**, un **groupement** des entités ayant $S(i,j)$ plus forte, tandis que les autres connexions s’amoindrissent.

Lorsqu’on ne modifie pas la synergie $S(i,j)$ (c’est-à-dire, si le temps se déroule sans ajout de nouvelles entités ni mise à jour de la mesure de proximité), on constate empiriquement que les pondérations **convergent**. Chaque $\omega_{i,j}(t)$ se rapproche d’un point fixe, souvent $\omega_{i,j}^* \approx S(i,j)/\tau$.

Le **réseau** en fin de phase présente alors quelques couples (ou sous-ensembles) dotés de liens nettement plus forts, mettant en relief leur affinité ou leur synergie.

B. Interprétation des Micro-Clusters

Dans un petit réseau (par exemple 3–5 entités), on peut observer la formation de **micro-clusters**. Un **micro-cluster** est généralement un groupe de 2 ou 3 entités qui se retrouvent reliées par des pondérations élevées, tandis que les connexions vers l’extérieur demeurent faibles ou nulles. Par exemple, $\{\mathcal{E}_1, \mathcal{E}_2\}$ peut nouer un **couple** au poids ω_{12} important, distinct d’un autre binôme $\{\mathcal{E}_3, \mathcal{E}_4\}$, affichant ses propres liens robustes.

Ces **sous-ensembles** correspondent à l’idée que \mathcal{E}_1 et \mathcal{E}_2 partagent une **synergie** suffisante (distance réduite, direction vectorielle similaire, etc.). Le réseau “**signe**” cette analogie en **renforçant** les liaisons internes au groupe et en **atténuant** les connexions vers les entités plus éloignées ou moins similaires. Ce mécanisme constitue une **illustration** de la façon dont, à plus grande échelle, le DSL forme spontanément des **clusters**, chaque entité rejoignant la communauté où elle trouve la **synergie** la plus marquée.

Une représentation géométrique du **réseau** dans un plan (positions \mathbf{x}_i et intensité des liens $\omega_{i,j}$) souligne rapidement ces **groupes** : les liaisons puissantes ressortent, dévoilant la présence de **clusters**. Même sur un petit nombre d’entités, la dynamique de pondérations $\omega_{i,j}(t)$ montre cette capacité du **DSL** à repérer et à **confirmer** les affinités dominantes, phénomène central de l’**auto-organisation**.

C. Prolongements et Évolutions Possibles

Pour un ensemble plus large (par exemple 10, 20 ou 100 entités), le **même** mécanisme opère à plus grande échelle, créant des **clusters** plus volumineux (voire hiérarchiques). L’adoption d’un **seuil** ω_{\min} (voir section 2.2.3) peut accélérer l’obtention d’une **structure** épars et révéler plus franchement les **groupes**. Sur de très petits réseaux (3 ou 4 entités), l’application d’un seuil conduit à des binômes ou trinômes aux liaisons très intenses, fournissant un exemple minimaliste de **partition**.

Si la synergie $\{S(i, j)\}$ varie dans le temps (les entités se déplacent, leur mémoire interne change, etc.), la configuration des micro-clusters peut se **réarranger**. Un binôme initialement serré peut se dissoudre, tandis qu’un nouveau couple se consolide. Le DSL, par sa nature locale, suit ces changements en réajustant $\omega_{i,j}(t)$ selon la nouvelle donne. Même dans un réseau minuscule, on peut ainsi observer des **réorganisations** temporelles et des refontes de clusters en réponse à des variations contextuelles.

Ces **observations** constituent un avant-goût du fonctionnement à échelle plus grande : le **DSL** identifie et **stabilise** des liaisons là où la **synergie** se maintient, dévoilant des **sous-groupes** de haute affinité. Cette logique de “clustering auto-organisé” (ou “micro-clustering” en petit nombre) fonde la démarche du DSL comme algorithme **non supervisé** de regroupement, applicable à des entités diversifiées.

Conclusion

Les simulations à 3–5 entités (section 2.2.5.2) attestent la **capacité** du Deep Synergy Learning à **discriminer** les connexions en fonction de la synergie, menant à la formation de **micro-clusters** :

quelques couples ou triplets voient leurs liaisons se hisser vers des valeurs élevées, tandis que d'autres tendent vers zéro. Cette **émergence** de sous-groupes, visible dès les petits réseaux, inaugure la **clusterisation auto-organisée** qu'on observe à plus grande échelle. Elle constitue la base du **DSL** pour l'**apprentissage non supervisé** sur des entités variées (vecteurs, objets symboliques, distributions, etc.).

QCM

1

Le **Deep Synergy Learning** (DSL) se distingue d'autres approches (type SOM, Hopfield) principalement parce que :

62. Il ne manipule **que** des pondérations fixes sans mise à jour.
63. Il fait intervenir une **fonction de synergie** et une **dynamique** des liaisons $\omega_{i,j}$.
64. Il ne considère aucune forme de plasticité locale.
65. Il utilise uniquement l'information mutuelle comme mesure de synergie.

Réponse attendue : la **2** (il met l'accent sur une fonction de synergie et la dynamique ω).

QCM

2

Dans la formule de mise à jour $\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)]$, le rôle du paramètre τ est de :

66. Forcer toutes les liaisons à croître sans limite.
67. Créer un renforcement multiplicatif basé sur $\omega_{ij}(t)$.
68. Introduire un effet de décroissance qui empêche la divergence des poids.
69. Rendre la convergence plus lente en augmentant la synergie.

Réponse attendue : la **3** (terme de décroissance).

QCM

3

L'ajout d'une **règle de parsimonie** avec un seuil ω_{\min} :

70. Sert à augmenter systématiquement toutes les pondérations sous ω_{\min} .
71. Vise à **supprimer** (ou mettre à zéro) les liaisons trop faibles pour rendre le réseau plus épars.
72. Oblige à fixer ω_{\min} de façon à ce que les pondérations restent toujours négatives.
73. S'applique seulement quand on utilise une distance euclidienne.

Réponse attendue : la **2**.

QCM

4

On parle d'**auto-organisation** parce que :

74. Un superviseur externe dicte explicitement les valeurs de $\omega_{i,j}$.

75. Les poids $\omega_{i,j}$ sont ajustés localement en fonction de la synergie, sans label global d'erreur.

76. Les entités restent fixes et il n'y a aucun apprentissage.

77. Les calculs de mise à jour ne sont valables que pour les SOM.

Réponse attendue : la **2** (mise à jour locale, pas de label externe).

QCM

5

Le choix de la **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$:

78. Peut reposer sur distance, similarité, information mutuelle...

79. Est toujours identique : on utilise la distance euclidienne.

80. Doit obligatoirement être une fonction négative.

81. Ne peut pas changer en cours d'apprentissage.

Réponse attendue : la **1** (il existe divers choix possibles).

QCM

6

Un mécanisme **d'inhibition compétitive** dans la mise à jour signifie que :

82. Chaque lien est renforcé indépendamment, sans influence des autres.

83. Les liens très faibles entraînent la suppression de tous les liens forts autour.

84. Un certain nombre de liaisons sont en « concurrence », limitant la croissance simultanée si l'une devient dominante.

85. On rend η nul pour bloquer l'apprentissage.

Réponse attendue : la **3**.

QCM

7

Dans un **DSL** avec états internes $\mathbf{s}_i(t)$ (par ex. LSTM local) :

86. Seules les pondérations $\omega_{i,j}$ évoluent et les entités sont statiques.

87. Chaque entité \mathcal{E}_i peut modifier son état $\mathbf{s}_i(t)$ en parallèle de la mise à jour des $\omega_{i,j}$.

88. Les états internes sont toujours des scalaires constants.

89. Les états internes ne jouent aucun rôle dans le calcul de la synergie.

Réponse attendue : la **2** (états internes évolutifs).

QCM**8**

Un **avantage** de la parsimonie (seuil ω_{\min}) dans un SCN est :

90. D'augmenter la complexité de calcul.
91. De maintenir plus de liens forts que faibles.
92. De clarifier la structure en coupant les liaisons quasi nulles, accélérant la stabilisation.
93. De forcer la synergie à toujours être nulle.

Réponse attendue : la **3**.

QCM**9**

Lorsqu'une fonction de synergie repose sur la **similarité cosinus** entre deux vecteurs :

94. Elle dépend avant tout de l'angle entre ces vecteurs.
95. Elle est maximale si les deux vecteurs sont orthogonaux.
96. Elle favorise les paires de vecteurs les plus distants.
97. Elle ne s'applique pas en dimension supérieure à 2.

Réponse attendue : la **1** (le cosinus dépend de l'angle).

QCM**10**

Dans les **petits exemples** de SCN (3 à 5 entités) illustrés dans le texte, on constate souvent que :

98. Tous les liens finissent par la même valeur finale.
99. Les pondérations s'organisent en **micro-clusters** (certains liens se renforcent nettement, d'autres s'affaiblissent).
100. Les poids $\omega_{i,j}$ deviennent tous nuls après la première itération.
101. Aucune clusterisation n'apparaît, quels que soient η et τ .

Réponse attendue : la **2** (formation de micro-clusters).

II. Exercices et problèmes

Chacun invite à **manipuler** ou **discuter** des formules, des propriétés, sans entrer dans un calcul numérique poussé.

Exercice 1

Règle de mise à jour — convergence qualitative

Soit la règle

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)].$$

102. Montrez formellement, à partir de l'écriture $\omega_{ij}(t+1) = (1 - \eta \tau) \omega_{ij}(t) + \eta S(i,j)$, comment on obtient le point fixe $\omega_{ij}^* = \frac{S(i,j)}{\tau}$.

103. Discuter brièvement (sans nombres) pourquoi cette valeur est un **attracteur** stable (indice : considérer le signe de la dérivée locale).

(*But : analyser la convergence qualitative sans faire de calcul numérique.*)

Exercice 2

Comparaison additive vs. Multiplicative

Considérez deux variantes de mise à jour :

104. $\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)]$ (additif)

105. $\omega_{ij}(t+1) = \omega_{ij}(t) [1 + \alpha S(i,j)] - \beta \omega_{ij}(t)$ (multiplicatif).

- Expliquez, sans calcul numérique, en quoi la **dépendance** à la valeur courante $\omega_{ij}(t)$ diffère entre ces deux versions.
- Qu'implique cette différence pour la **croissance** d'un lien déjà fort ?

Exercice 3

Parsimonie et filtrage

On ajoute une règle : toute pondération ω_{ij} restant strictement en dessous de ω_{\min} après chaque itération est mise à zéro.

106. Expliquez (sans chiffres) en quoi ce “cut-off” introduit un **filtrage** topologique sur le graphe.

107. Proposez un scénario (idéalement abstrait) où un lien commence faible et ne peut jamais “revenir” s'il est coupé trop tôt.

Exercice 4

États internes et mise à jour

On suppose que chaque entité \mathcal{E}_i a un état local $\mathbf{s}_i(t)$ mis à jour selon une fonction

$$\mathbf{s}_i(t+1) = \mathbf{F}(\mathbf{s}_i(t), \{\omega_{i,j}\}, \{\mathbf{s}_j(t)\}).$$

108. Expliquez comment la synergie $S(i,j)$ pourrait dépendre (partiellement) de $\mathbf{s}_i(t)$ et $\mathbf{s}_j(t)$.
109. Sans calcul, décrivez un mécanisme local (type “règle du voisinage”) pour \mathbf{F} qui tienne compte des états internes des entités voisines.

Exercice 5

Inhibition compétitive

On introduit un terme $\gamma \sum_k \omega_{ik}(t)$ dans la formule de mise à jour pour la liaison ω_{ij} .

110. Écrivez, symboliquement, à quoi pourrait ressembler la nouvelle équation.
111. Montrez, par une simple analyse de signe (sans chiffres), comment cette somme $\sum_k \omega_{ik}$ peut “freiner” la croissance de ω_{ij} si d’autres liens ω_{ik} sont déjà importants.

Exercice 6

Fonction de synergie composée

Supposons que la synergie se définisse comme

$$S(i,j) = \alpha \sigma_{\cos}(i,j) + (1 - \alpha) \sigma_{\text{dist}}(i,j),$$

où σ_{\cos} est une similarité cosinus et σ_{dist} est une distance inversée, et $\alpha \in [0,1]$.

112. Discutez comment la pondération α influe sur l’orientation du critère de synergie (plus focalisé sur l’angle ou la proximité absolue).
113. Proposez une justification (abstraite) montrant pourquoi cette approche peut convenir à un système multimodal (par ex. vecteurs normalisés vs. positions géométriques).

Exercice 7

Collage de similarités pour des espaces hétérogènes

On a des entités réparties en deux espaces \mathcal{X}_1 et \mathcal{X}_2 . On définit σ_1 sur \mathcal{X}_1 et σ_2 sur \mathcal{X}_2 , et on prolonge en “collant” à zéro pour les paires mélangeant \mathcal{X}_1 et \mathcal{X}_2 .

- Expliquez, sans chiffres, pourquoi ce collage peut **fragmenter** le réseau en deux composantes s'il n'y a aucun "pont" (valeur > 0) entre espaces.
- Quelle serait une stratégie pour introduire un pont $\alpha_{1,2} > 0$ (constante) assurant une légère synergie entre \mathcal{X}_1 et \mathcal{X}_2 ?

Exercice 8

Analyse de stabilité : variation autour du point fixe

Considérez un lien unique $\omega_{ij}(t)$ autour du point fixe $\omega^* = S(i,j)/\tau$. On écrit $\omega_{ij}(t) = \omega^* + \varepsilon(t)$.

114. En remplaçant dans la formule de mise à jour, montrez symboliquement que $\varepsilon(t+1) \approx (1 - \eta \tau) \varepsilon(t)$ (sans faire de développement numérique).
115. Concluez sur le fait que $|1 - \eta \tau| < 1$ implique une **décroissance** de $\varepsilon(t)$.

Exercice 9

Distances vs. Similarités

Soit un ensemble de vecteurs $\{\mathbf{x}_i\}$. On compare deux fonctions de synergie :

- $S_d(i,j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|)$.
- $S_c(i,j) = \max\{0, \mathbf{x}_i \cdot \mathbf{x}_j / (\|\mathbf{x}_i\| \|\mathbf{x}_j\|)\}$.

Expliquez, par un raisonnement purement qualitatif (sans calcul de normes), les différences de **regroupement** que l'on peut attendre pour un ensemble de 4 points disposés différemment (quelques exemples de configurations à imaginer).

Exercice 10

Petits réseaux et micro-clusters

Sur 4 entités $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$, on a initialement $\omega_{i,j}(0) \approx \varepsilon$ (petit).

116. Exposez un raisonnement (sans valeurs numériques) pour prédire quel(s) cluster(s) se forment si $S(1,2)$ et $S(3,4)$ sont très grands, tandis que les autres $S(i,j)$ sont moyens ou faibles.
117. Précisez l'effet d'un seuil ω_{\min} sur la séparation claire entre ces deux binômes (1,2) et (3,4).

Remarques finales

- Ces exercices visent à **manipuler** les formules et raisonnements **qualitatifs** (stabilité, attracteurs, influence d'un paramètre) sans entrer dans le calcul d'exemples chiffrés.
- Les QCM mettent en lumière l'**essence** du DSL : synergie, mise à jour locale, clusters émergents, parsimonie.
- Les exercices mathématiques explorent la **dynamique** des équations, l'effet des **paramètres** (inhibition, parsimonie, etc.) et la **flexibilité** de la fonction de synergie.

Voici **5 “grands problèmes”**, chacun décomposé en **5 sous-questions** à caractère **purement mathématique** (non numériques). Ils portent sur la **logique formelle** du Deep Synergy Learning (DSL) : définition et propriétés de la fonction de synergie, étude de la règle de mise à jour, analyse de stabilité, rôle de la parsimonie, etc. Les énoncés évitent tout calcul chiffré et se concentrent sur la manipulation d'équations, de propriétés ou de raisonnements symboliques.

Problème 1 : Collage de Fonctions de Synergie

On considère deux espaces \mathcal{X}_1 et \mathcal{X}_2 , avec des fonctions de similarité σ_1 et σ_2 définies respectivement sur $\mathcal{X}_1 \times \mathcal{X}_1$ et $\mathcal{X}_2 \times \mathcal{X}_2$. On construit l'union disjointe $\mathcal{U} = \sqcup_{m \in \{1,2\}} (\mathcal{X}_m \times \{m\})$ et on “colle” σ_1, σ_2 en une unique fonction

$$\Sigma(u, v) = \begin{cases} \sigma_1(x, y), & \text{si } u = (x, 1), v = (y, 1), \\ \sigma_2(x, y), & \text{si } u = (x, 2), v = (y, 2), \\ c_{1,2}, & \text{sinon.} \end{cases}$$

On suppose $c_{1,2} \geq 0$ constant, pour introduire une « passerelle » entre \mathcal{X}_1 et \mathcal{X}_2 .

- (a) Montrez (sans calcul numérique) que si σ_1, σ_2 sont symétriques et non négatives (p. ex. de vraies similarités), alors Σ l'est aussi sur $\mathcal{U} \times \mathcal{U}$.
- (b) Décrivez pourquoi prendre $c_{1,2} = 0$ risque de « fragmenter » le graphe en deux composantes.
- (c) Montrez, sous une hypothèse de continuité (ou uniform continuity) de σ_1, σ_2 , que Σ peut être construite de manière à être continue sur $\mathcal{U} \times \mathcal{U}$.
- (d) Discutez en quoi cette construction sert de **fonction de synergie** globale dans un DSL où coexistent deux types différents d'entités (\mathcal{X}_1 vs. \mathcal{X}_2).
- (e) Proposez un **argument** (purement formel) selon lequel, si σ_1, σ_2 sont bornées par 1, alors Σ l'est également, prouvant que Σ prend ses valeurs dans $[0,1]$.

Problème 2 : Analyse de la Règle de Mise à Jour et Stabilisation

Soit la règle de mise à jour pour la liaison $\omega_{ij}(t)$:

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta[S(i,j) - \tau\omega_{ij}(t)],$$

où $\eta > 0$, $\tau > 0$ et $S(i,j) \geq 0$ est la synergie.

(a) Déduisez la forme équivalente :

$$\omega_{ij}(t+1) = (1 - \eta\tau)\omega_{ij}(t) + \eta S(i,j).$$

Expliquez, sans chiffres, pourquoi c'est simplement un « mélange linéaire » entre l'ancienne valeur et le terme constant $\eta S(i,j)$.

(b) Montrez que le **point fixe** ω_{ij}^* satisfait $\omega_{ij}^* = S(i,j)/\tau$.

(c) Pour $\omega_{ij}(t)$ proche de ce point fixe, écrivez l'erreur $\varepsilon(t) = \omega_{ij}(t) - \omega_{ij}^*$, et montrez (symboliquement) que ε se contracte d'environ un facteur $|1 - \eta\tau|$ à chaque itération.

(d) Concluez sur la **stabilisation** de $\omega_{ij}(t)$ si $|1 - \eta\tau| < 1$.

(e) Indiquez comment cette convergence locale peut se généraliser à un ensemble de liens $\{\omega_{ij}\}$, en l'absence de couplage inhibiteur. (Vous pouvez simplement décrire le principe de superposition, sans rentrer dans un calcul complet.)

Problème 3 : Variantes de la Mise à Jour — Inhibition et Multiplicatif

On envisage deux mécanismes supplémentaires dans la mise à jour $\omega_{ij}(t)$.

Partie A : Inhibition compétitive

On ajoute un terme $\gamma \sum_{k \neq j} \omega_{ik}(t)$ pour freiner la croissance de ω_{ij} . Par exemple :

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \left[S(i,j) - \tau \omega_{ij}(t) - \gamma \sum_{k \neq j} \omega_{ik}(t) \right].$$

118. Écrivez l'équation sous forme factorisée pour faire apparaître $(1 - \eta\tau)\omega_{ij}(t)$ et un terme de type $-\eta\gamma \sum_{k \neq j} \omega_{ik}(t)$.

119. Discutez (sans valeur numérique) comment le paramètre $\gamma > 0$ crée de la **compétition** entre les liaisons $\omega_{i,j}$ et $\omega_{i,k}$.

120. Pourquoi cette concurrence peut-elle contribuer à obtenir un réseau « plus clairsemé » ?

Partie B : Schéma multiplicatif

On propose une mise à jour du type :

$$\omega_{ij}(t+1) = \omega_{ij}(t) \exp(\alpha [S(i,j) - \tau \omega_{ij}(t)]).$$

121. Comparez l'impact sur un ω_{ij} déjà fort, par rapport à la formule additive.
122. Quelle forme de **non-linéarité** cette version introduit-elle dans la croissance/décroissance de ω_{ij} ?

(Les deux parties A et B forment un seul grand problème, articulé en 5 questions.)

Problème 4 : États Internes et Couplage dans le Calcul de la Synergie

On considère qu'en plus des pondérations $\omega_{i,j}$, chaque entité \mathcal{E}_i dispose d'un **état local** $\mathbf{s}_i(t) \in \mathcal{S}$. La fonction de synergie dépend partiellement de ces états :

$$S(i,j) = F(\mathbf{s}_i(t), \mathbf{s}_j(t), \theta),$$

où θ symbolise d'autres paramètres ou une base de similarité globale.

- (a)** Expliquez, sans calcul, comment on peut généraliser le DSL pour que $\mathbf{s}_i(t)$ évolue selon une équation locale :

$$\mathbf{s}_i(t+1) = \mathbf{G}_i(\mathbf{s}_i(t), \omega_{i,\cdot}(t), \{\mathbf{s}_j(t)\}_{j \neq i}).$$

- (b)** Donnez un exemple abstrait (symbolique) où $F(\mathbf{s}_i, \mathbf{s}_j, \theta)$ mesure la **distance** entre \mathbf{s}_i et \mathbf{s}_j , ou bien leur **produit scalaire** (sans entrer dans des nombres).

- (c)** Comment le fait que \mathbf{s}_i et \mathbf{s}_j évoluent à chaque itération influence-t-il la valeur de $\omega_{ij}(t+1)$? (Décrivez le mécanisme, pas un calcul.)

- (d)** Supposez qu'on introduise un petit « oublin** » $\beta > 0$ dans \mathbf{s}_i , pour l'empêcher de croître indéfiniment. Expliquez (symboliquement) comment \mathbf{G}_i pourrait inclure ce terme.

- (e)** Montrez l'intérêt (toujours symboliquement) : si une entité \mathcal{E}_i a un état local la rendant « compatible » avec \mathcal{E}_j , alors la synergie $S(i,j)$ favorise ω_{ij} . En retour, l'évolution de \mathbf{s}_i peut se synchroniser avec celle de \mathbf{s}_j .

Problème 5 : Seuil de Parsimonie et Formation de Micro-Clusters

On introduit une règle de parsimonie : après chaque itération, on « coupe » (met à zéro) toute liaison ω_{ij} qui reste sous un seuil $\omega_{\min} > 0$.

- (a)** Écrivez formellement la règle de coupure :

$$\omega_{ij}(t+1) = \begin{cases} \omega_{ij}(t+1), & \text{si } \omega_{ij}(t+1) \geq \omega_{\min}, \\ 0, & \text{sinon.} \end{cases}$$

Justifiez le sens **topologique** (graphique) de cette coupure.

- (b) Expliquez pourquoi, s'il existe 2 entités $\mathcal{E}_p, \mathcal{E}_q$ dont la synergie est constamment faible, leur lien $\omega_{p,q}(t)$ sera rapidement amené à tomber sous ω_{\min} et se retrouver définitivement coupé.
- (c) Montrez, en revanche, qu'un lien $\omega_{r,s}$ dont la synergie est modérément élevée peut franchir ω_{\min} en quelques itérations (sans chiffrer), puis se maintenir actif.
- (d) À l'échelle globale, comment ce phénomène aboutit-il à des **micro-clusters** (petits groupes d'entités fortement reliées entre elles) ?
- (e) Discutez un risque d'“exclusion précoce” : un lien qui démarre trop faible peut être coupé avant d'avoir atteint un niveau satisfaisant. Quel mécanisme correctif (ou adaptatif) pourrait éviter ce problème ?

2.3. Hypothèses de Stabilité et Émergence de Clusters

Lorsque l'on considère un **Synergistic Connection Network (SCN)** soumis aux règles de mise à jour décrites (section 2.2), une question naturelle se pose : **le réseau converge-t-il** ? Et si oui, vers quelles formes de topologies ou de structures (clusters, attracteurs) ? La section (2.3) aborde ces problématiques à travers un point de vue “système dynamique” (2.3.1), la possibilité d'attracteurs multiples ou d'oscillations (2.3.2), l'analyse de la formation de clusters (2.3.3) et l'influence du bruit (2.3.4). Nous nous intéresserons également (2.3.5) aux hypothèses fortes et aux limites non résolues.

2.3.1. Point Fixe, Attracteurs : Définitions et Existence

L'évolution temporelle du réseau $\Omega(t)$ — où $\Omega(t)$ désigne l'ensemble des pondérations $\{\omega_{i,j}(t)\}$ (et éventuellement d'autres variables d'état) — peut souvent être examinée à travers la **théorie des systèmes dynamiques**. La première sous-section (2.3.1.1) formalise cette perspective ; les suivantes traiteront de l'existence d'un **point fixe** (2.3.1.2) et d'une **analyse locale** de stabilité (2.3.1.3).

2.3.1.1. Approche « système dynamique »

Dans une perspective théorique visant à décrire de manière globale l'évolution des connexions au sein d'un réseau d'entités interconnectées, il est particulièrement instructif d'adopter l'optique d'un système dynamique discret. L'idée centrale consiste à regrouper l'ensemble des pondérations, notées $\omega_{i,j}$ (éventuellement complétées par des variables représentant des états internes), dans une configuration globale qui sera représentée par le vecteur d'état

$$\Omega(t) = (\omega_{1,2}(t), \omega_{1,3}(t), \dots, \omega_{n-1,n}(t), \dots),$$

où t désigne l'instant discret (ou l'itération) et où, pour un réseau comportant n entités, le nombre total de pondérations peut atteindre $n(n - 1)$ (dans le cas orienté) ou $\frac{n(n-1)}{2}$ (dans le cas non orienté).

L'évolution de l'état du réseau dans le cadre du Deep Synergy Learning (DSL) se traduit par une règle de mise à jour collective, que l'on peut écrire sous la forme d'une équation fonctionnelle :

$$\Omega(t + 1) = F(\Omega(t)).$$

Ici, l'opérateur F représente l'ensemble des règles locales de mise à jour, telles que celles décrites dans la section 2.2.2, où chaque poids évolue selon l'équation

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

La fonction $S(i,j)$ — qui quantifie la synergie ou l'utilité mutuelle entre les entités \mathcal{E}_i et \mathcal{E}_j — intervient comme un signal de renforcement, tandis que le terme de décroissance proportionnel à $\tau \omega_{i,j}(t)$ permet de réguler la croissance des poids. Lorsque l'on regroupe l'ensemble de ces mises

à jour dans un unique opérateur F , on obtient ainsi une formulation globale de la dynamique du réseau.

A. Structure de l'espace d'états

Dans le formalisme usuel de la théorie des systèmes dynamiques discrets, l'évolution se décrit par

$$\Omega(t+1) = F(\Omega(t)), \quad \Omega(t) \in \mathcal{X} \subseteq \mathbb{R}^D,$$

où D représente la dimension totale du vecteur d'états (qui, dans notre contexte, est le nombre de pondérations considérées). Plusieurs propriétés de cet espace d'états sont essentielles pour l'analyse :

- **Continuité de F :** Pour appliquer des résultats classiques tels que le théorème du point fixe de Banach, il est nécessaire que F soit continue, voire contractante, dans une norme appropriée.
- **Compacité et Bornitude de \mathcal{X} :** Dans de nombreux modèles, on impose des contraintes (par exemple, $\omega_{i,j} \geq 0$ ou des saturations maximales) afin que l'ensemble des configurations possibles \mathcal{X} soit borné. Ces conditions favorisent l'existence d'attracteurs et empêchent la divergence de la norme $\|\Omega(t)\|$.

B. Analyse de la dynamique locale et globale

Pour illustrer la dynamique d'un lien individuel, considérons la règle linéaire-additive

$$\omega_{i,j}(t+1) = (1 - \eta \tau) \omega_{i,j}(t) + \eta S(i,j).$$

Si l'on suppose que les synergies $S(i,j)$ sont elles-mêmes bornées dans un intervalle $[0, S_{\max}]$, il est immédiat de constater que la dynamique unidimensionnelle de chaque $\omega_{i,j}$ tend vers l'équilibre

$$\omega_{i,j}^* \approx \frac{S(i,j)}{\tau},$$

à condition que les paramètres η et τ soient choisis de manière appropriée (c'est-à-dire suffisamment faibles pour garantir une convergence stable et éviter des oscillations indésirables).

Au niveau collectif, l'opérateur F regroupe l'ensemble de ces équations de mise à jour. Pour étudier la stabilité du système dynamique global, on s'intéresse notamment aux questions suivantes :

123. **Contraction ou Monotonie de F :** Dans quelle mesure existe-t-il une norme (ou une métrique) pour laquelle l'opérateur F contracte les distances entre configurations ? Un opérateur contractant garantira, par le théorème du point fixe, l'existence d'un unique point fixe attracteur, vers lequel converge toute trajectoire.
124. **Existence et Stabilité d'un Point Fixe :** Sous quelles conditions sur les paramètres η , τ et la fonction de synergie S peut-on montrer formellement l'existence d'un équilibre stable Ω^* tel que $F(\Omega^*) = \Omega^*$?
125. **Impact des Termes Non Linéaires :** L'introduction de mécanismes supplémentaires, tels que l'inhibition compétitive ou une mise à jour multiplicative, rend la

carte F plus complexe, car la mise à jour d'un poids peut alors dépendre non seulement de la synergie entre deux entités, mais aussi des interactions avec d'autres connexions (par exemple, via des sommes telles que $\sum_{k \neq j} \omega_{i,k}$). Néanmoins, même dans ce cas, l'analyse se situe dans le cadre général des systèmes dynamiques discrets, et l'on peut rechercher des attracteurs, des cycles ou des comportements plus complexes (éventuellement chaotiques).

C. Implications pour l'auto-organisation du SCN

La formulation $\Omega(t+1) = F(\Omega(t))$ permet d'aborder l'auto-organisation du Synergistic Connection Network (SCN) sous l'angle des systèmes dynamiques. En effet, la convergence des pondérations vers des points fixes ou des attracteurs locaux traduit la formation d'une structure stable au sein du réseau. Ces attracteurs correspondent, dans une interprétation globale, à des configurations dans lesquelles les connexions les plus pertinentes sont renforcées, tandis que les liens moins utiles s'affaiblissent voire disparaissent. Ainsi, l'analyse de la stabilité et de la convergence de F constitue un outil puissant pour comprendre l'émergence de clusters durables et la robustesse de l'architecture auto-organisée.

Dans des variantes plus avancées du DSL, l'introduction de termes d'inhibition compétitive ou de mécanismes multiplicatifs ne fait qu'enrichir la dynamique, tout en restant dans le cadre itératif général. L'étude des propriétés de l'opérateur F – par exemple, via la linéarisation autour d'un point fixe et l'analyse de sa jacobienne – permettra d'identifier des conditions précises garantissant l'existence d'un équilibre stable, ou au contraire révélant la présence de cycles limités ou de comportements non linéaires plus complexes.

Conclusion

La vision « système dynamique » du SCN s'exprime par la relation

$$\Omega(t+1) = F(\Omega(t)),$$

où $\Omega(t)$ regroupe l'ensemble des pondérations (ainsi que, éventuellement, d'autres variables d'état). Cette formulation offre une passerelle vers l'application des outils de la théorie des systèmes dynamiques discrets pour analyser la stabilité, l'existence d'attracteurs et les potentiels phénomènes non linéaires (oscillations, bifurcations, voire chaos) qui pourraient émerger dans le cadre du Deep Synergy Learning. Les développements ultérieurs (voir sections 2.3.1.2 et 2.3.1.3) approfondiront l'analyse formelle de l'existence d'un point fixe Ω^* et de sa stabilité locale, éclairant ainsi la manière dont l'auto-organisation conduit à l'émergence de structures de clusters robustes dans le réseau.

2.3.1.2. Existence d'un point fixe Ω^* : conditions théoriques, rôle de τ et η

Dans le cadre de l'approche système dynamique, l'idée centrale consiste à décrire l'évolution globale de la configuration des liaisons d'un réseau en regroupant l'ensemble des pondérations, notées $\omega_{i,j}(t)$, dans un vecteur d'état $\Omega(t)$. Ce vecteur, qui peut être considéré comme un élément de l'espace $\mathcal{X} \subset \mathbb{R}^D$ (où la dimension D correspond au nombre total de connexions, par exemple $n(n-1)$ pour un réseau orienté ou $\frac{n(n-1)}{2}$ dans le cas non orienté), évolue selon une dynamique discrète que l'on peut exprimer par l'équation fonctionnelle

$$\boldsymbol{\Omega}(t+1) = F(\boldsymbol{\Omega}(t)).$$

Ici, l'opérateur F regroupe de manière globale l'ensemble des règles de mise à jour locales qui, dans le DSL (Deep Synergy Learning), s'expriment typiquement par la relation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ représente la synergie ou l'utilité mutuelle entre les entités \mathcal{E}_i et \mathcal{E}_j , $\eta > 0$ est le taux d'apprentissage qui détermine la vitesse d'évolution, et $\tau > 0$ est un paramètre de décroissance agissant comme régulateur afin d'éviter une croissance indéfinie des poids.

Un point fixe $\boldsymbol{\Omega}^*$ de la dynamique est défini par la condition d'invariance

$$\boldsymbol{\Omega}^* = F(\boldsymbol{\Omega}^*).$$

La recherche d'un tel point fixe revient à examiner les conditions sous lesquelles la dynamique – issue de la mise à jour locale de chaque connexion – converge vers une configuration stable, c'est-à-dire une configuration pour laquelle les poids ne varient plus au fil des itérations.

A. Analyse dans le Cas Additif Simple

Pour mieux illustrer cette idée, considérons d'abord le cas de la mise à jour additive élémentaire. Pour une connexion donnée, la règle de mise à jour s'écrit

$$\omega_{i,j}(t+1) = (1 - \eta \tau) \omega_{i,j}(t) + \eta S(i,j),$$

en supposant que la valeur $S(i,j)$ soit constante (ou quasi-stationnaire) dans le temps. Rechercher un équilibre consiste alors à poser $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$. La relation d'équilibre devient immédiatement

$$\omega_{i,j}^* = (1 - \eta \tau) \omega_{i,j}^* + \eta S(i,j).$$

En isolant $\omega_{i,j}^*$ dans cette équation, nous obtenons

$$\eta S(i,j) = \eta \tau \omega_{i,j}^* \quad \Rightarrow \quad \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Ce résultat, qui est immédiatement visible dans le cas unidimensionnel, démontre que chaque poids converge vers un équilibre déterminé par le rapport de la synergie au coefficient de décroissance τ . Ainsi, le paramètre τ joue un rôle fondamental en modulant l'amplitude finale : si τ est élevé, l'équilibre s'établit à des valeurs relativement faibles, tandis qu'un τ plus faible permet aux poids d'atteindre des niveaux plus importants. Par ailleurs, le paramètre η conditionne la vitesse de convergence ; un η trop élevé risque de provoquer des oscillations ou des divergences, tandis qu'un η trop faible ralentit le processus de stabilisation.

B. Généralisation aux Systèmes Couplés

Dans un réseau complet (ou un Synergistic Connection Network – SCN), la synergie $S(i,j)$ peut dépendre non seulement de la relation entre les deux entités considérées, mais également des

interactions avec d'autres connexions ou états internes. Dans ce cas, l'opérateur de mise à jour s'écrit de façon plus générale

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j; \{\omega_{k,\ell}(t)\}, \{s_k(t)\}, \dots) - \tau \omega_{i,j}(t)].$$

Le point fixe global Ω^* satisfait alors l'ensemble des équations

$$S(i,j; \{\omega_{k,\ell}^*\}) = \tau \omega_{i,j}^*, \quad \forall (i,j),$$

ce qui conduit à un système non linéaire d'équations. Pour établir l'existence d'un tel point fixe, on peut s'appuyer sur des arguments classiques de théorie des points fixes tels que le théorème de Brouwer ou celui de Schauder. En effet, si l'opérateur F est continu et que l'ensemble des configurations possibles \mathcal{X} (souvent défini par des contraintes telles que $\omega_{i,j} \in [0, \omega_{\max}]$) est compact et convexe, alors il existe au moins un point fixe.

L'unicité et la stabilité de ce point fixe dépendent de propriétés supplémentaires de F – en particulier, si F est contractant dans une norme appropriée, c'est-à-dire si l'on peut trouver un $\kappa < 1$ tel que

$$\| F(\mathbf{x}) - F(\mathbf{y}) \| \leq \kappa \| \mathbf{x} - \mathbf{y} \|$$

pour tout $\mathbf{x}, \mathbf{y} \in \mathcal{X}$. Dans ce cas, non seulement l'existence mais également l'unicité du point fixe est garantie, et toutes les trajectoires convergeront vers celui-ci.

C. Rôle des Paramètres η et τ

Dans la dynamique proposée, le paramètre η agit comme le taux d'apprentissage. Il détermine la rapidité avec laquelle la configuration $\Omega(t)$ s'ajuste en réponse au signal de renforcement fourni par $S(i,j)$ et à l'effet de décroissance proportionnel à $\tau \omega_{i,j}(t)$. Un choix judicieux de η est indispensable pour éviter des comportements oscillatoires ou divergents tout en permettant une convergence suffisamment rapide.

Le coefficient τ quant à lui intervient comme un régulateur qui « freine » la croissance des poids. En effet, dans la formule additive, il apparaît sous la forme d'un facteur multiplicatif qui, en équilibrant l'influence de la synergie $S(i,j)$, fixe l'amplitude finale de chaque pondération par le rapport $S(i,j)/\tau$. Une valeur élevée de τ force ainsi les poids à converger vers des valeurs plus faibles, ce qui peut être souhaitable pour maintenir un réseau stable et éviter la domination excessive d'un lien sur les autres.

D. Conclusion

En somme, la formulation du DSL sous la forme d'un système dynamique discret

$$\Omega(t+1) = F(\Omega(t))$$

permet d'appréhender la convergence et la stabilisation des liaisons par l'intermédiaire de la recherche d'un point fixe Ω^* . Dans le cas le plus simple d'une mise à jour additive, on démontre que chaque pondération tend vers l'équilibre

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Dans des systèmes plus complexes, où S dépend de l'ensemble des pondérations et éventuellement d'états internes supplémentaires, l'existence d'un point fixe peut être garantie par les théorèmes classiques de la théorie des points fixes, à condition que l'opérateur F soit continu et que l'ensemble des états soit compact. Les paramètres η et τ jouent alors un rôle déterminant : η régule la vitesse de convergence tandis que τ module l'amplitude finale des poids, assurant ainsi que l'équilibre atteint est à la fois stable et cohérent avec la logique de renforcement/décroissance inhérente au DSL.

Cette analyse théorique jette les bases d'une étude approfondie de la stabilité locale – par linéarisation et analyse de la jacobienne de F – et ouvre la voie à la compréhension des phénomènes d'auto-organisation, tels que la formation de clusters durables au sein du réseau. Les développements ultérieurs s'attachent à préciser les conditions techniques garantissant non seulement l'existence, mais aussi l'unicité et la robustesse de l'attracteur Ω^* .

2.3.1.3. Analyse locale (linéarisation, Jacobienne) pour évaluer la stabilité

Dans l'étude des systèmes dynamiques discrets appliqués aux réseaux d'auto-organisation du Deep Synergy Learning (DSL), il est essentiel de s'intéresser à la stabilité des points fixes de la dynamique globale. En d'autres termes, une fois qu'une configuration Ω^* des poids (et éventuellement d'autres états internes) vérifie

$$\Omega^* = F(\Omega^*),$$

la question cruciale est de déterminer si, en cas de petite perturbation autour de Ω^* , le système tend à revenir vers cet équilibre ou s'en éloigne. Pour répondre à cette interrogation, on utilise la technique de linéarisation locale, qui s'appuie sur le calcul de la matrice Jacobienne de l'opérateur F évaluée en Ω^* .

A. Linéarisation locale autour d'un point fixe

Soit un système dynamique discret défini par

$$\Omega(t+1) = F(\Omega(t)), \quad \Omega(t) \in \mathbb{R}^D,$$

où D représente la dimension totale de l'espace d'états, par exemple le nombre total de pondérations $\omega_{i,j}$ (éventuellement augmenté par d'autres variables telles que les états internes s_i). Un point fixe Ω^* est tel que

$$\Omega^* = F(\Omega^*).$$

Pour étudier la stabilité locale de ce point fixe, on introduit une perturbation infinitésimale $\delta\Omega(t)$ et on écrit

$$\Omega(t) = \Omega^* + \delta\Omega(t).$$

En supposant que $\delta\Omega(t)$ reste suffisamment petit, il est alors possible de linéariser la fonction F au voisinage de Ω^* en négligeant les termes d'ordre supérieur. On obtient ainsi :

$$\boldsymbol{\Omega}(t+1) = F(\boldsymbol{\Omega}^* + \delta\boldsymbol{\Omega}(t)) \approx F(\boldsymbol{\Omega}^*) + DF(\boldsymbol{\Omega}^*) \delta\boldsymbol{\Omega}(t),$$

où $DF(\boldsymbol{\Omega}^*)$ désigne la matrice Jacobienne (de dimension $D \times D$) de F évaluée en $\boldsymbol{\Omega}^*$. Puisque $F(\boldsymbol{\Omega}^*) = \boldsymbol{\Omega}^*$, on peut réécrire :

$$\boldsymbol{\Omega}(t+1) \approx \boldsymbol{\Omega}^* + DF(\boldsymbol{\Omega}^*) \delta\boldsymbol{\Omega}(t).$$

En soustrayant $\boldsymbol{\Omega}^*$ des deux côtés, la dynamique des perturbations est donnée par :

$$\delta\boldsymbol{\Omega}(t+1) = DF(\boldsymbol{\Omega}^*) \delta\boldsymbol{\Omega}(t).$$

La stabilité locale du point fixe $\boldsymbol{\Omega}^*$ est alors entièrement caractérisée par les valeurs propres de la matrice Jacobienne $DF(\boldsymbol{\Omega}^*)$. En effet, pour un système discret, le point fixe est localement stable (c'est-à-dire, asymptotiquement stable) si et seulement si toutes les valeurs propres λ de $DF(\boldsymbol{\Omega}^*)$ satisfont la condition

$$|\lambda| < 1.$$

Si au moins une valeur propre vérifie $|\lambda| > 1$, toute petite perturbation dans la direction associée se verra amplifiée, rendant ainsi le point fixe instable. En présence de valeurs propres de module exactement égal à 1, la stabilité locale devient marginale et des comportements plus complexes (cycles, bifurcations) peuvent apparaître.

B. Jacobienne dans le cas “additif” simple

Pour illustrer concrètement le procédé, considérons le modèle additif simple de mise à jour présenté en section 2.2.2.1 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

que l'on peut réécrire sous la forme

$$\omega_{i,j}(t+1) = (1 - \eta \tau) \omega_{i,j}(t) + \eta S(i,j).$$

Si l'on suppose que la fonction de synergie $S(i,j)$ est constante (ou varie très lentement) et qu'elle ne dépend pas directement des autres poids, la dynamique de chaque connexion se réduit à un système unidimensionnel dont le point fixe satisfait

$$\omega_{i,j}^* = (1 - \eta \tau) \omega_{i,j}^* + \eta S(i,j).$$

Une simple manipulation conduit à

$$\eta S(i,j) = \eta \tau \omega_{i,j}^* \quad \Rightarrow \quad \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

La dérivée de la fonction de mise à jour par rapport à $\omega_{i,j}$ est donnée par

$$\frac{\partial \omega_{i,j}(t+1)}{\partial \omega_{i,j}(t)} = 1 - \eta \tau.$$

La stabilité de ce point fixe est alors garantie si

$$|1 - \eta \tau| < 1,$$

ce qui se traduit par la condition

$$0 < \eta \tau < 2.$$

Dans un réseau complet, si chaque connexion évolue indépendamment selon la même loi, la matrice Jacobienne DF sera diagonale, avec $1 - \eta \tau$ sur chacune de ses entrées diagonales, et ses valeurs propres seront toutes égales à $1 - \eta \tau$. Dans le cas où des interactions entre différentes pondérations apparaissent (par exemple, par le biais d'un mécanisme d'inhibition compétitive ou via l'influence des états internes), la Jacobienne ne sera plus strictement diagonale et des termes hors-diagonale, liés aux dérivées partielles $\frac{\partial S(i,j)}{\partial \omega_{p,q}}$, viendront modifier sa structure. Toutefois, le critère fondamental reste le même : l'ensemble des valeurs propres doit être inférieur à 1 en module pour garantir la stabilité locale.

C. Interprétation : perturbations mineures autour du point fixe

La relation obtenue

$$\delta\Omega(t+1) = DF(\Omega^*) \delta\Omega(t)$$

indique que la réponse du système à une perturbation infinitésimale est linéairement déterminée par la Jacobienne évaluée en Ω^* . En d'autres termes, si l'ensemble des valeurs propres de $DF(\Omega^*)$ se trouve dans le disque unité de \mathbb{C} , alors toute perturbation se contracte au fil du temps et le système revient asymptotiquement à son point fixe. Cette condition, formulée en norme opératoire par

$$\| DF(\Omega^*) \| < 1,$$

est l'exigence fondamentale pour la stabilité locale du point fixe. Dans le cas du modèle additif simple, cette condition se traduit par $0 < \eta \tau < 2$. Dans des systèmes plus complexes, le couplage entre les pondérations et l'influence d'autres variables induisent des contributions supplémentaires dans la Jacobienne, mais l'analyse reste structurée par le même principe : étudier la contraction locale des perturbations via les valeurs propres de la matrice dérivée.

D. Conclusion sur l'analyse locale

La linéarisation autour d'un point fixe Ω^* permet d'obtenir une description locale de la dynamique par le biais de la matrice Jacobienne $DF(\Omega^*)$. Le critère de stabilité locale, essentiel dans la théorie des systèmes dynamiques discrets, se réduit à l'exigence que toutes les valeurs propres λ de cette Jacobienne vérifient

$$|\lambda| < 1.$$

Dans le cas simple où la mise à jour est additive et $S(i,j)$ est constant, cela se traduit par la condition $|1 - \eta \tau| < 1$, soit $0 < \eta \tau < 2$. Lorsque le modèle est étendu pour intégrer des dépendances entre les pondérations et/ou l'influence d'états internes, la structure de la Jacobienne se complexifie, mais l'analyse demeure basée sur le contrôle de la norme de DF autour de Ω^* . Cette méthode analytique constitue ainsi un outil puissant pour évaluer la stabilité locale du point fixe et, par

conséquent, pour anticiper la convergence ou la divergence des trajectoires du système en réponse à de petites perturbations.

Conclusion

En résumé, l'analyse locale par linéarisation – via le calcul de la matrice Jacobienne $DF(\Omega^*)$ – fournit un cadre théorique rigoureux pour évaluer la stabilité du point fixe Ω^* dans un système dynamique discret. Le point fixe est localement attractif si et seulement si toutes les valeurs propres de cette Jacobienne sont strictement inférieures à 1 en module. Dans le modèle additif simple, cette condition se traduit par la contrainte $0 < \eta \tau < 2$, garantissant ainsi que toute petite perturbation autour de Ω^* sera amortie au fil des itérations. En présence de couplages supplémentaires, bien que la structure de la Jacobienne devienne plus complexe, la méthodologie reste identique : contrôler la contraction locale permet de justifier la stabilité de la configuration d'équilibre.

Ce résultat est fondamental pour le Deep Synergy Learning, car il assure que, sous un choix judicieux des paramètres η et τ et sous des hypothèses de bornage et de continuité, la dynamique des poids converge vers des configurations stables, ouvrant la voie à l'émergence de clusters durables et à l'auto-organisation du réseau. Les développements ultérieurs exploreront en profondeur les aspects liés à la multiplicité des attracteurs, aux cycles limités ou encore aux phénomènes de bifurcation qui peuvent survenir lorsque le système est soumis à des rétroactions non linéaires plus complexes.

2.3.2. Attracteurs Multiples et Oscillations

L'analyse locale (2.3.1) révèle que, autour d'un **point fixe** donné, on peut évaluer la stabilité via la linéarisation. Toutefois, même si un **point fixe** Ω^* est localement stable, rien ne garantit qu'il soit **unique** : en fonction du **caractère non linéaire** de la mise à jour (couplages, inhibition compétitive, synergies dépendant de plusieurs liaisons), il est fréquent de rencontrer **plusieurs** attracteurs ou même des **dynamismes** cycliques ou chaotiques. La section 2.3.2 s'intéresse à ces phénomènes, en commençant par la possibilité de **minima (ou attracteurs) multiples** (2.3.2.1), avant d'aborder les **cycles et pseudo-chaos** (2.3.2.2), puis les méthodes pour caractériser ces comportements (2.3.2.3).

2.3.2.1. Possibilité de *minima* multiples : le SCN peut converger vers des topologies différentes selon les conditions initiales

Dans l'approche dynamique adoptée pour le Synergistic Connection Network (SCN), la mise à jour itérative des pondérations conduit à une configuration globale représentée par le vecteur Ω . La dynamique globale du réseau est décrite par l'équation

$$\Omega(t+1) = F(\Omega(t)),$$

et un point fixe Ω^* satisfait la condition d'invariance

$$\Omega^* = F(\Omega^*).$$

Dans un tel contexte, la nature non linéaire de F peut engendrer un paysage complexe d'énergie ou de potentiel qui n'est pas convexe. En conséquence, plusieurs solutions à l'équation du point fixe peuvent exister, chacune correspondant à une configuration stable distincte des pondérations.

Une des implications majeures de cette multiplicité des minima est que l'espace des conditions initiales se décompose en différents bassins d'attraction. Autrement dit, selon l'état initial du système, ou même selon l'ordre d'arrivée des données dans un cadre d'apprentissage en flux, le SCN pourra converger vers l'un ou l'autre de ces attracteurs. Ces configurations, qui représentent des topologies de liaisons différentes, traduisent ainsi la possibilité qu'un même réseau, soumis aux mêmes règles de mise à jour, puisse finalement présenter des partitions ou des clusterisations distinctes.

Parmi les facteurs qui contribuent à l'émergence de minima multiples dans un SCN, on peut citer :

- **La structure de la fonction de synergie $S(\dots)$:**

Lorsque $S(i,j)$ dépend de plusieurs variables ou intègre des couplages non linéaires (par exemple, par l'intermédiaire de mécanismes d'inhibition compétitive ou de dépendances naires), le paysage d'énergie associé au réseau présente de multiples vallées. Chaque vallée correspond à une configuration où les poids ont convergé vers des valeurs stables, et ces configurations peuvent être équivalentes en termes de score global.

- **La présence de rétroactions non linéaires :**

Si la règle de mise à jour intègre des termes contextuels – par exemple, une synergie modulée par la somme des poids entrants ou par des états internes des entités – le couplage entre les connexions renforce le caractère non convexe du paysage. Ainsi, plusieurs arrangements cohérents peuvent satisfaire les équations d'équilibre, chacun étant localement stable.

- **Les états internes des entités :**

L'inclusion d'un vecteur d'état interne \mathbf{s}_i pour chaque entité, qui influence et est influencé par les pondérations $\omega_{i,j}$, apporte une dimension supplémentaire à la dynamique. Cette double rétroaction (entre poids et états internes) peut conduire à l'existence de plusieurs régimes d'auto-organisation, c'est-à-dire à des minima multiples correspondant à différentes topologies d'interconnexion.

Pour illustrer ces idées, considérons un exemple schématique. Supposons que nous disposions de quatre entités $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$ et que la structure de la synergie entre elles permette deux configurations stables équivalentes. Dans un premier scénario, les entités \mathcal{E}_1 et \mathcal{E}_2 forment un cluster tandis que \mathcal{E}_3 et \mathcal{E}_4 forment un second cluster. Dans un autre scénario, une partition alternative peut se dessiner, par exemple en regroupant \mathcal{E}_1 avec \mathcal{E}_3 et \mathcal{E}_2 avec \mathcal{E}_4 . Ces deux configurations, bien que différentes, peuvent présenter des scores globaux (ou niveaux d'énergie)

comparables. Le choix de l'attracteur final dépendra alors des conditions initiales et de l'ordre dans lequel les données arrivent, illustrant ainsi la sensibilité du système aux conditions de départ.

Un autre aspect important est que, dans un modèle intégrant des mécanismes d'inhibition compétitive, chaque entité est contrainte de ne maintenir qu'un nombre limité de liaisons fortes. Cette restriction peut favoriser l'émergence de plusieurs arrangements attracteurs, car plusieurs combinaisons de connexions fortes peuvent satisfaire cette contrainte tout en optimisant localement le score de synergie.

En conclusion, la possibilité d'obtenir des minima multiples dans un SCN découle de la nature non convexe et non linéaire de la fonction F qui gouverne l'évolution du vecteur d'états Ω . Les attracteurs multiples traduisent la présence de différents bassins d'attraction dans l'espace des conditions initiales, de sorte que le réseau peut converger vers des topologies distinctes selon l'historique ou l'ordre d'arrivée des données. Cette caractéristique est particulièrement pertinente pour l'auto-organisation et la formation de clusters, car elle souligne que la structure finale du réseau n'est pas unique mais dépend d'un processus d'apprentissage sensible aux conditions de départ et aux rétroactions internes. La section suivante (2.3.2.2) examinera des cas de cycles et de comportements dynamiques complexes, tandis que les analyses ultérieures approfondiront l'impact de ces multiples attracteurs sur la robustesse et la plasticité du SCN.

2.3.2.2. Apparition éventuelle de cycles ou de régimes “pseudo-chaotiques” dans certains paramétrages (discussion d'exemples)

Dans le cadre du Deep Synergy Learning (DSL), la dynamique globale du réseau est décrite par la relation itérative

$$\Omega(t+1) = F(\Omega(t)),$$

où $\Omega(t)$ représente la configuration complète des pondérations $\{\omega_{i,j}(t)\}$ (éventuellement complétée par d'autres états internes). Lorsque les règles de mise à jour intègrent des non-linéarités – par exemple, via l'inhibition compétitive, des couplages multiples ou la dépendance aux états internes s_i – la carte F acquiert une complexité telle que l'évolution du système ne converge pas nécessairement vers un point fixe unique. Au lieu de cela, plusieurs comportements dynamiques peuvent émerger, parmi lesquels l'apparition de cycles périodiques ou de régimes « pseudo-chaotiques ».

A. Pourquoi un SCN peut-il présenter des cycles ou du pseudo-chaos ?

La clé de l'apparition de telles dynamiques réside dans la richesse non linéaire de la fonction de mise à jour. En effet, si la règle de mise à jour d'un poids $\omega_{i,j}$ ne se contente pas de répondre de manière linéaire au signal de synergie $S(i,j)$ et au terme de décroissance proportionnel à $\tau \omega_{i,j}(t)$, mais intègre également des contributions contextuelles (par exemple, la somme des autres poids connectés à la même entité ou des rétroactions issues des états internes), alors la fonction F devient

suffisamment complexe pour permettre l'existence de plusieurs attracteurs ou même de cycles dynamiques.

Concrètement, si la dérivée locale – autrement dit, l'analyse de la Jacobienne de F au voisinage d'un point fixe – révèle qu'il existe des directions dans lesquelles le module des valeurs propres dépasse 1, le point fixe se déstabilise. Ce phénomène conduit alors, par des bifurcations (par exemple une bifurcation de flip), à l'émergence de cycles de période 2, 3, etc. Dans des systèmes de dimension plus élevée, des enchaînements de bifurcations successives (doublements de période) peuvent aboutir à des régimes où la trajectoire ne converge plus vers un point fixe ni même vers un cycle simple, mais oscille de manière irrégulière dans l'espace d'états – phénomène que l'on qualifie alors de pseudo-chaotique.

Un mécanisme typique dans ce contexte est celui de l'inhibition compétitive : lorsque le renforcement d'un lien, par exemple $\omega_{i,j}$, entraîne l'inhibition des autres liens $\omega_{i,k}$ pour $k \neq j$, le système peut se retrouver dans une situation de va-et-vient, dans laquelle certains liens alternent leur renforcement et leur affaiblissement de manière cyclique.

B. Exemples schématiques

Pour illustrer ces concepts, considérons quelques exemples simplifiés :

- **Cycle de période 2**

Dans un mini-réseau composé de trois entités, il est envisageable que, lors d'une première itération, le lien $\omega_{1,2}$ soit renforcé au détriment de $\omega_{1,3}$ en raison d'un mécanisme d'inhibition. Lors de l'itération suivante, la situation s'inverse : l'affaiblissement de $\omega_{1,2}$ permet à $\omega_{1,3}$ de se renforcer. Ce va-et-vient induit un cycle de période 2, où le système oscille entre deux configurations distinctes.

- **Cycle de période 3**

Dans un système avec des interactions plus complexes – par exemple, lorsque plusieurs connexions interagissent simultanément et que la dépendance aux états internes est prise en compte – le réseau peut alors adopter un cycle de trois états distincts $\Omega_1 \rightarrow \Omega_2 \rightarrow \Omega_3 \rightarrow \Omega_1$. Ce type de comportement est souvent le résultat de rétroactions multiples qui décalent l'équilibre de manière cyclique.

- **Pseudo-chaos dans un système de grande dimension**

Dans un SCN de haute dimension, où le couplage entre de nombreuses pondérations et états internes induit une dynamique non linéaire complexe, il est possible que le système n'atteigne ni un point fixe ni un cycle périodique simple. La trajectoire de $\Omega(t)$ peut alors parcourir de façon irrégulière une région de l'espace d'états sans jamais se répéter, évoquant ainsi un régime « pseudo-chaotique » caractérisé par une sensibilité accrue aux conditions initiales et une errance permanente dans l'attracteur.

C. Discussion mathématique

D'un point de vue théorique, la stabilité d'un point fixe dans un système dynamique discret est déterminée par les valeurs propres de la Jacobienne $DF(\Omega^*)$. Un point fixe est stable si toutes les valeurs propres vérifient $|\lambda| < 1$. Cependant, lorsque, dans certaines directions, une ou plusieurs valeurs propres deviennent supérieures à 1 en module, le point fixe se déstabilise et le système peut alors bifurquer vers un cycle ou, avec des enchaînements successifs de telles bifurcations, vers un comportement chaotique.

Le passage par des cycles de période 2, 4, 8, ... (le phénomène classique de doublement de période) est bien documenté dans la littérature sur les systèmes dynamiques non linéaires, notamment dans le modèle logistique. Bien que la structure exacte de F dans un SCN soit plus complexe, des heuristiques similaires s'appliquent : un taux d'apprentissage η trop élevé ou un couplage (via des mécanismes d'inhibition ou d'autres rétroactions) trop fort peut conduire à l'overshoot et, par conséquent, à l'émergence de cycles. À l'inverse, un ajustement plus modéré de ces paramètres tend à favoriser la convergence vers un point fixe stable.

D. Implications pratiques

D'un point de vue pratique, l'apparition de cycles ou de régimes pseudo-chaotiques dans le SCN présente plusieurs implications :

- **Non-convergence vers un état stable**

Dans un contexte de clustering ou de partitionnement, la présence de cycles signifie que la structure du réseau ne se stabilise pas, ce qui peut compliquer l'extraction d'une partition définitive.

- **Paramétrage délicat**

Afin d'éviter ces comportements oscillatoires indésirables, il est crucial d'ajuster soigneusement les paramètres tels que le taux d'apprentissage η , le coefficient de décroissance τ et l'intensité des mécanismes d'inhibition. Des réglages trop agressifs peuvent, en effet, provoquer des bifurcations multiples et conduire à un comportement pseudo-chaotique.

- **Approche cognitive alternative**

Dans certains modèles inspirés de la biologie ou de la cognition, ces régimes oscillatoires peuvent être interprétés comme des mécanismes d'exploration ou d'attention alternante, où le réseau alterne entre plusieurs configurations d'information. Ces dynamiques, loin d'être de simples défauts, pourraient servir à modéliser des processus cognitifs tels que la résonance neuronale ou la fluctuation de l'attention.

Conclusion (2.3.2.2)

En résumé, dans un SCN dont la règle de mise à jour est fortement non linéaire – notamment par l'intégration de mécanismes d'inhibition compétitive et de couplages complexes – il est tout à fait envisageable que le système n'atteigne pas une convergence vers un point fixe stable. Au lieu de cela, des cycles (de périodes 2, 3, etc.) ou des régimes « pseudo-chaotiques » peuvent apparaître. Cette dynamique, qui résulte du dépassement de certains seuils dans la Jacobienne locale de F (c'est-à-dire lorsque $|\lambda| > 1$ dans certaines directions), illustre la richesse des comportements possibles dans des systèmes non linéaires de grande dimension. D'un point de vue pratique, si l'objectif est de stabiliser la structure du réseau en vue d'une partition claire (clustering stable), il est souvent nécessaire d'ajuster les paramètres du système (notamment η et τ) pour éviter de tels régimes oscillatoires. Cependant, ces comportements peuvent aussi être interprétés positivement dans un cadre cognitif, où l'exploration constante des configurations pourrait être associée à des processus d'adaptation ou d'échantillonnage dynamique.

Ainsi, l'apparition éventuelle de cycles ou de régimes pseudo-chaotiques dans un SCN n'est pas simplement un artefact mathématique, mais reflète la complexité inhérente aux interactions non linéaires du système, et offre une richesse dynamique qui peut être exploitée selon les objectifs visés, que ce soit pour stabiliser l'auto-organisation ou pour modéliser des comportements d'exploration et de résonance cognitive.

2.3.2.3. Stratégies pour détecter et caractériser ces phénomènes

Dans le cadre de l'étude des systèmes dynamiques non linéaires appliqués au **Synergistic Connection Network (SCN)**, il est primordial de développer des stratégies rigoureuses permettant de détecter et de caractériser les comportements complexes tels que la convergence vers des attracteurs multiples, l'apparition de cycles périodiques ou encore des régimes de type pseudo-chaotique. La dynamique du SCN s'exprime par l'équation itérative

$$\boldsymbol{\Omega}(t+1) = F(\boldsymbol{\Omega}(t)),$$

où le vecteur $\boldsymbol{\Omega}(t) \in \mathbb{R}^D$ représente l'ensemble des **pondérations** $\omega_{i,j}(t)$ et éventuellement d'autres variables d'état, telles que les **états internes** $\mathbf{s}_i(t)$. La complexité inhérente à la fonction F – qui intègre les effets de la **synergie** $S(i, j)$, des mécanismes d'inhibition compétitive, et d'éventuelles dépendances croisées entre les pondérations – peut amener le système à ne pas converger vers un point fixe unique, mais à adopter des comportements oscillatoires ou irréguliers.

Une première approche pour détecter ces phénomènes consiste à réaliser des **simulations numériques**. On initialise le SCN avec différentes conditions initiales $\boldsymbol{\Omega}(0)$ et on varie les paramètres clés du modèle, tels que le **taux d'apprentissage** η et le **coefficient de décroissance** τ . En observant l'évolution des pondérations $\omega_{i,j}(t)$ au fil du temps, on peut distinguer trois scénarios typiques. Si, après un nombre suffisant d'itérations, les pondérations convergent vers des valeurs constantes, cela indique l'existence d'un point fixe attracteur. À l'inverse, si les valeurs oscillent entre deux configurations ou plus, on déduit la présence d'un cycle de période correspondante, par exemple en constatant que

$$\boldsymbol{\Omega}(t+p) \approx \boldsymbol{\Omega}(t)$$

pour une période p donnée. Enfin, si aucune périodicité claire ne se dégage et que la trajectoire de $\Omega(t)$ évolue de manière erratique, l'hypothèse d'un régime pseudo-chaotique est alors plausible.

Pour les systèmes de faible dimension, il est souvent utile d'effectuer une **projection** de la trajectoire dans un espace à deux ou trois dimensions afin de visualiser les trajectoires de $\Omega(t)$. Par exemple, si l'on considère que D est réduit, on peut représenter graphiquement la suite $\{\Omega(t)\}_{t=0}^T$ dans \mathbb{R}^3 pour observer d'éventuelles boucles ou motifs répétés. Dans des réseaux de grande dimension, des outils de visualisation tels que des **cartes de chaleur** ou des **animations** peuvent être employés pour suivre l'évolution des composantes individuelles des vecteurs $\Omega(t)$.

La théorie des systèmes dynamiques offre également des outils formels pour la caractérisation des comportements non convergents. Lorsqu'un point fixe Ω^* est identifié, on peut calculer la **matrice Jacobienne** $DF(\Omega^*)$ qui, localement, linéarise la dynamique autour de Ω^* par la relation

$$\delta\Omega(t+1) = DF(\Omega^*) \delta\Omega(t),$$

où $\delta\Omega(t) = \Omega(t) - \Omega^*$. La stabilité locale du point fixe est alors déterminée par les valeurs propres λ de $DF(\Omega^*)$. Si l'une des valeurs propres satisfait $|\lambda| > 1$, le point fixe se déstabilise dans la direction correspondante, ce qui peut entraîner l'apparition d'un cycle de période 2 (par exemple, via une bifurcation de flip) ou de comportements plus complexes lorsque plusieurs valeurs propres dépassent 1 en module.

Afin de quantifier la présence de régimes chaotiques, on peut recourir au calcul de l'**exposant de Lyapunov maximal** λ_{\max} , défini par

$$\lambda_{\max} = \lim_{t \rightarrow \infty} \frac{1}{t} \ln \frac{\|\delta\Omega(t)\|}{\|\delta\Omega(0)\|}.$$

Un exposant de Lyapunov positif indique une sensibilité extrême aux conditions initiales, caractéristique du chaos discret, et confirme ainsi la présence d'un régime pseudo-chaotique.

Pour détecter la **multi-stabilité** du système, il est recommandé de réaliser plusieurs simulations en démarrant à partir de différentes conditions initiales, appelées « multi-run ». Si ces simulations convergent vers des attracteurs différents, cela démontre que le paysage d'énergie du SCN comporte plusieurs minima locaux, chacun disposant de son propre bassin d'attraction. Une autre méthode consiste à perturber légèrement un attracteur supposé stable, par exemple en posant

$$\Omega(0) = \Omega^* + \delta\Omega(0),$$

et en observant si la trajectoire converge toujours vers Ω^* ou si elle bascule vers une autre configuration attractrice.

L'ensemble de ces approches, qu'il s'agisse de simulations numériques, de l'analyse de la matrice Jacobienne ou de l'estimation des exposants de Lyapunov, fournit un cadre complet pour détecter et caractériser les phénomènes de cycles, de multi-attracteurs et de pseudo-chaos dans un SCN. Ces méthodes permettent d'obtenir une compréhension fine des dynamiques non convergentes et, par conséquent, d'adapter le paramétrage du système (notamment les valeurs de η et de τ) afin de favoriser, selon les objectifs, la convergence vers des configurations stables ou, au contraire, de tirer parti des régimes oscillatoires pour modéliser des comportements cognitifs dynamiques.

En somme, la combinaison des techniques de simulation, de linéarisation locale par la matrice Jacobienne et d'analyse quantitative via les exposants de Lyapunov constitue un ensemble d'outils puissants pour détecter et caractériser les phénomènes complexes dans un SCN. Ces stratégies permettent d'identifier si le système converge vers un point fixe attracteur, s'il entre dans un cycle périodique ou s'il adopte un comportement pseudo-chaotique, et offrent ainsi des indications précieuses pour le réglage fin des paramètres du DSL ainsi que pour l'interprétation des processus d'auto-organisation sous-jacents.

2.3.3. Analyse de la Formation de Clusters

Dans un **SCN** (Synergistic Connection Network), l'une des manifestations les plus remarquables de la **stabilité** (voire de la multi-stabilité, 2.3.2) est l'**auto-organisation** en **clusters** plus ou moins durables. Les sections précédentes (2.3.1 et 2.3.2) se concentraient sur la dynamique globale, l'existence de points fixes ou de cycles. Ici (2.3.3), nous nous focalisons plus spécifiquement sur la **notion** de cluster et sur la manière dont ils émergent et se maintiennent. Après avoir défini ce qu'on entend par “cluster stable” (2.3.3.1), nous aborderons (2.3.3.2) les **mesures** (cohésion, modularité) permettant de quantifier la qualité de ces regroupements, puis nous distinguerons (2.3.3.3) les clusters **transitoires** de ceux plus établis (macro-clusters), susceptibles d'apparaître à grande échelle.

2.3.3.1. Concept de “cluster stable” : définition, critères de cohésion interne

Dans le cadre du **Synergistic Connection Network (SCN)**, la notion de **cluster** représente un concept central dans l'étude de l'auto-organisation. Un **cluster**, ou sous-groupe, se définit comme un **ensemble** d'entités $\mathcal{C} \subseteq \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ dans lequel les **liens** entre les entités internes, notés $\omega_{i,j}$ pour $i, j \in \mathcal{C}$, sont particulièrement **forts** et traduisent une **synergie** élevée. Parallèlement, les connexions reliant une entité $\mathcal{E}_i \in \mathcal{C}$ à une entité extérieure $\mathcal{E}_k \notin \mathcal{C}$ se caractérisent par des valeurs de $\omega_{i,k}$ nettement plus faibles. Cette dichotomie, intrinsèque à l'auto-organisation, émerge spontanément lorsque, au cours de l'évolution du réseau, certains liens se renforcent tandis que d'autres s'affaiblissent, conduisant à la formation de structures en « blocs » clairement identifiables dans la matrice des pondérations.

La notion de **stabilité** d'un cluster se décline en deux aspects complémentaires qui méritent d'être étudiés en profondeur. D'une part, la **stabilité dynamique** se traduit par le maintien dans le temps des **pondérations internes** du cluster, de sorte que les connexions $\omega_{i,j}$ entre les entités $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{C}$ restent élevées et ne se dégradent pas, alors que les liens avec l'extérieur demeurent faibles ou, dans certains cas, disparaissent complètement. Cette propriété dynamique signifie que de petites perturbations, telles que des fluctuations légères de $\omega_{i,j}$, n'ont pas la capacité de déstabiliser le sous-groupe ni de modifier substantiellement sa structure. D'autre part, la **cohérence interne** se manifeste par un principe d'auto-renforcement : les entités appartenant à \mathcal{C} maintiennent une synergie mutuelle qui est systématiquement supérieure à celle qu'elles entretiennent avec des entités extérieures. Cette cohérence peut être formalisée par la définition d'un indice de cohésion, tel que la moyenne des pondérations internes, qui doit rester au-dessus d'un certain seuil pour que l'on puisse parler d'une forte cohésion.

Pour formaliser ces idées, on introduit d'abord l'indicateur de **cohésion minimale interne** défini par

$$\bar{\omega}_{\text{in}}(\mathcal{C}) = \frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} \omega_{i,j}.$$

Cet indicateur doit être supérieur à un seuil prédéfini $\theta_{\text{in}} > 0$ pour garantir que les interactions au sein du cluster sont suffisamment fortes. Parallèlement, l'indicateur de **faible connectivité externe** est défini par

$$\bar{\omega}_{\text{ext}}(\mathcal{C}) = \frac{1}{|\mathcal{C}|(n - |\mathcal{C}|)} \sum_{\substack{i \in \mathcal{C} \\ k \notin \mathcal{C}}} \omega_{i,k},$$

et il est requis que $\bar{\omega}_{\text{ext}}(\mathcal{C})$ demeure inférieur à un seuil θ_{ext} . La persistance de ces deux critères sur une période temporelle significative, notée par l'intervalle $t \in [T_0, T_1]$, assure que le cluster reste stable face aux fluctuations, ce qui indique que l'ensemble \mathcal{C} agit comme un **attracteur local** dans l'espace dynamique du réseau.

Cette approche permet de traduire de manière formelle le concept de **cluster stable**. On peut exprimer cette stabilité par l'inégalité

$$\frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} \omega_{i,j}(t) \gg \frac{1}{|\mathcal{C}|(n - |\mathcal{C}|)} \sum_{\substack{i \in \mathcal{C} \\ k \notin \mathcal{C}}} \omega_{i,k}(t),$$

qui doit être vérifiée de manière persistante pour $t \in [T_0, T_1]$. L'interprétation mathématique de cette inégalité consiste à affirmer que la **moyenne interne** des pondérations est bien supérieure à la **moyenne externe**, garantissant ainsi une **séparation nette** entre les interactions intra-cluster et inter-cluster.

Les avantages de cette approche résident dans sa capacité à fournir une mesure quantitative de la cohésion interne et de la séparation externe, ce qui est essentiel pour identifier et analyser les regroupements dans le cadre d'un apprentissage non supervisé. L'utilisation de telles mesures permet non seulement de détecter des clusters stables, mais également de caractériser leur robustesse face aux perturbations. Toutefois, un inconvénient potentiel est que les seuils θ_{in} et θ_{ext} doivent être choisis en fonction de l'échelle typique des pondérations dans le SCN, ce qui peut rendre l'approche sensible à la normalisation des données et aux paramètres du système.

En conclusion, le **concept de “cluster stable”** dans le SCN repose sur la persistance d'un sous-ensemble d'entités \mathcal{C} dans lequel les **liens internes** sont significativement plus forts que les **liaisons externes**, traduisant ainsi une **stabilité dynamique** et une **cohérence interne** durable. La condition

$$\frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} \omega_{i,j}(t) \gg \frac{1}{|\mathcal{C}|(n - |\mathcal{C}|)} \sum_{\substack{i \in \mathcal{C} \\ k \notin \mathcal{C}}} \omega_{i,k}(t)$$

remplit un rôle essentiel pour quantifier cette stabilité et sert de fondement à l'analyse de la **clusterisation** dans un **Deep Synergy Learning**. Les approches développées ici seront enrichies dans les sections ultérieures, notamment dans **2.3.3.2** où des mesures plus élaborées, telles que la **modularité** et d'autres indices de cohésion, seront introduites afin de différencier les clusters transitoires des véritables **macro-clusters** stables dans le réseau.

2.3.3.2. Mesures possibles (cohésion, modularité) pour juger la qualité ou la force d'un regroupement

Dans le cadre de l'analyse des **clusters** issus du Synergistic Connection Network (**SCN**), il est fondamental de disposer de critères quantitatifs permettant de mesurer la **cohésion interne** et la **séparation externe** d'un sous-ensemble d'entités $\mathcal{C} \subseteq \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Ces mesures visent à déterminer dans quelle mesure les **liaisons** $\omega_{i,j}$ établies entre les entités au sein du cluster sont significativement plus fortes que celles reliant les membres de \mathcal{C} aux entités extérieures. Un premier indicateur est la **densité interne** qui évalue la moyenne (ou la somme) des pondérations entre toutes les paires d'entités appartenant à \mathcal{C} . Par exemple, on définit la densité interne par

$$\bar{\omega}_{\text{in}}(\mathcal{C}) = \frac{1}{|\mathcal{C}|(|\mathcal{C}| - 1)} \sum_{\substack{i, j \in \mathcal{C} \\ i \neq j}} \omega_{i,j}.$$

Cette moyenne quantifie le niveau de **force** des interactions à l'intérieur du cluster et, plus sa valeur est élevée, plus la cohésion interne est importante. En parallèle, il est pertinent de mesurer la **connectivité externe** du cluster en évaluant la moyenne des pondérations entre les entités à l'intérieur de \mathcal{C} et celles situées dans le complémentaire $V \setminus \mathcal{C}$ du réseau. On définit ainsi

$$\bar{\omega}_{\text{ext}}(\mathcal{C}) = \frac{1}{|\mathcal{C}|(n - |\mathcal{C}|)} \sum_{\substack{i \in \mathcal{C} \\ k \notin \mathcal{C}}} \omega_{i,k}.$$

L'interprétation de ces deux mesures permet d'établir un critère de séparation, qui peut être exprimé par le **ratio** suivant

$$\text{ratio}(\mathcal{C}) = \frac{\bar{\omega}_{\text{in}}(\mathcal{C})}{\bar{\omega}_{\text{ext}}(\mathcal{C}) + \epsilon'}$$

où $\epsilon' > 0$ est une constante très faible introduite pour éviter toute division par zéro. Un ratio élevé indique que le cluster est fortement cohésif par rapport à ses connexions extérieures, ce qui constitue un indice quantitatif fort de **stabilité** et de **qualité** du regroupement.

Une approche complémentaire consiste à recourir à la notion de **modularité**, empruntée à l'analyse des communautés dans les graphes pondérés. Dans ce contexte, la modularité permet de comparer la somme des **liaisons internes** à ce que l'on obtiendrait dans un modèle aléatoire conservant la même distribution des degrés. Pour un graphe pondéré $G = (V, E)$, la modularité, dans sa forme Newman-Girvan adaptée, se définit par

$$Q = \frac{1}{2W} \sum_{\substack{i, j \in V \\ \text{même cluster}}} \left[\omega_{ij} - \frac{k_i k_j}{2W} \right],$$

où $2W = \sum_{i,j} \omega_{ij}$ représente le total des pondérations du graphe et $k_i = \sum_j \omega_{ij}$ désigne le degré pondéré du noeud i . Un score de modularité élevé, souvent supérieur à 0.3 ou 0.4 selon l'échelle, témoigne d'une division du graphe en communautés (ou clusters) qui exhibent des **liaisons internes**

significativement supérieures à ce qui serait attendu par hasard. Cette mesure, en intégrant des références statistiques au modèle aléatoire, offre un cadre rigoureux pour évaluer la **qualité** des regroupements et, par extension, leur pertinence dans l'analyse des données.

Par ailleurs, il existe d'autres indices issus de l'analyse des graphes, tels que la **conductance** ou la **densité relative**, qui permettent d'évaluer la proportion des liaisons sortantes par rapport à l'ensemble des liaisons d'un cluster. La conductance, par exemple, est faible lorsque le cluster présente une forte cohésion interne et une faible connectivité externe. Bien que ces indices soient souvent utilisés en complément de la modularité, leur emploi dans le contexte du SCN permet de disposer d'une palette d'outils pour apprécier de manière fine la **robustesse** d'un regroupement.

Du point de vue pratique, ces mesures jouent un rôle crucial dans le cadre du **Deep Synergy Learning**. En effet, elles servent non seulement à identifier les clusters qui émergent de manière auto-organisée, mais également à comparer différentes partitions obtenues dans des simulations multi-run, notamment lorsque le réseau présente plusieurs attracteurs possibles. Ces outils facilitent ainsi la visualisation et l'analyse des structures en blocs qui se dégagent dans la matrice des pondérations, permettant de regrouper les entités en “macro-nœuds” pour simplifier l'interprétation globale du réseau.

En conclusion, la quantification de la **cohésion interne** et de la **séparation externe** via des mesures telles que $\bar{\omega}_{\text{in}}(\mathcal{C})$, $\bar{\omega}_{\text{ext}}(\mathcal{C})$ et le **ratio** associé, complétée par le calcul de la **modularité**,

$$Q = \frac{1}{2W} \sum_{\substack{i,j \in V \\ \text{même cluster}}} \left[\omega_{ij} - \frac{k_i k_j}{2W} \right],$$

fournit un cadre analytique robuste pour juger de la **qualité** et de la **force** d'un regroupement dans un SCN. Ces indicateurs permettent de distinguer les clusters **stables** – qui se caractérisent par une cohésion interne élevée et une séparation nette par rapport au reste du réseau – des configurations transitoires, apportant ainsi des bases solides pour l'analyse des communautés dans le contexte du Deep Synergy Learning. La section suivante (2.3.3.3) approfondira ces concepts en distinguant les clusters transitoires des véritables macro-clusters qui émergent sur le long terme.

2.3.3.3. Distinction entre clusters transitoires et macro-clusters à large échelle

Dans le cadre du **Synergistic Connection Network (SCN)**, l'analyse des regroupements d'entités conduit à distinguer deux phénomènes d'auto-organisation qui, bien que fondamentalement liés, diffèrent par leur durabilité et leur rôle dans la structure globale du réseau. Le premier phénomène correspond aux **clusters transitoires**, qui se caractérisent par une **cohésion interne** temporaire et une apparition éphémère dans la dynamique du SCN, tandis que le second se réfère aux **macro-clusters** qui, par leur stabilité et leur persistance sur de longues périodes, se révèlent comme des attracteurs dominants dans l'espace des configurations.

La notion de **cluster transitoire** désigne un sous-ensemble $\mathcal{C} \subseteq \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ dont les liaisons internes, représentées par l'ensemble des pondérations $\{\omega_{i,j} \mid i, j \in \mathcal{C}\}$, atteignent un niveau élevé pendant un intervalle de temps limité. Formellement, si l'on définit la cohésion interne moyenne par

$$\bar{\omega}_{\text{in}}(\mathcal{C}, t) = \frac{1}{|\mathcal{C}|(|\mathcal{C}| - 1)} \sum_{\substack{i,j \in \mathcal{C} \\ i \neq j}} \omega_{i,j}(t),$$

et la connectivité externe moyenne par

$$\bar{\omega}_{\text{ext}}(\mathcal{C}, t) = \frac{1}{|\mathcal{C}|(n - |\mathcal{C}|)} \sum_{\substack{i \in \mathcal{C} \\ k \notin \mathcal{C}}} \omega_{i,k}(t),$$

alors un cluster transitoire se manifeste par une condition telle que, pour un intervalle temporel restreint $t \in [T_0, T_1]$,

$$\bar{\omega}_{\text{in}}(\mathcal{C}, t) \gg \bar{\omega}_{\text{ext}}(\mathcal{C}, t).$$

Ce regroupement peut apparaître en raison de fluctuations temporaires dans les **états internes** des entités ou lors de phases de transition où de petits sous-groupes se forment brièvement avant de fusionner ou de se dissoudre. Ces phénomènes, bien que révélateurs d'une forte synergie momentanée, ne garantissent pas la pérennité de la structure car de nouvelles mises à jour, l'effet d'une inhibition compétitive ou l'évolution des synergies, peuvent entraîner la disparition rapide de ces regroupements.

À l'inverse, le concept de **macro-cluster** se réfère à un regroupement stable et persistant. Un macro-cluster est défini comme un sous-ensemble \mathcal{C} dont la cohésion interne, mesurée par

$$\bar{\omega}_{\text{in}}(\mathcal{C}, t),$$

se maintient à un niveau élevé pendant une période prolongée, et dont la connectivité externe, $\bar{\omega}_{\text{ext}}(\mathcal{C}, t)$, reste faible de façon durable. On peut caractériser la stabilité d'un macro-cluster par l'inégalité

$$\frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} \omega_{i,j}(t) \gg \frac{1}{|\mathcal{C}|(n - |\mathcal{C}|)} \sum_{\substack{i \in \mathcal{C} \\ k \notin \mathcal{C}}} \omega_{i,k}(t),$$

qui doit être vérifiée de manière continue sur un intervalle temporel $[T_0, T_1]$ suffisamment long pour certifier la **persistante** et la **robustesse** du regroupement. La stabilité d'un macro-cluster s'exprime également par une faible **variance** temporelle de $\bar{\omega}_{\text{in}}(\mathcal{C}, t)$, signe que les synergies internes ne fluctuent pas de manière excessive, et par la capacité du cluster à résister à des **perturbations** mineures (par exemple, un léger bruit ajouté aux pondérations).

Pour distinguer concrètement les clusters transitaires des macro-clusters, plusieurs approches méthodologiques peuvent être mises en œuvre. L'utilisation d'une **fenêtre glissante** permet de segmenter l'évolution temporelle en intervalles $[t, t + \Delta]$ sur lesquels les mesures $\bar{\omega}_{\text{in}}(\mathcal{C}, t)$ et $\bar{\omega}_{\text{ext}}(\mathcal{C}, t)$ sont calculées. Si, au sein d'une seule fenêtre, la cohésion interne atteint un pic mais retombe ensuite, on identifie un regroupement transitoire. Par ailleurs, l'application d'un **test de perturbation**, consistant à injecter un bruit faible dans la configuration des pondérations, peut révéler la robustesse d'un macro-cluster : un cluster stable se reconstitue rapidement après perturbation, tandis qu'un regroupement éphémère s'effondre.

Un exemple schématique permet d'illustrer cette distinction. Considérons un SCN de cinq entités où, lors des premières itérations, un petit sous-groupe tel que $\{\mathcal{E}_1, \mathcal{E}_2\}$ présente une forte cohésion interne, indiquée par un pic temporaire de $\bar{\omega}_{\text{in}}(\{\mathcal{E}_1, \mathcal{E}_2\}, t)$. Toutefois, si, lors des itérations suivantes, cette cohésion ne persiste pas et que ces entités finissent par être intégrées dans un regroupement plus vaste, alors $\{\mathcal{E}_1, \mathcal{E}_2\}$ correspond à un **cluster transitoire**. En revanche, si une partition plus large, par exemple $\mathcal{C} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$, se maintient avec un $\bar{\omega}_{\text{in}}(\mathcal{C}, t)$ élevé et une $\bar{\omega}_{\text{ext}}(\mathcal{C}, t)$ faible sur une durée prolongée, alors ce regroupement constitue un **macro-cluster** stable.

Les **avantages** d'une telle approche résident dans la possibilité de quantifier de manière rigoureuse la qualité d'un regroupement via des indices mathématiques tels que la densité interne et la **modularité**. Cette modularité, définie par

$$Q = \frac{1}{2W} \sum_{\substack{i,j \in V \\ \text{même cluster}}} \left[\omega_{ij} - \frac{k_i k_j}{2W} \right],$$

avec $2W = \sum_{i,j} \omega_{ij}$ et $k_i = \sum_j \omega_{ij}$, offre une mesure comparative entre le regroupement observé et ce que l'on pourrait obtenir dans un modèle aléatoire. Un score de modularité élevé indique que les liens internes sont bien supérieurs aux attentes d'un réseau aléatoire, attestant ainsi de la robustesse et de la pertinence du regroupement. Les **limites** de cette méthode résident principalement dans la nécessité d'ajuster les seuils θ_{in} et θ_{ext} en fonction de l'échelle des pondérations dans le SCN, ainsi que dans la sensibilité potentielle des mesures à des fluctuations temporelles qui, dans des systèmes très dynamiques, pourraient rendre la distinction entre clusters transitaires et macro-clusters moins nette.

En conclusion, la distinction entre **clusters transitaires** et **macro-clusters à large échelle** repose sur l'analyse de la persistance et de la robustesse des liens internes par rapport aux connexions externes. Un cluster transitoire est caractérisé par une **cohésion interne** momentanée qui ne se maintient pas sur la durée, tandis qu'un macro-cluster se distingue par une **stabilité dynamique** persistante et une séparation nette des interactions internes et externes. Cette distinction, formalisée par l'inégalité

$$\frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} \omega_{i,j}(t) \gg \frac{1}{|\mathcal{C}|(n - |\mathcal{C}|)} \sum_{\substack{i \in \mathcal{C} \\ k \notin \mathcal{C}}} \omega_{i,k}(t),$$

et complétée par des indicateurs tels que la modularité Q , constitue une base rigoureuse pour l'identification et l'analyse des regroupements au sein d'un SCN. Ces mesures quantitatives permettent ainsi de distinguer les regroupements éphémères, susceptibles de se dissoudre rapidement, des structures stables qui émergent en tant qu'attracteurs durables dans la dynamique d'auto-organisation du réseau. La section suivante (2.3.3.3) approfondira cette distinction en examinant des cas pratiques et en proposant des méthodes complémentaires pour évaluer la qualité des clusters dans un contexte d'apprentissage non supervisé.

2.3.4. Influence du Bruit et des Perturbations

L'analyse de la **stabilité** et de la **formation de clusters** (sections 2.3.1 à 2.3.3) s'est déroulée dans un cadre relativement “idéal”. Dans la **pratique**, un SCN (Synergistic Connection Network) peut faire face à diverses **perturbations** ou à un **bruit** injecté — que ce bruit affecte la **mesure de synergie** (données incertaines, capteurs bruyants, etc.) ou qu'il concerne directement la **dynamique** des pondérations (ajout d'aléas dans la mise à jour). La section 2.3.4 examine comment ces facteurs perturbateurs influent sur la trajectoire du SCN et la robustesse des **clusters** qu'il forme. Après une introduction (2.3.4.1) sur la façon dont le SCN réagit au bruit, on abordera la **résilience** (2.3.4.2) et l'idée qu'un **certain** niveau de bruit peut même favoriser l'exploration (2.3.4.3).

2.3.4.1. Comment le SCN réagit à un bruit injecté sur les entités ou sur la fonction de synergie

Dans le cadre du **Synergistic Connection Network (SCN)**, la présence de **bruit** joue un rôle crucial dans la dynamique d'auto-organisation. La perturbation aléatoire, qui peut être introduite à divers niveaux — que ce soit dans la **représentation** des entités \mathbf{x}_i ou de leurs **états internes** \mathbf{s}_i , dans le calcul de la **fonction de synergie** $S(i, j)$ ou encore directement dans la **mise à jour** des **pondérations** $\omega_{i,j}$ — influence directement la capacité du réseau à converger vers une configuration stable. En effet, le SCN évolue selon l'équation de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

où le terme $\eta [S(i, j) - \tau \omega_{i,j}(t)]$ détermine l'ajustement des poids. L'injection d'un **bruit** peut modifier l'issue de cette évolution de plusieurs manières.

Lorsqu'un **bruit** est injecté dans la **représentation** des entités, par exemple en ajoutant une perturbation aléatoire à chaque vecteur \mathbf{x}_i , la **distance** ou la **similarité** utilisée pour calculer $S(i, j)$ sera modifiée. Autrement dit, même si la structure intrinsèque des entités demeure identique, les mesures telles que

$$S(i, j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|)$$

peuvent varier en raison d'une fluctuation aléatoire dans \mathbf{x}_i ou \mathbf{x}_j . Ainsi, deux entités qui étaient auparavant considérées comme proches peuvent temporairement apparaître éloignées, ce qui affecte négativement la **consolidation** de leurs liens et peut retarder, voire empêcher, la formation d'un regroupement stable.

De même, si le **bruit** est directement appliqué sur la **fonction de synergie**, on peut modéliser cette perturbation par l'introduction d'un terme aléatoire $\varepsilon_{i,j}(t)$ dans le calcul de la synergie. On écrira alors

$$S_{\text{obs}}(i, j) = S_{\text{réel}}(i, j) + \varepsilon_{i,j}(t),$$

ce qui implique que la valeur utilisée pour mettre à jour les poids est bruitée. Même lorsque les représentations \mathbf{x}_i et \mathbf{x}_j sont parfaitement stables, la présence de $\varepsilon_{i,j}(t)$ conduit à une fluctuation dans le signal de renforcement, induisant ainsi des variations aléatoires dans l'évolution de $\omega_{i,j}$.

Il est également possible d'injecter le **bruit** directement dans la **mise à jour** des pondérations. Dans ce cas, l'équation de mise à jour se modifie en ajoutant un terme perturbateur $\xi_{i,j}(t)$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t).$$

Même si la valeur de $S(i,j)$ reste exacte, ce terme $\xi_{i,j}(t)$ génère des fluctuations supplémentaires qui font osciller les poids autour de leur point d'équilibre théorique $\omega_{i,j}^* \approx S(i,j)/\tau$.

Les **effets** du **bruit** dépendent de son amplitude relative au terme de correction $\eta [S(i,j) - \tau \omega_{i,j}(t)]$. Lorsque le **bruit** est modeste, la dynamique fondamentale de renforcement et de décroissance domine, de sorte que le réseau parvient à stabiliser ses pondérations malgré de légères fluctuations, voire à explorer de nouveaux minima locaux dans l'espace d'état, ce qui peut parfois éviter de rester bloqué dans une configuration sous-optimale. En revanche, si le **bruit** atteint une amplitude comparable, voire supérieure, à celle du terme de correction, la convergence vers un point fixe est perturbée. Le système adopte alors une dynamique où les pondérations évoluent de façon erratique, ce qui se traduit par un comportement de **diffusion aléatoire** ou par un « mouvement brownien » autour du point d'équilibre.

Les **avantages** potentiels d'un bruit modéré résident dans sa capacité à permettre au SCN de s'extraire de minima locaux, favorisant ainsi l'exploration de configurations alternatives qui pourraient être plus adaptées. Toutefois, les **limites** de cette approche apparaissent lorsque le bruit devient trop important, car il perturbe significativement la stabilité du réseau et empêche la formation de **clusters** robustes, rendant ainsi l'ensemble de la structure instable et difficilement interprétable.

En conclusion, l'effet du **bruit** dans un SCN est à la fois ambivalent et paramétrable. Si l'injection d'un bruit faible dans les **entités**, dans la **fonction de synergie** ou dans la **mise à jour** des **pondérations** permet de renforcer l'exploration du paysage d'énergie et d'éviter des blocages, une perturbation trop forte empêche la stabilisation des liens et retardé, voire annule, la formation de structures cohérentes. Les paramètres η et τ jouent ici un rôle déterminant en comparant leur contribution au signal de correction avec l'amplitude du bruit, déterminant ainsi si le **Deep Synergy Learning** parvient à conserver une topologie cohérente ou s'il bascule dans un régime plus erratique.

2.3.4.2. Résilience vs. Sensibilité : Cas d'Exemple

Dans l'analyse des **Synergistic Connection Networks (SCN)**, l'étude de la réponse du système à l'injection de bruit permet d'appréhender la robustesse de sa dynamique d'auto-organisation. On s'intéresse particulièrement à la manière dont le système réagit lorsque des perturbations aléatoires sont introduites dans le calcul de la **fonction de synergie** $S(i,j)$ ou directement dans l'équation de mise à jour des **pondérations** $\omega_{i,j}$. Pour illustrer ces phénomènes, considérons d'abord la règle de mise à jour dite « additive » qui constitue le cœur de la dynamique du SCN :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ représente le taux d'apprentissage et $\tau > 0$ le coefficient de décroissance. Dans le cas idéal, et en l'absence de perturbations, cette mise à jour conduit à une convergence locale des pondérations vers un **point fixe** théorique défini par

$$\omega_{i,j}^* \approx \frac{S(i,j)}{\tau}.$$

Pour modéliser l'impact des perturbations, on introduit un terme de bruit $\xi_{i,j}(t)$ dans la dynamique, ce qui donne l'équation suivante :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t).$$

Le terme $\xi_{i,j}(t)$ représente des fluctuations aléatoires qui viennent perturber la trajectoire de convergence de $\omega_{i,j}(t)$. La comparaison entre l'amplitude du terme de correction, $\eta [S(i,j) - \tau \omega_{i,j}(t)]$, et celle du bruit $\xi_{i,j}(t)$ détermine si le système affiche une **résilience** ou une **sensibilité** accentuée face aux perturbations.

Exemple de Résilience :

Supposons un mini-SCN composé de quatre entités, notées $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4\}$, pour lesquelles les synergies internes $S(1,2)$ et $S(3,4)$ sont élevées (par exemple, $S(1,2), S(3,4) > 0.7$), tandis que les synergies croisées telles que $S(1,3)$, $S(1,4)$, $S(2,3)$ et $S(2,4)$ restent faibles (typiquement < 0.2). Dans ce cas, la mise à jour itérative renforce fortement les pondérations internes, de sorte que les valeurs $\omega_{1,2}$ et $\omega_{3,4}$ atteignent des niveaux élevés (par exemple, entre 0.6 et 0.8), tandis que les liaisons entre les groupes restent faibles (inférieures à 0.1). Si l'on ajoute un bruit additif modeste, par exemple $\xi_{i,j}(t) \in [-0.02, +0.02]$, l'effet sur la dynamique demeure marginal. En d'autres termes, la structure de clusters constituée par les paires $\{\mathcal{E}_1, \mathcal{E}_2\}$ et $\{\mathcal{E}_3, \mathcal{E}_4\}$ reste stable, démontrant une **résilience** élevée du SCN face aux perturbations faibles.

Exemple de Sensibilité :

En revanche, considérons un scénario dans lequel le SCN comprend cinq entités $\{\mathcal{E}_1, \dots, \mathcal{E}_5\}$ et que la synergie interne d'un potentiel regroupement, par exemple $S(1,2) \approx 0.3$, est modérée et ne diffère pas significativement des synergies externes, qui se situeraient dans un intervalle approximatif de 0.1 à 0.2. Dans ce contexte, l'injection d'un bruit d'amplitude supérieure, disons $\xi_{i,j}(t) \in [-0.05, +0.1]$, peut provoquer des fluctuations plus importantes. La valeur de $\omega_{1,2}(t)$ peut alors chuter de valeurs stables initialement situées autour de 0.25–0.3 à des niveaux inférieurs à 0.15. Une telle réduction critique de la pondération interne conduit à la désagrégation du cluster potentiel, illustrant une **sensibilité** marquée aux perturbations. Ce cas démontre que lorsque la marge de différenciation entre les synergies internes et externes est faible, le SCN devient fragile et vulnérable à des fluctuations aléatoires, même si celles-ci ne sont pas très élevées.

Cas de Multi-Attracteurs :

Dans certains systèmes comportant plusieurs attracteurs, comme discuté en section 2.3.2.1, l'injection de bruit peut jouer un rôle double. D'un côté, un bruit de faible amplitude peut contribuer à maintenir la flexibilité du système en permettant des fluctuations autour d'un attracteur stable.

D'un autre côté, un bruit persistant, même modéré, peut suffire à faire sortir le système de son bassin d'attraction initial, entraînant ainsi une transition vers une autre configuration attractrice. Cette capacité à basculer entre différents états stables souligne la dualité du rôle du bruit : il favorise à la fois l'exploration de nouvelles configurations et, en excès, il peut perturber la convergence vers un état stable.

Synthèse et Conclusion :

La réaction d'un SCN à l'injection de bruit s'exprime sur un continuum allant de la **résilience** à la **sensibilité**. Lorsque l'amplitude du bruit $\xi_{i,j}(t)$ est faible par rapport à la force corrective $\eta [S(i,j) - \tau \omega_{i,j}(t)]$, le système maintient ses structures auto-organisées malgré des perturbations mineures. En revanche, si le bruit atteint ou dépasse l'ordre de grandeur de ce terme de correction, les fluctuations peuvent altérer significativement les pondérations, conduisant à une désorganisation des clusters. Ainsi, la stabilité globale du SCN dépend de la comparaison précise entre ces deux contributions. Ce phénomène souligne l'importance de paramétrier soigneusement les valeurs de η et τ pour assurer une dynamique robuste, comme évoqué dans les sections 2.3.1.3 et 2.3.2.1. En conclusion, le SCN peut démontrer une forte **résilience** en conservant ses regroupements lorsque le bruit est suffisamment faible, tandis qu'une **sensibilité** accrue, due à un bruit trop important ou à une faible différenciation entre synergies internes et externes, peut empêcher la stabilisation de structures cohérentes.

2.3.4.3. Introduction à l'idée qu'un certain niveau de bruit peut parfois aider l'exploration (analogie avec le recuit simulé)

Dans le cadre du **Synergistic Connection Network (SCN)**, l'injection d'un élément de **bruit** peut, de manière paradoxale, améliorer la capacité du système à explorer l'espace des configurations, en l'a aidant à échapper à des minima locaux qui ne sont pas nécessairement optimaux. Cette idée s'inspire du **recuit simulé** (simulated annealing) en optimisation, où l'on autorise initialement des fluctuations aléatoires significatives afin d'éviter un piégeage prématûr dans des configurations sous-optimales, avant de réduire progressivement l'intensité de ces perturbations pour permettre la stabilisation du système dans un état attracteur de haute qualité.

Considérons, à titre d'exemple, la règle de mise à jour additive fondamentale du SCN qui s'exprime sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le **taux d'apprentissage** et $\tau > 0$ représente le **coefficent de décroissance**. Pour intégrer l'effet de la **perturbation stochastique**, on modifie cette équation en ajoutant un terme aléatoire $\xi_{i,j}(t)$ qui est souvent modélisé par une distribution gaussienne de moyenne nulle et de variance dépendante d'une **température** $T(t)$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t), \quad \xi_{i,j}(t) \sim \mathcal{N}(0, \sigma^2(T(t))).$$

Au début de l'apprentissage, lorsque la **température** $T(0)$ est élevée, l'amplitude de $\xi_{i,j}(t)$ est grande, ce qui permet au réseau d'explorer un vaste espace de configurations. Cette phase d'exploration est essentielle pour éviter que le SCN ne se retrouve bloqué dans un minimum local qui pourrait être sous-optimal du point de vue de la **cohésion interne**. Au fil des itérations, un

refroidissement progressif de $T(t)$ est appliqué, ce qui réduit graduellement la variance $\sigma^2(T(t))$ du bruit. Ainsi, les fluctuations deviennent moins importantes et le système est amené à converger vers un arrangement plus stable, de sorte que, finalement, lorsque $T(t)$ tend vers zéro, l'évolution de $\omega_{i,j}(t)$ est dominée par le terme déterministe $\eta [S(i,j) - \tau \omega_{i,j}(t)]$.

L'analogie avec le **recuit simulé** se trouve dans la stratégie adoptée : initialement, le système autorise des sauts aléatoires – pouvant temporairement augmenter la valeur de la fonction « énergie » ou, dans notre cas, modifier les pondérations de manière non optimale – pour ensuite réduire progressivement ces fluctuations afin de "figer" la configuration dans un minimum qui est supposé être de qualité supérieure. On peut résumer ce mécanisme par l'idée qu'un certain niveau de **bruit** agit comme un moyen d'évasion des minima locaux et, par conséquent, d'**exploration** accrue du paysage de la **synergie**.

Les avantages d'une telle approche résident dans la capacité du système à éviter un piégeage précoce, favorisant ainsi l'émergence de regroupements (clusters) qui présentent une meilleure **cohésion interne**. En revanche, si le bruit persiste à un niveau trop élevé, il risque de perturber la convergence, empêchant le SCN de stabiliser ses liens et, par conséquent, de former des structures cohérentes. Ce compromis met en évidence la nécessité d'un **décroissement progressif** de la température, analogue à la planification de recuit dans le recuit simulé, afin de passer d'une phase d'exploration à une phase de **stabilisation**.

En conclusion, l'injection d'un bruit contrôlé dans un SCN peut être bénéfique en permettant au réseau d'explorer un plus grand nombre de configurations, d'éviter de se figer prématurément dans un minimum local et, finalement, d'aboutir à une structure globale plus optimale. Cette stratégie, inspirée du **recuit simulé**, souligne l'importance de paramétrier adéquatement les niveaux de bruit et leur décroissance (via la température $T(t)$) pour concilier **exploration** et **stabilisation** dans le processus d'auto-organisation du **Deep Synergy Learning**.

2.3.5. Limites Théoriques et Questions Non Résolues

Les sections précédentes (2.3.1 à 2.3.4) ont présenté divers aspects de la **stabilité** dans un SCN, de la formation de clusters et des phénomènes de multi-attracteurs ou d'oscillations. Toutefois, la **théorie** du Deep Synergy Learning (DSL), envisagée comme un *système dynamique*, demeure largement **incomplète**. Il existe un ensemble d'**hypothèses** simplificatrices sous-jacentes et de **questions non résolues** quant à l'extension des modèles (apparition/disparition d'entités, synergie n-aire généralisée, etc.). La section 2.3.5 synthétise ces points, ouvrant la voie aux développements ultérieurs (notamment chapitres 9 et 12).

2.3.5.1. Quelles hypothèses fortes sont faites (ex. synergie binaire, stationnarité, etc.)

Dans le cadre du Deep Synergy Learning, l'analyse théorique repose sur un ensemble d'hypothèses fortes qui permettent de simplifier l'étude de la dynamique du réseau, en particulier pour établir des résultats concernant la convergence, la stabilité et l'émergence de clusters. Ces hypothèses concernent principalement la **stationnarité** de la fonction de synergie, les **bornes** sur les pondérations, la forme de la **règle de mise à jour** ainsi que la nature même de la synergie.

La première hypothèse concerne la **stationnarité** (ou quasi-stationnarité) de la fonction de synergie. En effet, on postule souvent que la fonction $S(i, j)$ ne varie pas de manière brusque ou aléatoire au cours du temps. Autrement dit, il est supposé que $S(i, j)$ peut être considérée comme constante ou, à tout le moins, comme évoluant de manière suffisamment régulière. Cette hypothèse se formalise soit par le fait que $S(i, j)$ dépend uniquement de paramètres invariants, tels que les représentations fixes \mathbf{x}_i et \mathbf{x}_j des entités, soit par la supposition que, lorsqu'une dépendance aux états internes \mathbf{s}_i est prise en compte, la fonction $S(i, j)$ évolue à un rythme plus lent que les mises à jour des pondérations. Formellement, on peut exprimer cette hypothèse en admettant que, pour tout t et pour toute paire (i, j) , on a

$$S(i, j, t) \approx S(i, j),$$

ou bien que la variation temporelle, notée $\Delta S(i, j, t)$, est négligeable par rapport aux variations des poids.

Une autre hypothèse forte est celle concernant la **bornitude** des pondérations. On suppose qu'il existe un intervalle borné, souvent noté $[0, \omega_{\max}]$, dans lequel chaque $\omega_{i,j}(t)$ évolue. Cette contrainte empêche théoriquement la divergence des poids et facilite l'application de théorèmes classiques tels que le théorème de point fixe de Brouwer. On exprime cette hypothèse par l'inégalité

$$0 \leq \omega_{i,j}(t) \leq \omega_{\max}, \quad \forall i, j, \forall t,$$

ce qui permet également de considérer l'espace des configurations comme compact et convexe, une condition essentielle pour la convergence.

Une troisième hypothèse réside dans la **forme de mise à jour** des pondérations. Dans de nombreux travaux sur le DSL, la règle de mise à jour est adoptée sous une forme additive, généralement donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le taux d'apprentissage et $\tau > 0$ est le coefficient de décroissance. Cette forme linéaire simplifie l'analyse, notamment la linéarisation locale autour d'un point fixe, permettant de montrer que, dans le cas stationnaire, le point fixe pour une liaison donnée est

$$\omega_{i,j}^* = \frac{S(i, j)}{\tau}.$$

L'hypothèse d'une mise à jour additive permet ainsi d'établir, par linéarisation, des conditions de stabilité via l'étude de la matrice Jacobienne. Cependant, il convient de noter que dans des versions plus avancées, des mises à jour de nature multiplicative ou intégrant des couplages n-aires apparaissent, ce qui rend la dynamique plus complexe et nécessite de relâcher cette hypothèse de linéarité.

Enfin, certaines simplifications interviennent dans la **nature** même de la synergie. Pour rendre l'analyse plus tractable, il est parfois supposé que $S(i, j)$ est de nature **binaire**, c'est-à-dire que $S(i, j) \in \{0, 1\}$. Une telle hypothèse permet de transformer la mesure continue de synergie en un indicateur discret qui détermine simplement si une connexion entre deux entités est présente ou non. Bien que cette approche puisse être très utile pour certaines applications (notamment lorsque

l'on souhaite obtenir des structures denses et clairement définies), elle perd en finesse et en nuance, puisque la gradation de la synergie est alors remplacée par un seuil abrupt.

En synthèse, l'ensemble de ces hypothèses – la stationnarité (ou quasi-stationnarité) de $S(i,j)$, la contrainte de bornitude des poids dans $[0, \omega_{\max}]$, l'adoption d'une règle de mise à jour additive, ainsi que la simplification de $S(i,j)$ en une variable binaire ou en une mesure continue régularisée – constitue un cadre restreint mais opératoire pour l'analyse théorique du DSL. Ces hypothèses facilitent l'étude de la convergence, la détection de points fixes, ainsi que l'identification des clusters, en rendant l'opérateur dynamique F suffisamment « maniable » pour l'application de méthodes analytiques classiques. Toutefois, il est important de noter que dans des applications réelles, où les entités peuvent apparaître et disparaître, ou lorsque des interactions plus complexes (comme la synergie n-aire) sont à l'œuvre, il sera nécessaire de relâcher certaines de ces hypothèses pour capter toute la richesse du comportement du système. Les sections suivantes, notamment **2.3.5.2** et **2.3.5.3**, aborderont ces ouvertures et exploreront des généralisations qui permettront de modéliser des phénomènes plus complexes tout en gardant une base théorique solide.

2.3.5.2. Ouvertures vers une gestion plus dynamique (entités qui apparaissent ou disparaissent)

Dans le cadre du **Deep Synergy Learning**, il est fondamental de considérer des scénarios où le nombre d'**entités** n'est pas fixe, mais évolue au fil du temps. Cette ouverture vers une gestion dynamique des entités reflète la réalité de nombreux systèmes complexes, tels que les réseaux de capteurs, les plateformes sociales ou les systèmes de robotique coopérative, dans lesquels de nouveaux éléments peuvent apparaître tandis que d'autres disparaissent ou deviennent inactifs. Cette dimension temporelle supplémentaire exige une extension du modèle de mise à jour des **pondérations** $\{\omega_{i,j}\}$ de sorte à permettre l'adaptation du **Synergistic Connection Network (SCN)\$ face à l'évolution du nombre d'unités.

A. Motivations et enjeux

Dans de nombreux domaines, la dynamique du système ne se limite pas à l'évolution des **liaisons** entre entités statiques, mais doit également intégrer la **naissance** et la **disparition** d'unités. Par exemple, dans un réseau de capteurs IoT, l'ajout de nouveaux dispositifs ou la défaillance de certains capteurs modifie la structure globale du réseau, de même qu'un réseau social, soumis à des flux continus d'inscriptions et de désinscriptions, présente une topologie en constante évolution. En robotique coopérative, la flotte de robots peut s'agrandir ou se réduire selon les besoins opérationnels ou les incidents techniques. Pour que le SCN puisse modéliser de telles situations, il faut autoriser la structure de **pondérations** à évoluer non seulement en réponse aux fluctuations de la **synergie** $S(i,j)$, mais également en fonction du nombre d'**entités** présentes.

B. Approche mathématique pour la gestion dynamique

Si l'on note $\Omega(t)$ l'ensemble des pondérations pour un réseau comportant initialement n entités, alors $\Omega(t)$ est un vecteur dont les composantes sont données par

$$\Omega(t) = (\omega_{i,j}(t) \mid i,j \in \{1, \dots, n\}).$$

Lorsqu'une nouvelle entité, que l'on notera \mathcal{E}_{n+1} , rejoint le réseau, l'espace d'état des pondérations s'étend naturellement. En effet, il devient nécessaire d'introduire de nouvelles variables, par exemple $\omega_{n+1,j}(t)$ et $\omega_{j,n+1}(t)$ pour $j = 1, \dots, n$, afin de décrire les interactions entre la nouvelle entité et celles déjà existantes. L'opérateur de mise à jour, qui était auparavant défini sur l'espace \mathcal{X}_n , doit être adapté pour opérer sur \mathcal{X}_{n+1} . Une telle extension se traduit par l'initialisation de ces nouvelles pondérations, par exemple par

$$\omega_{n+1,j}(0) \approx 0 \quad \text{et} \quad \omega_{j,n+1}(0) \approx 0,$$

afin de laisser à la dynamique du SCN le temps de déterminer, via la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

la pertinence des liaisons entre la nouvelle entité et le reste du réseau. Inversement, lorsque qu'une entité \mathcal{E}_k disparaît ou devient inactif, on suppose que l'on met à zéro les pondérations correspondantes, c'est-à-dire que pour tout j , on fixe

$$\omega_{k,j}(t) = \omega_{j,k}(t) = 0,$$

et l'espace des pondérations se contracte de manière appropriée de \mathcal{X}_n vers \mathcal{X}_{n-1} . Cette procédure de retrait impose un rééquilibrage du réseau, car les liens qui dépendaient de l'entité disparue devront être redistribués parmi les entités restantes, pouvant entraîner une réorganisation de la structure des **clusters**.

C. Conséquences sur la stabilité et la formation de clusters

L'arrivée ou le départ d'**entités** constitue une perturbation dans la dynamique globale du SCN. Si, par exemple, une nouvelle entité \mathcal{E}_{n+1} manifeste une forte synergie avec certains sous-groupes existants, cela peut conduire à une reconfiguration des clusters. Une partie du cluster initial peut se scinder pour intégrer la nouvelle entité, ou bien les pondérations existantes peuvent être ajustées pour optimiser la cohésion globale. Dans le cas d'une entité qui se retire, l'équilibre du réseau est modifié : le cluster auquel elle appartenait peut se morceler ou fusionner avec un autre, selon l'importance de sa contribution initiale au niveau des **synergies** internes.

D'un point de vue algorithmique, deux stratégies principales peuvent être envisagées pour gérer ces fluctuations de dimension. La première consiste à adopter une approche de **continuité stricte**, dans laquelle l'insertion ou le retrait d'une entité s'effectue de manière incrémentale, permettant aux pondérations existantes de se réajuster progressivement sans nécessiter une réinitialisation complète du système. La seconde stratégie, appelée **réinitialisation partielle**, consiste à réinitialiser partiellement certaines pondérations lorsque le changement de dimension est jugé majeur, afin de faciliter la recomposition de la dynamique du réseau.

D. Exemples pratiques et implications

Considérons un scénario typique dans un réseau de capteurs où de nouveaux dispositifs sont ajoutés régulièrement. Dès l'insertion d'une nouvelle entité \mathcal{E}_{n+1} , les nouvelles pondérations $\omega_{n+1,j}$ sont initialisées à une valeur proche de zéro. La règle de mise à jour,

$$\omega_{n+1,j}(t+1) = \omega_{n+1,j}(t) + \eta [S(n+1,j) - \tau \omega_{n+1,j}(t)],$$

détermine ensuite la manière dont \mathcal{E}_{n+1} se connecte aux entités existantes, ce qui peut entraîner la formation de nouveaux clusters ou la fusion de clusters existants pour mieux intégrer la nouvelle information. Inversement, dans un réseau social, la disparition d'un utilisateur \mathcal{E}_k implique la mise à zéro de toutes les pondérations associées, ce qui peut provoquer une reconfiguration du cluster auquel il appartenait, modifiant ainsi la partition finale du réseau.

E. Limites théoriques et perspectives

L'ouverture vers une gestion dynamique des entités pose cependant plusieurs défis théoriques. En effet, lorsque la dimension de l'espace des pondérations n'est plus fixe, les outils classiques de la théorie des points fixes, qui reposent sur la compacité et la convexité de cet espace, deviennent partiellement inopérants. Le passage de \mathcal{X}_n à \mathcal{X}_{n+1} ou \mathcal{X}_{n-1} entraîne une modification de la structure même de l'opérateur F , ce qui nécessite le développement de méthodes incrémentales pour analyser la stabilité du système en évolution. De plus, l'inflation du nombre de liaisons, qui croît typiquement en $O(n)$ pour chaque nouvelle entité, peut entraîner une augmentation significative des coûts computationnels, ce qui impose souvent l'emploi de techniques de **sparsification** pour garantir la praticabilité du modèle.

Ces défis ouvrent des perspectives intéressantes pour le développement du **Deep Synergy Learning**. Des recherches futures pourraient explorer des stratégies de réinitialisation partielle ou des méthodes adaptatives de recalibrage des pondérations afin de conserver un équilibre entre la stabilité du réseau et sa capacité à s'adapter aux évolutions du nombre d'entités. Des travaux ultérieurs, notamment ceux qui seront abordés dans le Chapitre 9, se pencheront sur ces questions en profondeur, intégrant des approches de **mise à jour incrémentale** et des algorithmes d'**apprentissage continu** qui tiennent compte de l'apparition et de la disparition des entités.

Conclusion (2.3.5.2)

L'ouverture vers une gestion dynamique des entités, où le SCN peut croître avec l'arrivée de nouvelles unités ou se réduire en cas de disparition, constitue une extension essentielle du modèle de **Deep Synergy Learning**. Cette gestion implique une adaptation de la structure des **pondérations** $\{\omega_{i,j}\}$ par l'élargissement ou la contraction de l'espace d'état, ce qui modifie l'opérateur de mise à jour F . Bien que cette approche rende l'analyse théorique plus complexe – notamment en remettant en question les hypothèses classiques de compacité et de stabilité dans un espace de dimension fixe – elle reflète de manière réaliste les dynamiques de systèmes évolutifs. Les stratégies envisagées, telles que l'insertion incrémentale ou la réinitialisation partielle, offrent des pistes prometteuses pour conserver la **stabilité** et la **cohérence** des clusters malgré la fluctuation du nombre d'entités. Ces perspectives seront approfondies dans les chapitres suivants, notamment dans le contexte de l'apprentissage continu et de la synergie multimodale, afin d'étendre le cadre théorique du DSL à des systèmes véritablement dynamiques.

2.3.5.3. Préfigure les Chapitres 9 et 12 sur l'apprentissage continu et la synergie n-aire

Dans les sections précédentes, notamment en **2.3.5.1** et **2.3.5.2**, nous avons établi un cadre théorique fondé sur des hypothèses fortes concernant la stationnarité de la **synergie** $S(i,j)$ ainsi

que sur la gestion dynamique des entités dans un **Synergistic Connection Network** (SCN). Il apparaît désormais indispensable d'envisager deux axes majeurs d'extension du modèle, axes qui seront développés de manière approfondie dans les **Chapitre 9** et **Chapitre 12** de cet ouvrage. La première orientation consiste en l'**apprentissage continu** (ou *lifelong learning*), tandis que la seconde porte sur l'extension de la synergie aux interactions **n-aires**. Ces deux axes permettent d'élargir le domaine d'application du DSL et de prendre en compte des scénarios où la dynamique des entités et la complexité des interactions dépassent le cadre strictement binaire et stationnaire.

Dans le contexte de l'**apprentissage continu**, le SCN se trouve confronté à des environnements évolutifs, où de nouvelles données ou de nouvelles entités apparaissent, tandis que d'autres disparaissent ou deviennent inactives. Au lieu de supposer un nombre fixe d'entités – hypothèse qui permettait d'appliquer des théorèmes de convergence dans un espace compact et convexe – il faut maintenant envisager une extension de la dynamique du réseau. Plus précisément, si l'on note initialement par

$$\Omega(t) = (\omega_{1,2}(t), \omega_{1,3}(t), \dots, \omega_{n-1,n}(t))$$

la configuration des pondérations pour un ensemble de n entités, l'arrivée d'une nouvelle entité \mathcal{E}_{n+1} entraîne l'extension de l'espace d'état à

$$\Omega'(t) = (\omega_{1,2}(t), \dots, \omega_{n-1,n}(t), \omega_{n+1,1}(t), \omega_{n+1,2}(t), \dots, \omega_{n+1,n}(t)).$$

La règle de mise à jour demeure alors de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

mais l'opérateur F doit désormais être défini sur un espace de dimension variable. Ce processus d'intégration progressive de nouvelles entités nécessite des mécanismes spécifiques pour éviter le phénomène de « retrait catastrophique » (catastrophic forgetting) et pour permettre une adaptation continue sans réinitialiser l'ensemble du réseau. C'est précisément l'objet du **Chapitre 9**, qui développera des stratégies d'apprentissage incrémental permettant de concilier la préservation des connaissances acquises avec l'adaptation à de nouveaux contextes, ainsi que des ajustements paramétriques dynamiques de η et τ pour optimiser la stabilité dans un environnement évolutif.

En parallèle, la généralisation de la synergie aux interactions **n-aires** représente une deuxième extension fondamentale. Jusqu'ici, le DSL s'est principalement appuyé sur une fonction de synergie définie pour des paires d'entités, c'est-à-dire $S(i,j)$. Or, de nombreuses applications requièrent de modéliser des interactions collectives impliquant trois, quatre ou un nombre plus grand d'entités simultanément. Dans ce cadre, il convient d'introduire une fonction générale

$$S(i_1, i_2, \dots, i_k),$$

qui mesure la **co-information** ou la **synergie collective** d'un groupe d'entités $\{\mathcal{E}_{i_1}, \mathcal{E}_{i_2}, \dots, \mathcal{E}_{i_k}\}$. Cette généralisation conduit naturellement au passage d'un graphe pondéré à un **hypergraphe**, dans lequel les hyper-liens représentent des interactions entre plusieurs nœuds de manière intrinsèquement non binaire. Formellement, il devient alors pertinent de définir des pondérations telles que

$$\omega_{(i_1, i_2, \dots, i_k)}(t+1) = \omega_{(i_1, i_2, \dots, i_k)}(t) + \eta [S(i_1, i_2, \dots, i_k) - \tau \omega_{(i_1, i_2, \dots, i_k)}(t)],$$

ce qui élargit considérablement le champ d'application du DSL. Le **Chapitre 12** se consacrera à formaliser ce concept de synergie **n-aire** et à étudier l'impact de ces interactions multiples sur l'auto-organisation et la stabilité globale du réseau. L'analyse portera notamment sur la manière dont ces hyper-liens influencent la formation de clusters et sur les nouveaux phénomènes dynamiques qui émergent dans un hypergraphe, en intégrant des aspects de co-information collective et d'interactions non linéaires plus sophistiquées.

En résumé, l'intégration de l'**apprentissage continu** et de la **synergie n-aire** constitue une avancée majeure qui préfigure un DSL de nouvelle génération, capable de s'adapter de manière incrémentale aux évolutions de son environnement tout en prenant en compte des interactions multi-entités. Ces deux axes d'extension ouvrent des perspectives particulièrement intéressantes pour des applications dans des domaines tels que les réseaux de capteurs, la robotique multi-agents ou encore les systèmes neuronaux inspirés de la biologie, où la flexibilité et la complexité des interactions sont essentielles. Les **Chapitre 9** et **Chapitre 12** développeront ces idées en profondeur, en proposant des modèles mathématiques rigoureux ainsi que des approches algorithmiques adaptées pour garantir la **stabilité** et la **cohérence** dans des environnements dynamiques et multidimensionnels.

Ainsi, cette section sert de pont vers une compréhension plus large du DSL, en préparant le terrain pour une gestion avancée de l'**apprentissage continu** et l'extension de la **synergie** au-delà du cadre binaire traditionnel, contribuant à l'émergence de réseaux véritablement **adaptatifs** et **complexes**.

2.4. Raccords avec la Physique Statistique et la Théorie des Systèmes Dynamiques

Le **Deep Synergy Learning** (DSL), vu sous l'angle d'un **réseau** $\{\omega_{i,j}\}$ évoluant sous des règles non linéaires, peut être relié à des approches classiques de la **physique statistique** (modèles de spins, énergies) et de la **théorie des systèmes dynamiques** (discrets ou continus). La section 2.4 initie ce rapprochement, en commençant (2.4.1) par l'interprétation de l'équation de mise à jour $\omega_{i,j}(t+1)$ comme un **système dynamique discret** et ses variantes continues, puis (2.4.2) et (2.4.3) feront des parallèles explicites avec les modèles de spin et la notion d'énergie ou de fonction potentielle.

2.4.1. Systèmes Dynamiques Discrets ou Continus

On a souvent présenté la mise à jour du SCN (voir 2.2.2) sous forme discrète :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j)\tau \omega_{i,j}(t)],$$

mais il est tout à fait possible de concevoir une version **continue** (2.4.1.2) ou même de tenter des analogies directes avec les **systèmes dynamiques** utilisés en physique statistique (2.4.2). Commençons (2.4.1.1) par voir en quoi $\omega_{i,j}(t+1) = F(\omega_{i,j}(t), \dots)$ s'interprète déjà comme un **système dynamique discret** et comment les outils de cette théorie peuvent s'appliquer.

2.4.1.1. Équation de mise à jour $\omega_{i,j}(t+1)$ vue comme un système dynamique discréte

Dans la théorie des **systèmes dynamiques**, un modèle discret se représente typiquement par l'équation

$$\mathbf{x}(t+1) = F(\mathbf{x}(t)), \quad \mathbf{x} \in \mathbb{R}^D,$$

où chaque itération convertit l'état $\mathbf{x}(t)$ en $\mathbf{x}(t+1)$ au moyen d'une **application** (ou **opérateur**) F . Dans le cadre du **Deep Synergy Learning (DSL)**, l'état du réseau est encapsulé dans le vecteur $\Omega(t)$ qui regroupe l'ensemble des **pondérations** $\{\omega_{i,j}(t)\}$ et, éventuellement, d'autres variables d'état telles que les **états internes** des entités. On peut donc écrire

$$\Omega(t+1) = F(\Omega(t)),$$

où l'opérateur F intègre la **règle de mise à jour** du DSL, que l'on exprime généralement sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j; \Omega(t)) - \tau \omega_{i,j}(t)].$$

Dans cette équation, $S(i,j; \Omega(t))$ représente la **synergie effective** évaluée à l'itération t et peut dépendre non seulement des représentations intrinsèques des entités, telles que leurs **vecteurs caractéristiques** \mathbf{x}_i ou leurs **états internes** \mathbf{s}_i , mais aussi de l'ensemble des pondérations $\omega_{p,q}(t)$ existant dans le réseau. Le paramètre $\eta > 0$ est le **taux d'apprentissage** qui détermine la vitesse d'évolution du système, tandis que $\tau > 0$ représente le **coefficent de décroissance**, essentiel pour éviter une croissance non bornée des pondérations et pour assurer la régulation du processus.

Si l'on considère un réseau composé de n entités et que les liens sont orientés, le nombre total de pondérations possibles est de l'ordre de $n(n - 1)$; par conséquent, le vecteur d'état $\Omega(t)$ appartient à un espace de dimension $D \approx n(n - 1)$. L'opérateur F agit donc sur un espace de grande dimension, et sa définition se fait de manière à ce que, pour toute condition initiale $\Omega(0)$ appartenant à un ensemble borné $\mathcal{X} \subset \mathbb{R}^D$ (par exemple, chaque $\omega_{i,j}$ étant contraint à l'intervalle $[0, \omega_{\max}]$), la suite $\{\Omega(t)\}$ demeure dans \mathcal{X} . Autrement dit, la dynamique se décrit par

$$\Omega(t + 1) = F(\Omega(t)), \quad \text{avec } \Omega(t) \in \mathcal{X},$$

et l'**orbite** ou la **trajectoire** du système est définie par l'ensemble $\{\Omega(0), \Omega(1), \Omega(2), \dots\}$.

L'intérêt de cette formalisation réside dans la capacité à analyser, du point de vue de la **stabilité** et de la **convergence**, la dynamique des poids. En effet, comme indiqué dans la section 2.3.1.3, la stabilité locale d'un point fixe Ω^* repose sur l'étude de la linéarisation de F autour de ce point. Lorsque la norme de la **matrice Jacobienne** $DF(\Omega^*)$ est strictement inférieure à 1, c'est-à-dire

$$\|DF(\Omega^*)\| < 1,$$

le point fixe Ω^* se révèle stable ; sinon, des phénomènes tels que des cycles, des multi-attracteurs ou même des comportements pseudo-chaotiques peuvent apparaître (voir également la discussion des sections 2.3.2 et 2.3.3).

Une illustration théorique simple de la dynamique consiste à considérer le cas stationnaire où $S(i, j; \Omega(t))$ est approximativement constant. Dans ce cas, la règle de mise à jour se simplifie en

$$\omega_{i,j}(t + 1) = (1 - \eta \tau) \omega_{i,j}(t) + \eta S(i, j),$$

ce qui, sous l'hypothèse de stabilité, conduit à la convergence des poids vers le point fixe

$$\omega_{i,j}^* = \frac{S(i, j)}{\tau}.$$

Cette condition de stabilité, qui se traduit par l'inégalité

$$|1 - \eta \tau| < 1,$$

implique que, dans un cas simple, le produit $\eta \tau$ doit satisfaire $0 < \eta \tau < 2$. Les **paramètres** η et τ sont donc cruciaux pour la bonne marche du système : un η trop élevé peut induire des oscillations, tandis qu'un τ trop grand risque d'amener les poids à s'écraser, limitant ainsi l'amplitude des interactions.

Par analogie avec des modèles de **spins** tels que le modèle d'**Ising** ou le réseau de **Hopfield**, qui sont souvent mis à jour de manière itérative (par exemple, via l'algorithme Metropolis-Hastings), le DSL peut être vu comme un système dynamique discret opérant sur le vecteur $(\omega_{1,2}, \omega_{1,3}, \dots, \omega_{n-1,n})$. Cette analogie s'étendra dans les sections 2.4.2 et 2.4.3, où seront établis des liens supplémentaires avec la **fonction d'énergie** et les transitions de phase, offrant ainsi une vision énergétique de la dynamique du DSL.

Pour formaliser cette approche, nous pouvons énoncer le théorème suivant :

Théorème (Existence et Itération de la Dynamique Discrète)

Soit $\mathcal{X} \subset \mathbb{R}^D$ un ensemble **compact** (par exemple, chaque $\omega_{i,j}$ étant borné dans $[0, \omega_{\max}]$). Soit $F: \mathcal{X} \rightarrow \mathcal{X}$ une application **continue** qui définit la mise à jour du vecteur d'état selon

$$\Omega(t+1) = F(\Omega(t)).$$

Alors, pour toute condition initiale $\Omega(0) \in \mathcal{X}$, la suite $\{\Omega(t)\}_{t \geq 0}$ reste dans \mathcal{X} et, par le **théorème de Brouwer** (ou Schauder), il existe au moins un **point fixe** $\Omega^* \in \mathcal{X}$ tel que

$$\Omega^* = F(\Omega^*).$$

La stabilité ou la multiplicité de ce point fixe dépendra des propriétés locales de l'opérateur F , notamment de la norme $\|DF(\Omega^*)\|$.

En pratique, le fait de **borner** les valeurs de $\omega_{i,j}$ (par clipping ou par l'application de règles de **parsimonie**) et d'adopter une règle de mise à jour continue assure que la trajectoire $\Omega(0) \rightarrow \Omega(1) \rightarrow \Omega(2) \rightarrow \dots$ reste bien définie dans l'espace \mathcal{X} . Ainsi, la dynamique du DSL s'inscrit pleinement dans le cadre standard des systèmes dynamiques discrets, et la stabilité des clusters ainsi formés dépend essentiellement des propriétés de F .

Conclusion

L'équation de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j; \Omega(t)) - \tau \omega_{i,j}(t)],$$

se lit comme une représentation typique d'un **système dynamique discret** opérant sur le vecteur $\Omega(t)$ dans un espace de dimension $D \approx n(n-1)$. L'opérateur $F: \mathcal{X} \rightarrow \mathcal{X}$ qui en résulte, supposé continu et défini sur un ensemble compact (par exemple, par des bornes imposées aux $\omega_{i,j}$), garantit par le théorème de Brouwer l'existence d'au moins un point fixe. La stabilité locale de ce point fixe se mesure par l'analyse de la **matrice Jacobienne** $DF(\Omega^*)$ et l'exigence que $\|DF(\Omega^*)\| < 1$. Par ailleurs, l'analogie avec les modèles de spins (Ising, Hopfield) et l'approche énergétique permet de mieux comprendre les transitions de phase et les dynamiques complexes observées dans le DSL. Dans la section suivante (2.4.1.2), nous aborderons la variante continue de ce système, en formulant une équation différentielle qui offre un nouveau pont entre le DSL et la modélisation des systèmes hors équilibre en physique et en biologie.

2.4.1.2. Variante continue : $\frac{d\omega_{ij}}{dt} = \eta [S(i,j) - \tau \omega_{ij}]$

Dans la plupart des présentations du **Deep Synergy Learning (DSL)**, la mise à jour des pondérations se réalise en temps discret par des itérations successives, ce qui conduit à une équation de la forme

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)].$$

Cependant, il est tout à fait envisageable de modéliser l'évolution des pondérations par une **équation différentielle** en temps continu. Pour ce faire, on part de la différence discrète exprimée par

$$\frac{\omega_{ij}(t+1) - \omega_{ij}(t)}{\Delta t} = \eta [S(i,j) - \tau \omega_{ij}(t)],$$

et, en posant $\Delta t = 1$ puis en faisant tendre Δt vers zéro, la limite conduit à l'équation différentielle

$$\frac{d \omega_{ij}}{dt} = \eta [S(i,j) - \tau \omega_{ij}(t)].$$

Cette formulation permet d'envisager la dynamique des pondérations de manière « fluide » plutôt que par paliers discrets. Dans le cas où la **synergie** $S(i,j)$ est considérée comme constante (ou stationnaire), l'équation différentielle devient linéaire du premier ordre :

$$\frac{d \omega_{ij}}{dt} = \eta [S(i,j) - \tau \omega_{ij}(t)].$$

Pour une condition initiale $\omega_{ij}(0) = \omega_0$, la solution générale s'obtient par intégration et s'écrit ainsi :

$$\omega_{ij}(t) = \frac{S(i,j)}{\tau} + \left(\omega_0 - \frac{S(i,j)}{\tau} \right) \exp(-\eta \tau t).$$

Cette solution démontre que, lorsque $t \rightarrow +\infty$, la pondération converge de manière exponentielle vers le point fixe

$$\omega_{ij}^* = \frac{S(i,j)}{\tau},$$

la vitesse de convergence étant régie par le produit $\eta \tau$. Ce résultat rejoint exactement la conclusion obtenue dans le modèle discret, tout en offrant une interprétation continue et en ouvrant la voie à l'utilisation d'outils analytiques propres aux **équations différentielles ordinaires (EDO)**.

Lorsque la **synergie** $S(i,j)$ dépend non seulement de \mathbf{x}_i et \mathbf{x}_j mais également d'autres pondérations ou d'états internes (par exemple, lorsque S devient une fonction de l'ensemble $\{\omega_{p,q}\}$ ou des états \mathbf{s}_k), le système se généralise à un ensemble couplé d'équations différentielles :

$$\frac{d \omega_{ij}}{dt} = \eta [S(i,j; \{\omega_{p,q}\}) - \tau \omega_{ij}(t)], \quad \forall (i,j).$$

Le vecteur global $\boldsymbol{\Omega}(t)$, constitué de l'ensemble des pondérations, évolue alors dans un espace de dimension potentiellement élevée, souvent de l'ordre de $n(n-1)$ si les liens sont orientés. Les propriétés de **continuité** et de **Lipschitz** de la fonction S garantissent, par le théorème de **Cauchy-Lipschitz**, l'existence et l'unicité locale des solutions. La stabilité d'un point fixe $\boldsymbol{\Omega}^*$ dans ce contexte se caractérise par l'étude de la linéarisation de l'opérateur F associé à la dynamique continue, qui se traduit par

$$\delta\Omega'(t) = DF(\Omega^*) \delta\Omega(t),$$

où $DF(\Omega^*)$ désigne la **matrice Jacobienne** évaluée en l'équilibre. Si toutes les valeurs propres de cette Jacobienne ont des parties réelles strictement négatives, l'équilibre est localement attractif.

Il convient de noter que, dans la pratique, bien que le DSL soit souvent implémenté en temps discret avec un pas Δt très petit (par exemple, entre 0.01 et 0.1), l'analyse en temps continu offre un cadre conceptuel riche, permettant d'établir des analogies avec des modèles de **spins** et d'autres systèmes neuronaux (comme ceux de **Hopfield** ou de **Cohen-Grossberg**). Ce passage à une formulation continue rapproche ainsi le DSL des modèles de systèmes dynamiques continus rencontrés en physique et en biologie, enrichissant l'analyse théorique et facilitant l'étude des propriétés de convergence et de stabilité.

En somme, la variante continue du DSL s'exprime par

$$\frac{d\omega_{ij}}{dt} = \eta [S(i,j) - \tau \omega_{ij}],$$

ce qui, dans le cas stationnaire, implique que $\omega_{ij}(t)$ converge exponentiellement vers le point fixe $\omega_{ij}^* = \frac{S(i,j)}{\tau}$. Dans le cas général, où $S(i,j)$ dépend des pondérations ou des états internes, le système se complexifie en un ensemble couplé d'équations différentielles dont l'analyse repose sur des techniques classiques de linéarisation et d'étude de la stabilité par la Jacobienne. Cette formulation continue offre une perspective alternative et complémentaire à la dynamique discrète, et permet de mieux comprendre les analogies entre le DSL et les modèles neuronaux ou les systèmes de spins étudiés en physique mathématique.

2.4.1.3. Conditions d'existence d'une solution fermée ou d'une approximation

Dans la version continue du **Deep Synergy Learning (DSL)**, la dynamique des pondérations $\omega_{ij}(t)$ est régie par un système d'équations différentielles ordinaires (EDO) généralement écrit sous la forme

$$\frac{d\omega_{ij}}{dt} = \eta [S(i,j; \{\omega_{p,q}\}, \dots) - \tau \omega_{ij}(t)],$$

où $\eta > 0$ représente le **taux d'apprentissage**, $\tau > 0$ le **coefficent de décroissance** et $S(i,j; \{\omega_{p,q}\}, \dots)$ décrit la **synergie effective** qui peut, selon les cas, dépendre de l'ensemble des pondérations $\{\omega_{p,q}\}$ et d'autres variables (par exemple, les états internes des entités). L'objectif est de déterminer, d'une part, dans quels cas on peut obtenir une **solution fermée** (analytique) pour ces équations et, d'autre part, quelles méthodes théoriques et numériques permettent de prouver l'**existence**, l'**unicité** et la **qualité** des solutions lorsque la solution fermée n'est pas accessible.

A. Solutions fermées dans les cas simples

Lorsque la fonction de synergie $S(i,j)$ est supposée **constante** ou **indépendante** de l'ensemble des pondérations $\{\omega_{p,q}\}$, l'équation se simplifie en une équation différentielle linéaire du premier ordre :

$$\frac{d\omega_{ij}}{dt} = \eta [S(i,j) - \tau \omega_{ij}(t)].$$

Cette équation est classique et se résout par séparation des variables ou par la méthode de variation des constantes. En imposant la condition initiale $\omega_{ij}(0) = \omega_0$, la solution analytique se trouve être

$$\omega_{ij}(t) = \frac{S(i,j)}{\tau} + \left(\omega_0 - \frac{S(i,j)}{\tau} \right) \exp(-\eta \tau t).$$

On constate ainsi que, pour $t \rightarrow +\infty$, la pondération converge exponentiellement vers le point fixe

$$\omega_{ij}^* = \frac{S(i,j)}{\tau},$$

ce qui confirme la validité de cette solution fermée dans le cas linéaire stationnaire. De même, si l'on suppose que $S(i,j)$ dépend de manière linéaire des pondérations et reste stationnaire, le système se réduit à un système linéaire de la forme

$$\mathbf{x}'(t) = M \mathbf{x}(t) + \mathbf{b},$$

où l'on peut alors résoudre le système par diagonalisation de la matrice M . Ces cas simples illustrent que, sous certaines hypothèses restrictives, une **solution fermée** est effectivement accessible.

B. Cas non linéaire : S dépendant de $\{\omega_{p,q}\}$

Dans la plupart des scénarios pratiques du DSL, la synergie $S(i,j; \{\omega_{p,q}(t)\}, \dots)$ intègre des **non-linéarités** dues à des fonctions de distance, des mesures de co-information ou à des mécanismes d'inhibition compétitive. Dans ces cas, l'équation devient un système couplé non linéaire :

$$\frac{d\omega_{ij}}{dt} = \eta [S(i,j; \{\omega_{p,q}(t)\}) - \tau \omega_{ij}(t)], \quad \forall(i,j).$$

Ce système, que l'on peut écrire sous la forme

$$\boldsymbol{\Omega}'(t) = \mathbf{F}(\boldsymbol{\Omega}(t)),$$

avec $\boldsymbol{\Omega}(t) \in \mathbb{R}^D$ et $D \approx n(n - 1)$ dans le cas d'un réseau orienté, ne permet généralement pas d'obtenir une solution fermée. Cependant, si la fonction \mathbf{F} est **localement Lipschitz**, le théorème de **Cauchy-Lipschitz** (ou théorème d'existence et d'unicité) garantit qu'une solution unique existe pour chaque condition initiale $\boldsymbol{\Omega}(0)$. En pratique, l'intégration de telles équations non linéaires requiert le recours à des méthodes d'**approximation numérique** ou à des techniques d'analyse qualitative telles que la linéarisation autour d'un point fixe.

C. Méthodes numériques et linéarisation

Pour étudier ou simuler la trajectoire de $\omega_{ij}(t)$ dans le cas non linéaire, plusieurs schémas numériques peuvent être employés. Par exemple, la méthode d'**Euler** consiste à approximer, pour un pas de temps δt suffisamment petit,

$$\omega_{ij}(t + \delta t) \approx \omega_{ij}(t) + \delta t \eta [S(i,j) - \tau \omega_{ij}(t)],$$

ce qui rapproche cette méthode de la version discrète étudiée en section 2.4.1.1. Des méthodes plus sophistiquées, telles que les schémas de **Runge–Kutta** d'ordre 2 ou 4, offrent une meilleure stabilité numérique et une précision accrue en évaluant l'opérateur \mathbf{F} en plusieurs points intermédiaires.

Par ailleurs, une technique d'analyse qualitative repose sur la **linéarisation** du système autour d'un **point fixe** Ω^* . Si l'on note la perturbation $\delta\Omega(t) = \Omega(t) - \Omega^*$, on a

$$\frac{d}{dt} \delta\Omega(t) = \mathbf{DF}(\Omega^*) \delta\Omega(t),$$

où $\mathbf{DF}(\Omega^*)$ est la **matrice Jacobienne** évaluée au point fixe. L'analyse des valeurs propres de cette Jacobienne permet de déterminer la **stabilité locale** : si toutes les valeurs propres ont une partie réelle négative, Ω^* est un attracteur local stable ; sinon, des comportements tels que des cycles ou du pseudo-chaos peuvent apparaître.

D. Bornage et évitemment d'explosions

Il est fréquent, pour éviter la divergence des solutions, d'imposer une condition de **bornitude** sur les pondérations, par exemple en contrignant

$$\omega_{ij}(t) \in [0, \omega_{\max}],$$

ce qui fait vivre $\Omega(t)$ dans un ensemble $\mathcal{X} \subset \mathbb{R}^D$ **compact**. Si l'opérateur \mathbf{F} est continu et renvoie \mathcal{X} dans \mathcal{X} , on obtient alors une existence globale de la solution, et les théorèmes de point fixe (tels que le théorème de Brouwer ou celui de Schauder) assurent l'existence d'au moins un attracteur dans l'espace des pondérations. Même en l'absence de bornes strictes, le terme de décroissance $-\tau \omega_{ij}(t)$ joue un rôle régulateur, empêchant une croissance incontrôlée des pondérations, à condition que la synergie $S(i, j)$ ne soit pas pathologiquement infinie.

E. Conclusion et synthèse

En résumé, la dynamique continue du DSL est gouvernée par l'équation

$$\frac{d \omega_{ij}}{dt} = \eta [S(i, j; \{\omega_{p,q}\}) - \tau \omega_{ij}(t)],$$

qui, dans le cas le plus simple où $S(i, j)$ est constant, admet une solution fermée donnée par

$$\omega_{ij}(t) = \frac{S(i, j)}{\tau} + \left(\omega_{ij}(0) - \frac{S(i, j)}{\tau} \right) \exp(-\eta \tau t).$$

Cette solution montre une convergence exponentielle vers le point fixe

$$\omega_{ij}^* = \frac{S(i, j)}{\tau}.$$

Cependant, lorsque $S(i, j)$ dépend des pondérations elles-mêmes ou d'autres états internes, le système devient non linéaire et couplé. Dans ce cas, bien que l'existence et l'unicité de la solution soient garanties par la condition de **local Lipschitz** sur \mathbf{F} (selon le théorème de Cauchy–Lipschitz), il n'existe généralement pas de solution fermée et l'analyse repose sur des méthodes numériques

telles que les schémas d’Euler ou de Runge–Kutta, ainsi que sur des techniques de linéarisation pour étudier la stabilité locale via la matrice Jacobienne $\mathbf{DF}(\boldsymbol{\Omega}^*)$.

En outre, l’introduction de mécanismes de **bornage** (par exemple, le clipping des ω_{ij} dans l’intervalle $[0, \omega_{\max}]$) assure que l’espace d’état \mathcal{X} est compact, ce qui, en conjonction avec la continuité de F , garantit l’existence globale d’au moins un attracteur. Ainsi, même si la majorité des cas non linéaires du DSL ne permettent pas une intégration fermée, la solution est en général bien définie et ne diverge pas, ce qui constitue une propriété essentielle pour l’auto-organisation du réseau.

Pour résumer, la **solution fermée** est accessible uniquement dans des situations simplifiées (par exemple, lorsque $S(i, j)$ est constant ou linéaire et stationnaire), tandis que pour les scénarios non linéaires plus réalistes, on s’appuie sur des **approches numériques** et des techniques d’analyse qualitative. L’existence et l’unicité des solutions reposent sur la continuité et la condition de local Lipschitz de la fonction \mathbf{F} , et la présence d’un terme de décroissance $-\tau \omega_{ij}(t)$ (ou l’application de bornes explicites) assure que la dynamique reste contractante et qu’un attracteur est présent dans l’espace des pondérations. Ces arguments illustrent comment la théorie des EDO s’applique au DSL et fournit les fondations mathématiques nécessaires à l’étude de ses propriétés dynamiques.

2.4.2. Parallèles avec les Modèles de Spin (Ising, Potts, Hopfield)

Les **modèles de spin** (Ising, Potts, Hopfield) en physique statistique et en neurosciences computationnelles partagent avec le **DSL** (Deep Synergy Learning) une approche où des **entités** interagissent via des **couplages** susceptibles de se renforcer ou de conduire à des configurations d’énergie plus ou moins stable. La section 2.4.2 met en évidence ces similitudes, soulignant à la fois l’inspiration possible pour le DSL et les limites de l’analogie. Nous commençons (2.4.2.1) par la notion de **minima énergétiques** et de **transitions de phase** dans un réseau, avant d’évoquer (2.4.2.2) la comparaison $\omega_{ij} \leftrightarrow$ couplage spin/neuron, et de modérer (2.4.2.3) la portée de ce parallèle du fait de la **non-linéarité** évolutive de la synergie en DSL.

2.4.2.1. Notion de minima énergétiques, transitions de phase dans un réseau

La modélisation des systèmes complexes, qu’ils soient issus des modèles de spins classiques ou des réseaux neuronaux auto-organisés tels que le **Deep Synergy Learning (DSL)**, s’appuie souvent sur l’idée d’un paysage d’**énergie** comportant divers **minima**. Dans les modèles de spins, comme celui d’**Ising**, de **Potts** ou le réseau de **Hopfield**, la fonction d’énergie (ou **Hamiltonien**) définit une configuration stable ou un attracteur du système. Par exemple, dans le modèle d’Ising, chaque spin s_i (avec $s_i \in \{-1, +1\}$) est associé à une énergie donnée par

$$\mathcal{H}(\{s_i\}) = - \sum_{\langle i,j \rangle} J_{ij} s_i s_j - \sum_i h_i s_i,$$

où la somme $\langle i,j \rangle$ se fait sur les paires de spins voisins et J_{ij} désigne le **couplage** entre les spins i et j . Dans le modèle de **Potts**, qui généralise le modèle d’Ising à q états, le principe reste le même

: l'alignement des spins sur une valeur commune est favorisé, conduisant à des configurations d'énergie minimale. Le réseau de **Hopfield**, quant à lui, définit sa fonction d'énergie par

$$\mathcal{H} = -\frac{1}{2} \sum_{i,j} W_{ij} s_i s_j,$$

où W_{ij} représentent les poids ou **coupages synaptiques**, et les minima de cette fonction correspondent aux états mémorisés ou aux attracteurs stables du système.

Dans le cadre du DSL, bien que la dynamique ne soit pas explicitement définie en termes d'un Hamiltonien classique, l'évolution des **pondérations** ω_{ij} peut être interprétée comme la recherche d'un état d'**optimisation** du réseau. Par analogie, on peut définir une pseudo-énergie associée aux pondérations par

$$\mathcal{E}(\Omega) \approx - \sum_{(i,j)} \omega_{ij} S(i,j),$$

où $S(i,j)$ représente la **synergie** entre les entités \mathcal{E}_i et \mathcal{E}_j . Dans cette optique, renforcer une liaison w_{ij} lorsque la synergie $S(i,j)$ est élevée équivaut à abaisser l'énergie du système, ce qui est favorable du point de vue de l'**auto-organisation**. Les **minima énergétiques** ainsi définis correspondent à des configurations stables où l'ensemble des pondérations converge vers un état attracteur, semblable aux minima locaux ou globaux observés dans les modèles de spins.

Un aspect crucial dans l'étude de ces systèmes est l'apparition de **transitions de phase**. Dans le modèle d'Ising, par exemple, on observe une transition d'un état désordonné (où les spins sont aléatoirement orientés) à un état ordonné (où la majorité des spins s'alignent) lorsque la température T chute en dessous d'un certain seuil critique T_c . Par analogie, dans un SCN, si l'on fait varier un paramètre équivalent à la température – par exemple, le niveau de bruit injecté dans la dynamique ou le rapport η/τ – le système peut passer d'un régime où les pondérations sont trop fluctuantes pour permettre une structure stable à un régime où des **clusters** bien définis émergent. Cette transition, caractérisée par une « brisure de symétrie » dans l'organisation des pondérations, est analogue à la transition ordre-désordre des modèles de spins.

Les exemples de **minima énergétiques** se retrouvent, dans un DSL simplifié, lorsque l'on impose une synergie binaire ou continue mais stationnaire, ce qui permet de résoudre analytiquement l'équation de mise à jour continue (comme vu en section 2.4.1.2) et d'obtenir le point fixe

$$\omega_{ij}^* = \frac{S(i,j)}{\tau}.$$

Dans un cadre plus complexe, où $S(i,j)$ dépend de manière non linéaire des pondérations ou d'états internes, le système présente des **attracteurs multiples** et le paysage d'énergie devient riche en minima locaux. Chaque minimum représente une configuration stable locale du réseau, correspondant à un cluster ou à une partition particulière de l'ensemble des entités. Les **transitions de phase** se manifestent alors par des changements brusques dans l'organisation du réseau lorsqu'un paramètre (tel que le bruit ou le rapport η/τ) atteint un seuil critique, conduisant à un basculement d'un attracteur à un autre.

En résumé, les modèles de spins – Ising, Potts, Hopfield – nous fournissent un vocabulaire riche qui peut être transposé au DSL, où l'on parle de **minima énergétiques** et de **transitions de phase** pour interpréter les configurations stables des pondérations. Un **minimum local** d'énergie dans le DSL correspond à une configuration des poids Ω qui maximise la synergie interne, et la transition d'un état diffus à un état clusterisé s'interprète comme une transition ordre-désordre. Bien que la synergie $S(i,j)$ dans le DSL puisse varier dans le temps ou dépendre d'états internes, l'analogie avec la minimisation d'un Hamiltonien offre une perspective éclairante pour comprendre l'émergence des **clusters** et l'existence d'attracteurs dans l'espace des pondérations.

Conclusion

La notion de **minima énergétiques** et de **transitions de phase** dans les modèles de spins trouve un parallèle évident dans l'analyse du DSL. Dans les deux cadres, l'optimisation d'une fonction (qu'elle soit l'énergie ou une pseudo-énergie définie par $\mathcal{E}(\Omega) \approx -\sum_{(i,j)} \omega_{ij} S(i,j)$) conduit à la formation d'états stables, et des transitions abruptes peuvent apparaître lorsque des paramètres critiques sont franchis. Ces analogies, en reliant les couplages ω_{ij} aux interactions de spins et en interprétant la dynamique des pondérations comme une recherche de minima dans un paysage d'énergie, fournissent un langage commun permettant de comprendre les phénomènes d'auto-organisation observés dans le DSL, même lorsque la fonction de synergie varie dans le temps ou dépend de plusieurs facteurs. Les sections suivantes approfondiront ces correspondances et examineront les limites de cette analogie, notamment lorsque la dynamique du DSL intègre des aspects temporels et non linéaires complexes.

2.4.2.2. Interprétation de ω_{ij} comme “couplage” entre spins ou neurones

Dans les modèles classiques de la physique statistique, tels que les modèles de **spins d'Ising** ou de **Potts**, ainsi que dans les réseaux de **Hopfield**, la notion de couplage est centrale. Dans le modèle d'Ising, par exemple, chaque spin s_i prenant des valeurs dans $\{-1, +1\}$ interagit avec ses voisins via des paramètres J_{ij} qui, selon leur signe, favorisent l'alignement ou l'antialignement. La fonction d'énergie (Hamiltonien) s'exprime typiquement sous la forme

$$\mathcal{H}(\{s_i\}) = - \sum_{\langle i,j \rangle} J_{ij} s_i s_j - \sum_i h_i s_i,$$

où la somme $\langle i,j \rangle$ parcourt les paires de spins voisins et h_i représente un champ externe appliqué sur le site i . Dans le modèle de Potts, la généralisation aux q états se traduit par une évaluation similaire de la similitude ou de la différence d'états entre spins, tandis que dans le réseau de Hopfield, l'énergie est définie par

$$E(\mathbf{s}) = -\frac{1}{2} \sum_{i,j} W_{ij} s_i s_j,$$

les poids W_{ij} constituant alors les **couplages synaptiques** qui restent fixés au cours du temps et dont les minima énergétiques correspondent aux états stables du système (mémoires associatives).

Dans le contexte du **Deep Synergy Learning (DSL)**, la pondération ω_{ij} remplit une fonction analogue à celle des couplages J_{ij} ou W_{ij} dans les modèles de spins. En effet, ω_{ij} quantifie la force du lien entre deux entités \mathcal{E}_i et \mathcal{E}_j . Une valeur élevée de ω_{ij} signifie que la synergie entre les entités, mesurée par $S(i,j)$, est forte et que le lien entre elles est donc très robuste. La dynamique du DSL repose sur une mise à jour de type

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)],$$

ce qui signifie que la pondération évolue en fonction de la différence entre la synergie effective et un terme de décroissance proportionnel à ω_{ij} . À cet égard, ω_{ij} joue un rôle similaire à un couplage adaptatif. En effet, dans les réseaux de Hopfield ou d'Ising, le couplage entre deux spins est un paramètre constant qui oriente l'alignement des états s_i et s_j . Dans le DSL, ce sont les **liaisons** elles-mêmes qui évoluent en temps réel en réponse aux mesures de synergie, de sorte que ω_{ij} est non seulement un indicateur de la force de connexion, mais également un paramètre qui se met à jour pour refléter l'état d'interaction actuel entre les entités.

Une autre différence fondamentale réside dans le fait que, dans les modèles classiques, l'énergie du système est exprimée directement en fonction des spins, tandis que dans le DSL, l'analogie se construit en définissant une pseudo-énergie du réseau. Par exemple, on peut envisager la fonction

$$\mathcal{E}(\Omega) \approx - \sum_{(i,j)} \omega_{ij} S(i,j),$$

qui traduit l'idée que renforcer les pondérations ω_{ij} lorsque $S(i,j)$ est élevée réduit l'énergie globale du système, à la manière dont un couplage J_{ij} élevé favorise un état ordonné dans un modèle de spins. Cette interprétation permet d'assimiler, d'une certaine manière, le rôle de ω_{ij} à celui d'un couplage classique, même si la synergie $S(i,j)$ dans le DSL peut résulter de calculs plus complexes qu'un simple produit scalaire ou une simple corrélation.

Par ailleurs, la dynamique DSL se distingue par le fait que, contrairement aux modèles d'Ising ou de Hopfield où les couplages sont fixes et ce sont les états (spins ou activations) qui évoluent, dans le DSL, ce sont les **liaisons** elles-mêmes qui évoluent tandis que l'état interne de chaque entité peut rester relativement stable ou évoluer indépendamment. Cette inversion dans le rôle des variables signifie que le DSL est axé sur l'adaptation des **relations** entre les entités, ce qui permet au réseau de se réorganiser de manière flexible en réponse à de nouvelles informations.

Enfin, il convient d'envisager les extensions possibles de cette analogie vers des interactions plus complexes. Tandis que les modèles classiques de spins se limitent à des couplages binaires, le DSL ouvre la possibilité d'introduire des interactions **n-aires**. Ces interactions, qui impliquent la synergie collective de plusieurs entités, peuvent être représentées dans un cadre d'**hypergraphes** où les hyper-liens traduisent des relations qui ne se décomposent pas simplement en interactions par paires. Même si cette généralisation dépasse le cadre des modèles de spins classiques, elle enrichit la compréhension du DSL en offrant une perspective plus large sur la **coopération** entre entités.

En conclusion, l'interprétation de ω_{ij} dans le DSL comme un **couplage** entre entités s'appuie sur plusieurs points de convergence avec les modèles de spins et les réseaux de Hopfield. D'une part,

ω_{ij} joue le rôle de modulateur de la force d'interaction, tout comme J_{ij} ou W_{ij} favorisent l'alignement des spins dans les modèles classiques. D'autre part, la dynamique évolutive des pondérations, qui suit une mise à jour continue ou discrète, permet au DSL de s'adapter de manière dynamique, en modifiant les liaisons en fonction de la **synergie effective** mesurée entre les entités. Bien que cette dynamique diffère du cadre statique traditionnel des modèles de spins, l'analogie reste pertinente et fournit un langage commun pour l'analyse des attracteurs, des transitions de phase et de l'auto-organisation dans les réseaux complexes. Les développements ultérieurs aborderont les limites de cette analogie et exploreront comment l'intégration de synergies n-aires peut enrichir cette approche en modélisant des interactions plus collectives et sophistiquées.

2.4.2.3. Limites de la comparaison : le DSL introduit une synergie “non-linéaire” pouvant varier au fil du temps

Dans les sections précédentes, les analogies établies entre le **DSL** (Deep Synergy Learning) et les modèles de spin classiques – notamment ceux d'Ising, de Potts ou de Hopfield – se fondent sur l'assimilation de la pondération ω_{ij} à un **couplage** comparable aux paramètres J_{ij} ou W_{ij} . Dans ces modèles, le couplage est généralement considéré comme une constante fixe, tandis que dans le DSL, la dynamique des pondérations se fait évoluer en temps réel. Toutefois, cette comparaison présente des limites majeures, car la **synergie** $S(i,j)$ dans le DSL est souvent caractérisée par une nature **non-linéaire** et par une variabilité temporelle qui ne se retrouve pas dans les modèles de spins stationnaires.

Dans les modèles classiques tels que ceux d'Ising ou de Potts, chaque spin $s_i \in \{-1, +1\}$ interagit avec ses voisins via des couplages J_{ij} qui sont fixés dans le temps, sauf dans certaines variantes où l'on peut envisager une plasticité synaptique. Par exemple, l'énergie du modèle d'Ising est donnée par

$$\mathcal{H}(\{s_i\}) = - \sum_{\langle i,j \rangle} J_{ij} s_i s_j - \sum_i h_i s_i,$$

où la matrice $\{J_{ij}\}$ est supposée constante, permettant ainsi une descente d'énergie par l'évolution des spins $\{s_i\}$. Dans un réseau de Hopfield, les poids $\{W_{ij}\}$ sont également fixés après apprentissage et la dynamique repose sur l'évolution des activations s_i pour minimiser la fonction d'énergie

$$\mathcal{H}(\mathbf{s}) = -\frac{1}{2} \sum_{i,j} W_{ij} s_i s_j.$$

En contraste, dans le DSL, la pondération ω_{ij} évolue en fonction de la synergie mesurée, par exemple selon une règle de mise à jour discrète

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)],$$

où η est le **taux d'apprentissage** et τ le **coefficent de décroissance**. Ici, la synergie $S(i,j)$ peut être issue de diverses mesures (distance, similarité, co-information) et, contrairement aux couplages classiques, elle peut dépendre de variables évolutives telles que les représentations \mathbf{x}_i des entités ou même des états internes \mathbf{s}_i . De plus, dans certaines variantes du DSL, $S(i,j)$ intègre des termes dépendant d'autres pondérations, par exemple sous la forme d'un terme d'**inhibition compétitive** tel que

$$S(i,j) \propto -\gamma \sum_{k \neq j} \omega_{i,k},$$

ce qui introduit une non-linéarité supplémentaire dans la dynamique. Par conséquent, la fonction $S(i,j)$ ne se réduit pas à un simple produit scalaire $s_i \times s_j$ comme dans les modèles de spins, mais peut varier en fonction des **représentations** et évoluer en temps réel.

Cette variabilité pose des problèmes pour établir une analogie rigoureuse avec une fonction d'énergie stationnaire. En effet, dans un système tel que celui de Hopfield, l'énergie

$$\mathcal{H}(\mathbf{s}) = -\frac{1}{2} \sum_{i,j} W_{ij} s_i s_j$$

reste fixe en fonction des couplages W_{ij} tandis que les spins évoluent. Dans le DSL, en revanche, les pondérations ω_{ij} ainsi que la synergie $S(i,j)$ se réévaluent continuellement, ce qui rend difficile l'établissement d'un paysage d'énergie immuable, c'est-à-dire d'une fonction $\mathcal{E}(\Omega)$ telle que

$$\mathcal{E}(\Omega) \approx - \sum_{(i,j)} \omega_{ij} S(i,j)$$

soit une mesure stable sur laquelle le système effectuerait une descente d'énergie. Par ailleurs, la dynamique du DSL se caractérise par le fait que les entités elles-mêmes, représentées par leurs vecteurs \mathbf{x}_i ou leurs états internes \mathbf{s}_i , sont relativement stables par rapport aux pondérations. Ainsi, dans les modèles classiques, ce sont les spins qui varient tandis que le couplage reste fixe, tandis que dans le DSL, le couplage ω_{ij} évolue, ce qui inverse la dynamique habituelle.

De plus, la synergie $S(i,j)$ peut être influencée par des mécanismes complexes et non linéaires, tels que la dépendance aux états internes ou l'intégration d'une co-information multi-entités. Par conséquent, le paysage d'énergie du DSL est en constante évolution, et une configuration de pondérations optimale à un instant donné peut se transformer à l'instant suivant si la synergie se modifie. Ce caractère dynamique et non stationnaire limite la validité d'une comparaison stricte avec des modèles de spins classiques, où l'on dispose d'un Hamiltonien fixe et d'une descente d'énergie bien définie.

En conclusion, bien que l'analogie entre ω_{ij} dans le DSL et les couplages J_{ij} ou W_{ij} des modèles de spins fournisse un langage utile pour comprendre la formation des clusters et la multi-stabilité du système, elle reste partielle. Le DSL introduit une **synergie non-linéaire** et variable dans le temps, ce qui complexifie la notion d'un paysage d'énergie statique et remet en question l'application directe des concepts de descente d'énergie et de transitions de phase issus de la physique statistique. Néanmoins, cette analogie demeure précieuse pour interpréter qualitativement les phénomènes d'**attraction** et de **basculement** observés dans le DSL, tout en reconnaissant que la dynamique de ce système dépasse largement la structure d'un couplage fixe et d'un Hamiltonien stationnaire.

Conclusion

La comparaison entre le DSL et les modèles de spins, bien que riche en enseignements, présente des limites importantes. Dans les modèles classiques, le couplage est fixe et l'énergie est définie de

manière stationnaire, ce qui permet une analyse rigoureuse de la descente d'énergie et des transitions de phase. En revanche, dans le DSL, la **synergie** $S(i,j)$ est souvent non linéaire et évolutive, ce qui empêche de définir un paysage d'énergie immuable et complique l'interprétation quantitative des attracteurs et des transitions de phase. Cette analogie, bien qu'instructive, ne peut être considérée que comme partielle et doit être nuancée en tenant compte de la nature dynamique et variable du DSL, qui intègre des mécanismes d'adaptation et des interactions évolutives dépassant le cadre d'un couplage statique.

2.4.3. Notion d'Énergie ou de Fonction Potentielle

Les **modèles de spin** (Ising, Potts, Hopfield) s'appuient sur une **énergie** \mathcal{H} (ou \mathcal{E}) que l'évolution du système vise à **minimiser** ou dont il cherche à **explorer** les minima. Dans le cadre du **Deep Synergy Learning** (DSL), on aimerait parfois définir un **anologue** de cette fonction d'énergie ou de potentiel, afin de mieux comprendre la **stabilité** et la **convergence** du SCN (Synergistic Connection Network). La section 2.4.3 soulève la question de savoir si on peut formaliser $\mathcal{J}(\Omega)$ (2.4.3.1), examiner l'existence de “descente d'énergie” (2.4.3.2) et évaluer les avantages/inconvénients de cette approche (2.4.3.3).

2.4.3.1. Peut-on formaliser $\mathcal{J}(\Omega) = -\sum_{i,j} \omega_{ij} S(i,j) + \dots?$

Dans l'étude du **Deep Synergy Learning (DSL)**, une question centrale consiste à savoir si la dynamique des pondérations ω_{ij} peut être interprétée comme une descente d'énergie, analogue à la minimisation d'un Hamiltonien dans les modèles de spins. Pour ce faire, on cherche à définir une fonction de coût ou une **fonction énergétique** $\mathcal{J}(\Omega)$ dont la minimisation conduirait naturellement aux mises à jour observées dans le DSL. Une proposition heuristique couramment avancée est de poser

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i,j) + (\text{termes de régularisation}),$$

où $S(i,j)$ désigne la **synergie effective** entre les entités \mathcal{E}_i et \mathcal{E}_j . Le premier terme, $-\omega_{ij} S(i,j)$, est conçu pour encourager l'augmentation des pondérations lorsque la synergie est élevée, c'est-à-dire qu'en minimisant \mathcal{J} on favorise une configuration dans laquelle les liens forts correspondent aux synergies importantes. Pour éviter que les pondérations ne croissent indéfiniment, on ajoute généralement un terme de régularisation qui peut prendre la forme d'une pénalisation quadratique, par exemple

$$\frac{\tau}{2} \sum_{i,j} \omega_{ij}^2,$$

de sorte que l'expression complète devient

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

où $\tau > 0$ agit comme un **paramètre de régularisation**. Le gradient partiel par rapport à une pondération ω_{ij} s'obtient alors en différentiant \mathcal{J} par rapport à ω_{ij} , ce qui conduit à

$$\frac{\partial \mathcal{J}}{\partial \omega_{ij}} = -S(i,j) + \tau \omega_{ij}.$$

Ainsi, si l'on interprète la règle de mise à jour du DSL comme un pas de gradient négatif, on obtient

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)} = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)],$$

ce qui correspond exactement à la mise à jour standard utilisée dans le DSL dans le cas stationnaire où $S(i,j)$ est indépendant de ω_{ij} et des autres pondérations.

Cette correspondance formelle permet de comprendre que, dans un cadre idéal et simplifié, le DSL opère comme une **descente de gradient** sur la fonction \mathcal{J} , ce qui aurait pour effet de minimiser l'énergie globale du réseau. L'hypothèse sous-jacente est que la synergie $S(i,j)$ est stationnaire ou du moins qu'elle varie lentement par rapport aux mises à jour des pondérations. Dans ce cas, le paysage énergétique défini par

$$\mathcal{J}(\boldsymbol{\Omega}) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2$$

possède un minimum global que le système cherche à atteindre, avec pour point fixe

$$\omega_{ij}^* = \frac{S(i,j)}{\tau},$$

ce qui reflète une convergence exponentielle, comme nous l'avons vu dans l'analyse de la variante continue.

Cependant, des **obstacles** apparaissent dès lors que l'on introduit des éléments de non-linéarité dans la fonction de synergie. Lorsque $S(i,j)$ dépend de manière non triviale des autres pondérations $\{\omega_{p,q}\}$ ou d'états internes variables, le terme $S(i,j)$ devient lui-même une fonction de $\boldsymbol{\Omega}$ et ne peut plus être considéré comme une constante. Dans ce cas, la dérivation de \mathcal{J} ne se réduit plus à une simple fonction linéaire en ω_{ij} et l'opérateur de mise à jour

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j; \{\omega_{p,q}(t)\}) - \tau \omega_{ij}(t)]$$

ne correspond plus à une descente de gradient d'une unique fonction énergétique $\mathcal{J}(\boldsymbol{\Omega})$ fixe. Le paysage énergétique devient alors **dynamique** et se reconfigure en continu, reflétant des interactions complexes et souvent non stationnaires. En d'autres termes, la fonction $S(i,j)$ peut évoluer en réponse aux changements dans $\boldsymbol{\Omega}$ et aux entrées extérieures, ce qui empêche la formulation d'un Hamiltonien fixe et stable comme dans les modèles de spins classiques.

Dans cette situation, il est souvent nécessaire de recourir à des **approches numériques** pour étudier l'évolution du système et pour déterminer qualitativement l'existence et la nature des attracteurs. On peut ainsi utiliser des schémas d'intégration numérique (comme Euler ou Runge–Kutta) pour approximer la trajectoire $\omega_{ij}(t)$, et des techniques de **linéarisation** autour des points fixes pour

analyser la stabilité locale à l'aide de la matrice Jacobienne $\mathbf{DF}(\boldsymbol{\Omega}^*)$. Ces méthodes permettent de caractériser les attracteurs locaux, même si l'on ne peut pas écrire explicitement une solution fermée pour l'ensemble des pondérations.

De plus, des contraintes de **bornage** (par exemple, imposer que $\omega_{ij} \in [0, \omega_{\max}]$) ou l'utilisation de mécanismes de **saturation** garantissent que l'espace des solutions reste compact, ce qui est indispensable pour appliquer les théorèmes classiques de point fixe (Brouwer, Schauder). Cela permet d'assurer, malgré l'absence de solution analytique globale, que le système admet au moins un attracteur dans l'espace des configurations, même si ce dernier dépend de paramètres externes et de la dynamique non linéaire de $S(i,j)$.

En synthèse, la formalisation d'une fonction énergétique globale dans le DSL se fait aisément dans des cas **simples** où $S(i,j)$ est constant ou linéaire, ce qui conduit à une solution fermée du type

$$\omega_{ij}(t) = \frac{S(i,j)}{\tau} + \left(\omega_{ij}(0) - \frac{S(i,j)}{\tau} \right) \exp(-\eta \tau t).$$

Dans les scénarios plus **complexes** et non linéaires, la dynamique du DSL ne correspond pas à la descente d'un Hamiltonien fixe, mais plutôt à un processus adaptatif évoluant dans un paysage énergétique variable. Les approches pour étudier ces systèmes reposent alors sur des méthodes numériques et sur l'analyse qualitative via la linéarisation locale. Ainsi, bien que la vision « $\mathcal{J}(\boldsymbol{\Omega}) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots$ » soit pertinente dans un cadre stationnaire, elle doit être nuancée lorsque la synergie dépend de l'état du système et varie dans le temps.

Conclusion

On peut formaliser, dans un scénario restreint, une fonction énergétique

$$\mathcal{J}(\boldsymbol{\Omega}) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

telle que la mise à jour des pondérations s'apparente à une descente de gradient

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)} = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)].$$

Cependant, dès que la fonction $S(i,j)$ présente des dépendances non linéaires ou évolutives en fonction de $\boldsymbol{\Omega}$ ou d'états internes, la notion d'un paysage d'énergie unique et fixe devient moins pertinente, et le système évolue dans un cadre « hors équilibre ». Dans ce cas, l'approche repose sur des méthodes numériques et une analyse qualitative de la dynamique. Ainsi, la question d'une solution fermée se résout par un “oui” dans des cas idéalisés et par un “non” dans la majorité des situations réalistes, nécessitant des approches approximatives pour garantir l'existence et la stabilité des solutions du DSL.

2.4.3.2. Existence de descentes d'énergie locales : recuit simulé ou itération stochastique

Dans le cadre du **Deep Synergy Learning (DSL)**, l'idée d'interpréter la dynamique des pondérations ω_{ij} comme une descente dans un paysage énergétique a suscité un intérêt particulier, en raison de ses analogies avec les méthodes d'optimisation stochastique telles que le **recuit simulé** (simulated annealing). Nous allons examiner ici la possibilité de formaliser une **descente d'énergie locale** dans le DSL, ainsi que les méthodes par lesquelles un tel processus, éventuellement perturbé par du bruit aléatoire, peut permettre au système d'explorer différents minima énergétiques et ainsi d'échapper à des minima locaux non optimaux.

Dans les modèles de spins classiques, par exemple dans le modèle d'**Ising** ou dans les réseaux de **Hopfield**, l'algorithme de recuit simulé est utilisé pour échapper aux pièges des minima locaux. Le principe est d'introduire une **température** T qui module la probabilité d'accepter une augmentation de l'énergie $\Delta\mathcal{H}$ selon une loi de probabilité de type

$$P(\Delta\mathcal{H}) \propto \exp\left(-\frac{\Delta\mathcal{H}}{T}\right).$$

Lorsque T est élevé, le système explore librement l'espace des configurations, même si certaines transitions font temporairement augmenter l'énergie. Au fil du temps, T diminue progressivement, limitant ainsi les fluctuations et permettant au système de converger vers un minimum global ou, du moins, vers un minimum local satisfaisant.

Pour transposer cette approche au DSL, on suppose qu'il est possible de définir une **pseudo-énergie** $\mathcal{J}(\Omega)$ qui dépend de l'ensemble des pondérations ω_{ij} et qui prend la forme

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

où $S(i,j)$ représente la **synergie effective** entre les entités \mathcal{E}_i et \mathcal{E}_j . Dans le cas stationnaire et en l'absence de dépendances complexes de S en fonction de Ω , la descente de gradient appliquée à \mathcal{J} conduit à une mise à jour

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)} = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)],$$

ce qui correspond exactement à la règle de mise à jour usuelle du DSL. Ce constat démontre que, dans des cas idéalisés, le DSL effectue effectivement une **descente d'énergie locale** vers le point fixe $\omega_{ij}^* = S(i,j)/\tau$.

Néanmoins, la réalité est souvent plus complexe. Dans de nombreuses applications, la fonction de synergie $S(i,j)$ est sujette à des **non-linéarités** et peut varier en fonction des autres pondérations $\{\omega_{p,q}\}$ ou des états internes des entités. Pour intégrer ce comportement, on peut modifier la règle de mise à jour en y ajoutant un terme de bruit stochastique, de sorte que l'équation prenne la forme

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega(t)} + \sigma \xi_{ij}(t),$$

où $\xi_{ij}(t)$ représente une perturbation aléatoire et σ en détermine l'amplitude. Ce terme de bruit permet d'imaginer une dynamique d'**itération stochastique** ou de **recuit simulé** qui, en phase initiale, autorise le système à franchir des barrières énergétiques, explorant ainsi différents minima du paysage. Au fil du temps, un **planning de refroidissement** (par exemple, une décroissance exponentielle de σ telle que $\sigma(t) \propto \alpha^t$ avec $0 < \alpha < 1$) est appliqué afin de réduire progressivement l'influence du bruit, permettant alors au système de se stabiliser dans une configuration potentiellement plus globale.

La démarche proposée s'inspire directement des algorithmes de **recuit simulé** utilisés dans les modèles d'optimisation stochastique. En Ising ou dans une machine de Boltzmann, on utilise des critères probabilistes pour accepter ou refuser des changements qui augmentent l'énergie, ce qui permet de surmonter le problème des minima locaux. Dans le DSL, bien que le paysage énergétique ne soit pas nécessairement fixe – en particulier lorsque la synergie $S(i,j)$ est fonction des pondérations elles-mêmes – l'injection d'un bruit contrôlé offre un levier pour encourager l'exploration et éviter que le réseau ne se bloque prématurément dans une configuration sous-optimale.

Pour illustrer cette approche, il convient d'imaginer un protocole « à la recuit simulé » appliquée aux pondérations. Dans la **phase chaude**, le niveau de bruit $\sigma(0)$ est élevé, ce qui autorise de fortes fluctuations dans ω_{ij} et favorise l'exploration de configurations variées. Puis, lors d'un **refroidissement progressif**, $\sigma(t)$ décroît de manière prédefinie, par exemple suivant une loi exponentielle ou linéaire, et par conséquent, les modifications aléatoires deviennent de plus en plus rares. Finalement, dans la **phase finale** où $\sigma(t) \approx 0$, le système se stabilise dans l'un des minima locaux de la fonction pseudo-énergétique $\mathcal{J}(\Omega)$.

Ce mécanisme est particulièrement pertinent dans les situations où la fonction $S(i,j)$ est suffisamment stationnaire pour que la notion de descente d'énergie ait un sens. Toutefois, dans des scénarios où $S(i,j)$ varie de manière significative en fonction des pondérations ou des états internes, le paysage d'énergie devient mouvant et la descente n'est qu'une **approximation** de la dynamique réelle. Dans ce cas, le recuit simulé ne garantit pas la minimisation d'une fonction énergétique fixe, mais sert plutôt de stratégie pour favoriser l'exploration et pour empêcher le système de se figer dans des minima locaux non désirés.

Conclusion

Ainsi, dans un cadre restreint où la synergie $S(i,j)$ reste relativement stationnaire, il est possible de formaliser la dynamique du DSL comme une descente d'énergie locale à partir d'une fonction pseudo-énergétique

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

ce qui conduit à la mise à jour

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)} = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)].$$

Pour explorer d'autres configurations et potentiellement échapper à des minima locaux, un terme de bruit $\sigma \xi_{ij}(t)$ peut être ajouté, reproduisant ainsi le comportement d'un recuit simulé. Ce

protocole stochastique permet au système d'explorer divers attracteurs avant de converger vers une configuration stable lorsque le niveau de bruit décroît progressivement. Cependant, si la synergie $S(i, j)$ dépend fortement des pondérations ou d'états internes variables, le paysage d'énergie n'est plus fixe et la descente d'énergie devient une approximation qui ne capture que partiellement la dynamique réelle. Néanmoins, l'introduction d'un recuit simulé ou d'itérations stochastiques dans le DSL constitue un levier important pour encourager l'exploration et pour éviter un blocage prématué du système dans un minimum local sous-optimum.

En conclusion, le DSL peut, dans des conditions appropriées, être interprété comme effectuant une descente d'énergie locale, et l'injection contrôlée de bruit – de type recuit simulé – permet d'explorer le paysage énergétique et d'optimiser la structure du réseau. Cette approche, bien que dépendante d'hypothèses stationnaires, fournit une base théorique et pratique pour comprendre comment le système peut échapper aux minima locaux et potentiellement converger vers une configuration plus optimale.

2.4.3.3. Avantages et inconvénients de cette vision “énergétique” pour expliquer la convergence

L'approche consistant à interpréter la dynamique du DSL à travers une fonction d'énergie $\mathcal{J}(\Omega)$ offre une perspective particulièrement intéressante pour analyser la convergence des pondérations ω_{ij} dans un Synergistic Connection Network (DSL). En effet, en posant par exemple

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i, j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

on traduit l'idée que le système cherche à maximiser la synergie entre entités tout en régulant la croissance des pondérations à travers un terme quadratique de **régularisation**. Dans ce cadre, la mise à jour

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)}$$

se simplifie en

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i, j) - \tau \omega_{ij}(t)],$$

ce qui correspond exactement à la règle de mise à jour adoptée dans le DSL dans un cadre stationnaire. Dans cette situation idéale, la dynamique s'apparente à une **descente de gradient** dans un paysage d'énergie fixe, permettant d'expliquer l'**auto-organisation** des pondérations comme une recherche d'un minimum local de \mathcal{J} . Le principal avantage de cette vision est qu'elle fournit un cadre conceptuel puissant et familier aux chercheurs en physique statistique. Par analogie avec les modèles d'Ising et de Hopfield, où l'énergie $\mathcal{H}(\mathbf{s})$ guide l'évolution des spins, cette approche énergétique permet de justifier l'émergence d'attracteurs et de **clusters** en termes de minimisation d'un potentiel global.

D'un point de vue théorique, cette formalisation simplifie l'analyse des propriétés de convergence, car elle permet de mobiliser des outils issus de l'**optimisation** et de la **théorie des systèmes**

dynamiques. En effet, dans le cas où la synergie $S(i, j)$ demeure relativement stationnaire et ne dépend pas trop des pondérations elles-mêmes, le paysage d'énergie défini par $\mathcal{J}(\Omega)$ est stable et la dynamique converge vers un minimum global ou, au moins, vers un attracteur local caractérisé par

$$\omega_{ij}^* = \frac{S(i, j)}{\tau}.$$

Cette interprétation favorise l'emploi de stratégies de **recuit simulé** ou de **gradient stochastique** qui permettent d'explorer l'espace des configurations et d'éviter de se figer prématûrement dans des minima locaux sous-optimaux. Le concept de "température" dans ces méthodes, qui module l'amplitude du bruit injecté dans la mise à jour, offre un levier pour contrôler le compromis entre exploration et exploitation du paysage énergétique.

Cependant, cette vision présente également des **limites** importantes lorsqu'elle est appliquée à des scénarios réels du DSL. En effet, dans la majorité des cas, la fonction de synergie $S(i, j)$ n'est pas une valeur fixe mais une fonction **non-linéaire** qui peut dépendre des autres pondérations $\{\omega_{p,q}\}$, des états internes s_i et même de paramètres externes. Ainsi, le paysage d'énergie $\mathcal{J}(\Omega)$ n'est pas immuable et se reconfigure en continu. Ce phénomène remet en cause l'hypothèse d'un **Hamiltonien stationnaire** et limite l'applicabilité de la descente d'énergie telle que décrite dans les modèles classiques. Dans un tel cadre, même si la mise à jour locale des pondérations peut être interprétée comme une descente de gradient sur une fonction d'énergie approximative, il est difficile de garantir que le système converge vers un minimum global ou même vers un attracteur stable, puisque la fonction $S(i, j)$ elle-même évolue en temps réel.

De plus, contrairement aux modèles d'Ising ou de Hopfield, dans lesquels les couplages (comme J_{ij} ou W_{ij}) sont généralement **fixes** et seuls les états (spins, activations) varient, le DSL fait évoluer les pondérations ω_{ij} de manière adaptative. Cette dynamique, qui s'appuie sur des mécanismes d'auto-organisation complexes, engendre un système souvent **hors équilibre**, dans lequel la notion de descente d'un paysage énergétique fixe est plus difficile à appliquer de manière rigoureuse.

En synthèse, la **vision énergétique** du DSL offre des avantages considérables, notamment en permettant d'interpréter la formation de clusters et la convergence des pondérations comme le résultat d'une **descente de gradient** dans un paysage d'énergie. Ce cadre permet d'employer des techniques d'**optimisation stochastique** et de recuit simulé pour favoriser l'exploration de l'espace de configurations et éviter le piégeage dans des minima locaux peu optimaux. Néanmoins, lorsque la synergie $S(i, j)$ varie significativement en fonction des pondérations et des états internes, le paysage d'énergie devient dynamique, et la correspondance avec un Hamiltonien stationnaire s'efface. Par conséquent, cette approche doit être considérée comme une **approximation** utile dans des scénarios stationnaires ou faiblement dépendants, mais elle perd en pertinence lorsque le système se comporte de manière fortement non linéaire et adaptative. Ainsi, bien que l'analogie avec une fonction d'énergie soit instructive pour expliquer qualitativement la convergence du DSL, elle présente des limites quant à sa capacité à rendre compte de la totalité de la dynamique du système.

Conclusion

La vision énergétique du DSL, qui repose sur l'hypothèse que les pondérations ω_{ij} évoluent de manière à minimiser une fonction pseudo-énergétique telle que

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

permet de comprendre la formation de clusters et la convergence du réseau dans des cadres stationnaires. Elle offre un langage commun avec les modèles de spins classiques et autorise l'utilisation de méthodes d'optimisation stochastique telles que le recuit simulé pour éviter le piégeage dans des minima locaux. Toutefois, cette approche reste partielle car, dans la majorité des scénarios réels du DSL, la synergie $S(i,j)$ est non linéaire et évolutive, ce qui entraîne une reconfiguration continue du paysage d'énergie et rend difficile l'application d'une descente d'un potentiel immuable. La convergence observée dans le DSL ne peut alors être expliquée que partiellement par la minimisation d'une fonction énergétique, et doit être complétée par une analyse qualitative de la dynamique adaptative du système.

2.4.4. Rôle de l'Inhibition et de la Saturation Synaptique

Dans un SCN (Synergistic Connection Network), la simple règle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ peut, dans certaines conditions, conduire à un **développement excessif** de liaisons (pour peu que $S(i,j)$ reste élevée) ou, au contraire, à l'émergence d'oscillations indésirables. Des **mécanismes** supplémentaires, tels que l'**inhibition** ou la **saturation** (limitation) des poids, viennent alors **réguler** la dynamique du réseau. La section (2.4.4) se focalise sur ce **rôle** de l'inhibition et de la saturation, en commençant (2.4.4.1) par une **comparaison** avec les modèles classiques d'**inhibition latérale** chez Amari ou Grossberg.

2.4.4.1. Comparaison avec les mécanismes d'inhibition latérale chez Amari ou Grossberg

Dans la littérature sur les réseaux neuronaux, les travaux d'**Amari** et de **Grossberg** constituent des références majeures pour comprendre comment un réseau peut instaurer une dynamique de **compétition** et de **sélection** via des mécanismes d'**inhibition latérale**. Ces modèles, qui sont généralement formulés en temps continu, mettent en œuvre des termes inhibiteurs permettant de restreindre l'activité collective et de favoriser l'émergence de structures distinctes dans le réseau. Il convient dès lors de comparer ces approches classiques avec celles du **Deep Synergy Learning (DSL)**, dans lequel les pondérations ω_{ij} évoluent de manière adaptative et peuvent intégrer des termes d'inhibition destinés à limiter la croissance excessive des liaisons.

Dans le modèle d'Amari, par exemple, l'évolution de l'activation u_i d'une unité est souvent décrite par une équation différentielle de la forme

$$\frac{du_i}{dt} = -u_i + \sum_j w_{ij} f(u_j) - \beta \sum_{j \neq i} g(u_j),$$

où w_{ij} représente le **couplage excitateur** entre les unités i et j , f est une fonction d'activation non linéaire, et le terme $-\beta \sum_{j \neq i} g(u_j)$ incarne l'**inhibition latérale**. Cette inhibition, par son action négative, réduit l'activation des unités voisines lorsqu'une unité donnée devient fortement active, aboutissant ainsi à un phénomène de **compétition** qui peut, par exemple, engendrer des zones d'activité distinctes dans une carte neuronale.

Grossberg, quant à lui, a développé une approche similaire en mettant en évidence la **compétition** entre neurones par le biais d'une répartition limitée de la ressource synaptique. Dans ses modèles, l'évolution de l'activation est soumise à un mécanisme de régulation tel que

$$\frac{du_i}{dt} = -\alpha u_i + \sum_j W_{ij} f(u_j) - \gamma \phi \left(\sum_j W_{ij} f(u_j) \right),$$

où γ et α sont des constantes positives, et le terme inhibiteur $-\gamma \phi(\cdot)$ permet de moduler l'activation en fonction de la somme des contributions des autres unités. Ce mécanisme, souvent qualifié de **winner-takes-most**, fait en sorte que seules quelques unités parviennent à conserver une activité significative tandis que les autres sont fortement inhibées.

Dans le **DSL**, l'évolution des pondérations ω_{ij} est régie par une mise à jour de type

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)],$$

où η est le **taux d'apprentissage**, τ le **coefficent de décroissance** et $S(i,j)$ la **synergie effective** entre les entités \mathcal{E}_i et \mathcal{E}_j . Pour intégrer un mécanisme d'**inhibition latérale** similaire à celui des modèles d'Amari et de Grossberg, il est possible d'enrichir la mise à jour en ajoutant un terme inhibiteur explicite, par exemple

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \left[S(i,j) - \tau \omega_{ij}(t) - \gamma \sum_{k \neq j} \omega_{ik}(t) \right],$$

où le terme $-\gamma \sum_{k \neq j} \omega_{ik}(t)$ représente une **pénalisation** de la croissance des autres pondérations sortantes du même noeud \mathcal{E}_i . Ce mécanisme incite un noeud à concentrer son potentiel de couplage sur un nombre limité de connexions plutôt que de diffuser de manière uniforme ses ressources synaptiques, imitant ainsi la dynamique d'inhibition observée dans les modèles continus d'Amari et Grossberg.

L'avantage principal de cette approche dans le DSL réside dans sa capacité à **sparsifier** le réseau. En effet, l'introduction de l'inhibition latérale force chaque entité à **sélectionner** les liaisons les plus pertinentes, ce qui conduit à la formation de **clusters** distincts et bien définis. Par analogie avec les réseaux de neurones biologiques, ce mécanisme assure que la compétition entre les connexions maintienne une **structure hiérarchisée** du réseau, dans laquelle seuls les liens les plus forts et les plus significatifs persistent, tandis que les connexions moins importantes sont naturellement atténuées.

D'un point de vue mathématique, la similitude avec les modèles d'Amari et Grossberg se reflète dans la forme des équations différentielles. Dans les deux cas, un **terme de décroissance** – qu'il

s’agisse de $-u_i$ dans l’équation d’Amari ou de $-\alpha u_i$ dans celle de Grossberg – joue un rôle fondamental en limitant la croissance des variables et en assurant la **stabilité** de la dynamique. De même, dans le DSL, le terme $-\tau \omega_{ij}(t)$ agit pour contrer l’augmentation induite par la synergie $S(i, j)$. La présence du terme inhibiteur supplémentaire $-\gamma \sum_{k \neq j} \omega_{ik}(t)$ introduit une compétition entre les liaisons sortantes d’un même nœud, ce qui est directement comparable à l’inhibition latérale classique.

En conclusion, la **comparaison** entre les mécanismes d’inhibition latérale d’Amari et de Grossberg et ceux du DSL met en lumière une convergence conceptuelle forte. Alors que les modèles classiques utilisent des termes d’inhibition pour restreindre l’activation globale des unités et favoriser l’émergence de structures distinctes, le DSL exploite des termes analogues dans la mise à jour de ses pondérations pour induire une **compétition** entre les connexions et ainsi favoriser la formation de **clusters** stables et sparsifiés. Cette approche, bien que formulée dans un cadre numérique et évolutif, reprend les principes fondamentaux de la **compétition neuronale** et offre un puissant levier pour la **stabilisation** et la **segmentation** du réseau.

Conclusion

Le parallèle établi avec les mécanismes d’inhibition latérale chez Amari et Grossberg démontre que l’ajout d’un terme inhibiteur dans la mise à jour des pondérations ω_{ij} du DSL constitue un régulateur efficace. Ce mécanisme, qui limite la somme des connexions sortantes d’un nœud et favorise la sélection de liens dominants, est en parfaite adéquation avec les principes biologiques d’inhibition latérale observés dans les réseaux neuronaux. Les analogies avec les modèles continus d’Amari et Grossberg offrent ainsi un cadre conceptuel robuste pour expliquer comment la **compétition** entre les liaisons favorise l'**auto-organisation** du réseau en clusters distincts et stables. Les développements ultérieurs exploreront de manière plus approfondie l’impact de ces mécanismes sur la stabilité globale et les analogies avec des modèles biologiques plus complexes.

2.4.4.2. Effet sur la stabilisation d’un nombre limité de clusters ou sur la rupture d’oscillations excessives

Dans le cadre du **Deep Synergy Learning (DSL)**, l’introduction d’un mécanisme d’**inhibition compétitive** joue un rôle fondamental en régulant l’évolution des pondérations ω_{ij} . Cette régulation vise à limiter le nombre de liaisons fortes qu’un même nœud peut entretenir, tout en empêchant l’apparition d’oscillations excessives dans la dynamique du réseau. L’objectif est double : d’une part, forcer la **sélection** de connexions significatives qui se regroupent en clusters distincts et, d’autre part, amortir les rétroactions positives qui pourraient conduire à des oscillations auto-entretenues, voire à des régimes pseudo-chaotiques.

Lorsque l’on considère un DSL sans mécanisme inhibiteur explicite, il est fréquent qu’une entité \mathcal{E}_i développe de fortes connexions avec un grand nombre d’autres entités \mathcal{E}_j si la **synergie** $S(i, j)$ reste suffisamment élevée pour chacune de ces paires. Un tel comportement peut conduire à la formation d’un réseau très dense, dans lequel presque toutes les entités sont fortement couplées, rendant floue l’identification de sous-groupes ou de clusters. Afin de remédier à ce phénomène, l’ajout d’un terme d’inhibition compétitive modifie la règle de mise à jour en intégrant une

pénalisation sur la somme des pondérations sortantes d'un nœud. Par exemple, la mise à jour peut être modifiée comme suit :

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \left[S(i,j) - \tau \omega_{ij}(t) - \gamma \sum_{k \neq j} \omega_{ik}(t) \right],$$

où $\gamma > 0$ représente le **paramètre d'inhibition**. Ce terme additionnel, $-\gamma \sum_{k \neq j} \omega_{ik}(t)$, exerce une pression négative sur la croissance simultanée de plusieurs liaisons issues d'un même nœud \mathcal{E}_i . Ainsi, lorsqu'une pondération $\omega_{ik}(t)$ croît de manière significative, la somme $\sum_{k \neq j} \omega_{ik}(t)$ augmente, et par conséquent, la progression de $\omega_{ij}(t)$ est freinée. Ce mécanisme force chaque entité à « choisir » un nombre limité de connexions fortes, ce qui conduit à la **formation** de clusters distincts et moins nombreux, facilitant ainsi l'**interprétation** de la structure du réseau.

D'autre part, dans les systèmes non linéaires où des rétroactions peuvent induire des oscillations ou des fluctuations exagérées, l'inhibition compétitive agit également comme un **stabilisateur** dynamique. En imposant une contrainte sur la somme totale des liaisons sortantes, le terme inhibiteur contribue à amortir les variations et à réduire la possibilité d'un renforcement en cascade des pondérations qui, autrement, pourrait entraîner des oscillations de grande amplitude. Du point de vue de l'analyse des systèmes dynamiques, cette inhibition se traduit par l'ajout d'un terme négatif dans la **Jacobienne** du système, diminuant ainsi la norme des valeurs propres et favorisant la convergence vers un attracteur stable. Ainsi, la dynamique globale se retrouve mieux équilibrée et la trajectoire des pondérations $\{\omega_{ij}(t)\}$ se stabilise, limitant les phénomènes de résonance excessive.

En synthèse, l'introduction de l'inhibition dans le DSL présente deux avantages majeurs. Premièrement, elle aboutit à une **sparsification** de la connectivité du réseau en contrignant chaque entité à ne maintenir que quelques liaisons fortes, ce qui rend la formation des clusters plus claire et la segmentation du réseau plus nette. Deuxièmement, elle contribue à **amortir** les oscillations potentielles, en introduisant un mécanisme de retour négatif qui empêche l'exacerbation des rétroactions positives, favorisant ainsi une **stabilité dynamique** accrue. Cette approche, qui s'inspire directement des modèles d'inhibition latérale chez Amari et Grossberg, permet d'obtenir un réseau auto-organisé présentant à la fois une structure hiérarchisée et une robustesse face aux fluctuations internes.

Conclusion

En incorporant des mécanismes d'inhibition compétitive – par exemple, en ajoutant un terme du type $-\gamma \sum_{k \neq j} \omega_{ik}(t)$ dans la règle de mise à jour – le DSL parvient à limiter la prolifération des liaisons fortes pour chaque entité, ce qui conduit à la formation d'un nombre restreint et bien défini de clusters. Par ailleurs, cette inhibition joue un rôle crucial dans la rupture des oscillations excessives en amortissant les rétroactions positives, ce qui stabilise la dynamique globale du réseau. Ces effets combinés assurent que la structure auto-organisée demeure à la fois **sparse** et **stable**, permettant ainsi une meilleure interprétation et une exploitation efficace des regroupements observés.

2.4.4.3. Discussion : analogies biologiques, applications en robotique sensorielle

L'approche du **Deep Synergy Learning (DSL)**, par l'intermédiaire de ses mécanismes d'inhibition et de saturation, trouve de fortes correspondances avec les observations faites dans la **biologie neuronale**. Dans le cortex sensoriel, par exemple, il est largement démontré que l'**inhibition latérale** joue un rôle déterminant dans la formation de **cartes corticales**. Les neurones fortement activés exercent une inhibition sur leurs voisins, ce qui permet d'isoler des zones d'activité et de créer des **patches** ou des **colonnes d'orientation** dans le cortex visuel. Ce phénomène est essentiel pour la **compétition neuronale** et la spécialisation fonctionnelle, des principes qui ont été théorisés par des auteurs tels qu'**Amari et Grossberg**. Ces travaux démontrent que la dynamique d'un réseau de neurones peut être modulée par des mécanismes inhibiteurs qui limitent la propagation de l'excitation et favorisent la sélection d'une **activité dominante** au sein de sous-groupes distincts.

Dans le DSL, la **mise à jour** des pondérations ω_{ij} se fait selon une règle qui, lorsqu'enrichie d'un terme inhibiteur tel que

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \left[S(i,j) - \tau \omega_{ij}(t) - \gamma \sum_{k \neq j} \omega_{ik}(t) \right],$$

permet de limiter la croissance simultanée de nombreuses connexions issues d'un même nœud \mathcal{E}_i . Ce terme inhibiteur $-\gamma \sum_{k \neq j} \omega_{ik}(t)$ reproduit en quelque sorte le principe de la **ressource synaptique limitée** qui est observé dans les systèmes biologiques : lorsqu'une synapse se renforce, les autres connexions du même neurone sont freinées, conduisant à une **sélection** des liens les plus pertinents. Ainsi, cette stratégie favorise l'émergence d'**assemblées neuronales** ou de **clusters** distincts, où seules quelques connexions dominantes persistent, tout en inhibant les connexions superflues.

Du point de vue des **applications en robotique sensorielle**, l'intérêt de ces mécanismes se manifeste de manière très concrète. Dans un système robotique intégrant de multiples capteurs – tels qu'une caméra, un sonar, un LiDAR ou encore des micros – chaque capteur peut être modélisé comme une entité \mathcal{E}_i au sein d'un SCN. La **synergie** $S(i,j)$ mesure alors la complémentarité ou la redondance entre deux capteurs. Sans un mécanisme d'inhibition, il est envisageable que le réseau attribue une importance excessive à plusieurs canaux simultanément, ce qui pourrait mener à une **fusion de données** confuse et à une surcharge informationnelle. L'inhibition permet, en revanche, de contraindre le système à privilégier un nombre restreint de connexions robustes – par exemple, en mettant en avant la combinaison d'une vision frontale et d'un microphone ambiant – tout en atténuant l'impact des capteurs moins pertinents à un moment donné. En outre, l'inhibition agit comme un **verrou** contre l'auto-renforcement excessif qui pourrait amplifier indéfiniment le bruit dans certains canaux, garantissant ainsi une **robustesse accrue** du traitement sensoriel.

En somme, cette **discussion** met en lumière l'importance de l'inhibition dans le DSL, tant sur le plan théorique que dans ses **applications pratiques**. L'inhibition, par son effet de **compétition** entre les connexions, permet de réduire la densité des liaisons et de favoriser une **structuration** en clusters, ce qui facilite la segmentation et la hiérarchisation de l'information. Ce mécanisme, qui s'inspire directement des observations biologiques – notamment celles concernant l'inhibition latérale dans le cortex – trouve également une application essentielle dans la robotique sensorielle,

où la gestion efficace des flux de données est cruciale pour une perception fiable et une prise de décision autonome.

Conclusion

Le rôle de l'**inhibition** et de la **saturation** dans un SCN n'est pas uniquement un outil algorithmique, mais il s'inscrit dans une tradition de modèles neuronaux visant à instaurer une compétition synaptique qui permet une **auto-organisation** sélective. Les analogies avec les mécanismes d'inhibition latérale observés dans le cortex et décrits par Amari et Grossberg offrent un cadre conceptuel robuste pour comprendre comment le DSL parvient à stabiliser un nombre limité de clusters tout en évitant les oscillations excessives. Par ailleurs, ces principes se retrouvent également dans des applications concrètes telles que la robotique sensorielle, où ils permettent de gérer de manière efficace et robuste des flux de données complexes et diversifiés. Ainsi, l'intégration de ces mécanismes dans le DSL contribue à la **robustesse** et à la **sélectivité** du réseau, assurant une organisation hiérarchisée et fonctionnelle similaire à celle observée dans les systèmes biologiques.

2.4.5. Approches Hybrides et Collaboration Interdisciplinaire

Tout au long de la section 2.4, nous avons souligné les **connexions** entre le DSL (Deep Synergy Learning) et divers champs : théorie des systèmes dynamiques (2.4.1), modèles de spin (2.4.2) et notion d'énergie potentielle (2.4.3), ou encore mécanismes d'inhibition/saturation (2.4.4). Ces croisements laissent entrevoir de **nombreuses pistes** d'approches dites "hybrides" ou inter-filières, mobilisant la **physique**, la **biologie**, l'**informatique** et l'**ingénierie**. La section 2.4.5 (Approches Hybrides et Collaboration Interdisciplinaire) discute d'abord (2.4.5.1) du **recuit simulé** et d'autres algorithmes inspirés par la **physique statistique** pour régler le SCN, avant de poser (2.4.5.2) des perspectives dans des champs variés comme l'écologie numérique ou les réseaux sociaux, puis de faire (2.4.5.3) une transition vers les chapitres 3, 4, 5 où les mises en forme plus "ingénierie" seront développées en détail.

2.4.5.1. Comment la physique statistique peut fournir des algorithmes inspirés du recuit (simulated annealing) pour régler le SCN

L'approche du **recuit simulé** en physique statistique repose sur une analogie avec la trempe des métaux, dans laquelle un matériau est chauffé à une température élevée afin de permettre aux atomes de se réarranger librement, puis refroidi lentement pour atteindre une configuration d'énergie minimale. Dans le contexte algorithmique, chaque configuration d'un système est associée à une fonction « énergie » $\mathcal{H}(\mathbf{x})$ et l'objectif est de minimiser cette énergie en effectuant des transitions aléatoires. La probabilité d'accepter une modification qui augmente l'énergie est donnée par la loi exponentielle

$$P(\Delta\mathcal{H}) \propto \exp\left(-\frac{\Delta\mathcal{H}}{T}\right),$$

où T représente la **température**. Lorsque T est élevé, le système explore de manière large l'espace des configurations, et à mesure que T diminue, les changements qui augmentent \mathcal{H} deviennent de moins en moins probables, permettant ainsi au système de se stabiliser dans un minimum d'énergie.

Dans le **Deep Synergy Learning (DSL)**, il est naturel de chercher à transposer ce principe au réglage des pondérations ω_{ij} du **Synergistic Connection Network (SCN)**. Pour ce faire, on définit une **pseudo-énergie** $\mathcal{J}(\Omega)$ fonctionnelle des pondérations, dont une expression heuristique pourrait être donnée par

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

où $S(i,j)$ désigne la **synergie effective** entre les entités \mathcal{E}_i et \mathcal{E}_j , et τ est un **coefficent de décroissance** qui sert de régularisateur pour éviter une croissance non bornée des poids. Dans un cadre idéal, si l'on considère que $S(i,j)$ est constant ou ne dépend pas fortement de ω_{ij} , la dynamique des pondérations peut être vue comme une **descente de gradient** de \mathcal{J} :

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)},$$

ce qui, en posant

$$\frac{\partial \mathcal{J}}{\partial \omega_{ij}} = -S(i,j) + \tau \omega_{ij},$$

se traduit par

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)].$$

Ce processus, qui dans ce cas particulier correspond à une descente d'énergie locale, permet d'expliquer la formation des clusters en considérant que le système tend à minimiser la fonction \mathcal{J} . L'injection de **bruit** dans ce mécanisme – par exemple en modifiant la mise à jour comme suit

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)} + \sigma \xi_{ij}(t),$$

où $\xi_{ij}(t)$ est une variable aléatoire et σ en contrôle l'amplitude – permet alors d'implémenter une procédure de **recuit simulé**. Dans cette approche, une température effective $T(t)$ (ou une amplitude de bruit $\sigma(t)$) est introduite et décroît progressivement. Au départ, le système, exposé à un niveau de bruit élevé, peut explorer divers minima locaux en acceptant occasionnellement des mouvements qui augmentent la pseudo-énergie \mathcal{J} . Au fur et à mesure que $T(t)$ décroît, ces mouvements deviennent moins probables et le système se stabilise dans une configuration optimisée.

Il convient de noter que, dans des scénarios plus complexes, où la synergie $S(i,j)$ dépend de manière non linéaire des pondérations ω_{ij} ou de paramètres externes, le paysage d'énergie $\mathcal{J}(\Omega)$ ne sera pas fixe mais évoluera dans le temps. Dans ce cas, l'application rigoureuse d'un algorithme de recuit simulé devient plus délicate, et la descente d'énergie n'est qu'une **approximation** de la

dynamique réelle du système. Cependant, même dans ces conditions, l'ajout contrôlé de bruit peut aider le SCN à **échapper** à des minima superficiels et à explorer un espace de solutions plus large.

Conclusion

En définitive, les algorithmes inspirés de la **physique statistique**, tels que le recuit simulé, fournissent un cadre puissant pour réguler la dynamique d'un SCN. En définissant une pseudo-énergie

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{ij} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{ij}^2 + \dots,$$

et en appliquant une mise à jour stochastique de type

$$\omega_{ij}(t+1) = \omega_{ij}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{ij}}|_{\omega_{ij}(t)} + \sigma(t) \xi_{ij}(t),$$

on parvient à introduire une **température** contrôlée qui, en diminuant progressivement, permet au réseau d'explorer l'espace des configurations tout en évitant le piégeage dans un minimum local non optimal. Cette approche, bien qu'elle repose sur l'hypothèse d'une synergie relativement stationnaire, représente un levier essentiel pour optimiser la structure des pondérations et, par extension, la formation de clusters dans le DSL.

2.4.5.2. Perspectives d'usage en écologie numérique, réseaux sociaux évolutifs, etc.

La démarche du **Deep Synergy Learning (DSL)**, qui s'appuie sur des concepts issus de la physique statistique tels que le recuit simulé, l'approche énergétique et les mécanismes d'inhibition, peut être étendue bien au-delà du domaine des réseaux neuronaux classiques ou de l'optimisation algorithmique. En effet, ces techniques trouvent également un écho dans des domaines où l'on traite de **réseaux dynamiques** et de systèmes multi-entités, notamment dans l'**écologie numérique** et les **réseaux sociaux évolutifs**. Cette section présente une analyse détaillée de la manière dont la vision du DSL peut contribuer à l'**organisation**, à l'**adaptation** et à la **convergence** de systèmes complexes, soumis à d'importantes évolutions structurelles et temporelles.

Dans le cadre de l'**écologie numérique**, on considère un ensemble d'entités logicielles – par exemple, des microservices, des agents autonomes ou des protocoles distribués – qui interagissent de manière continue pour former un écosystème virtuel. Chaque entité, notée \mathcal{E}_i , est associée à des caractéristiques spécifiques et à une dynamique de mise à jour de ses liens, représentés par des pondérations ω_{ij} . La **synergie** $S(i,j)$ entre deux entités évalue la valeur ajoutée de leur interaction, qu'il s'agisse d'un gain en performance, en complémentarité fonctionnelle ou en compatibilité technique. Dans ce contexte, le DSL permet de modéliser les interactions comme une forme d'**auto-organisation** où, par des mises à jour locales inspirées de mécanismes de recuit simulé, seules les connexions pertinentes se renforcent tandis que les autres s'affaiblissent progressivement. Par exemple, si l'on définit un opérateur de mise à jour du type

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)],$$

la dynamique résulte en la formation de sous-groupes d'entités fortement couplées, ce qui se traduit par l'émergence de **communautés** de microservices. La présence de mécanismes d'inhibition, qui

limitent la somme des pondérations sortantes d'un même nœud, assure une **sparsification** du réseau et empêche la sur-connexion généralisée. Ainsi, dans une plateforme distribuée, la structure auto-organisée obtenue facilite la coordination entre services, permettant une répartition efficace des tâches et une meilleure gestion des ressources, sans nécessiter de contrôleur centralisé.

D'un autre côté, dans le domaine des **réseaux sociaux évolutifs**, les utilisateurs – modélisés comme des entités \mathcal{E}_i – interagissent par le biais de liens dont l'intensité, représentée par ω_{ij} , reflète la fréquence ou la qualité des interactions (comme la communication, le partage de contenu ou l'affinité d'opinion). La **synergie** $S(i, j)$ peut être dérivée d'indices tels que la similarité des centres d'intérêt, le nombre de messages échangés ou encore la co-occurrence d'activités. En intégrant des mécanismes d'inhibition dans la mise à jour des liens, le DSL permet de modéliser la **sélectivité relationnelle** qui caractérise les réseaux sociaux : un utilisateur ne peut entretenir simultanément des connexions fortes avec un très grand nombre d'autres, ce qui conduit à la formation de **communautés** stables. La règle de mise à jour

$$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta \left[S(i, j) - \tau \omega_{ij}(t) - \gamma \sum_{k \neq j} \omega_{ik}(t) \right]$$

illustre comment le terme inhibiteur $-\gamma \sum_{k \neq j} \omega_{ik}(t)$ force l'utilisateur à concentrer ses interactions sur un sous-ensemble d'autres utilisateurs, reproduisant ainsi l'effet de **compétition synaptique** observé dans le cerveau. Ce mécanisme aide à prévenir la dispersion des ressources relationnelles et contribue à la formation de clusters sociaux, où les interactions internes sont beaucoup plus intenses que celles avec le reste du réseau.

Les **perspectives d'usage** du DSL dans ces domaines sont multiples. Dans l'**écologie numérique**, par exemple, l'auto-organisation des microservices peut permettre de construire des écosystèmes résilients où l'interopérabilité est constamment optimisée par des mises à jour locales. Chaque service ajuste ses connexions en fonction de la synergie mesurée avec d'autres services, et l'ensemble du système évolue vers une configuration optimale qui minimise les coûts de communication et maximise la performance globale. De même, dans les **réseaux sociaux évolutifs**, la dynamique du DSL peut aider à prédire et à comprendre l'évolution des communautés. La capacité du système à s'adapter à l'arrivée ou au départ d'utilisateurs, combinée aux mécanismes de recuit simulé et d'inhibition, permet d'obtenir une modélisation robuste de la formation, de la fusion ou de la fragmentation des groupes sociaux, tout en tenant compte des fluctuations inhérentes aux interactions humaines.

Enfin, l'approche DSL, grâce à sa **flexibilité** dans la définition de la fonction de synergie $S(i, j)$ et de la mise à jour des pondérations, offre un cadre général capable de s'adapter à de nombreux autres domaines. Que ce soit pour modéliser des réseaux d'innovation, des collaborations scientifiques ou des systèmes logistiques distribués, l'analogie avec les techniques d'**optimisation stochastique** et de recuit simulé reste pertinente. Les méthodes DSL permettent alors de gérer des **interactions** complexes et de produire des configurations stables dans un environnement évolutif, favorisant ainsi l'émergence de structures cohérentes à partir d'un ensemble initialement désordonné.

Conclusion

Les approches inspirées de la physique statistique, notamment le recuit simulé, confèrent au DSL une capacité interdisciplinaire remarquable. En appliquant ces techniques à des domaines tels que l'écologie numérique et les réseaux sociaux évolutifs, le DSL permet de modéliser l'auto-organisation de systèmes complexes où les entités interagissent de manière dynamique et adaptative. Dans ces contextes, la **flexibilité** du DSL se traduit par une mise à jour auto-adaptative des liaisons, capable de gérer les fluctuations inhérentes aux environnements évolutifs, tout en favorisant la formation de **clusters** robustes et pertinents. Cette vision ouvre la voie à de nombreuses applications concrètes, allant de la gestion distribuée des microservices dans le cloud aux mécanismes de segmentation et d'évolution des communautés dans les réseaux sociaux, en passant par des systèmes collaboratifs dans divers domaines d'innovation.

2.4.5.3. Transition vers les Chapitres 3, 4, 5, où l'on verra des mises en forme plus “ingénierie” du SCN

Dans les sections précédentes, nous avons établi un cadre théorique robuste pour le **Deep Synergy Learning (DSL)** en nous appuyant sur des analogies avec la **physique statistique**, les **modèles de spin** (tels que Ising, Potts ou Hopfield) ainsi que sur des concepts issus de la **théorie des systèmes dynamiques** et des **mécanismes d'inhibition** observés en neurosciences. La formulation du DSL repose sur une dynamique de pondérations $\{\omega_{ij}\}$ évoluant suivant une règle de mise à jour qui, dans sa version continue, s'exprime par l'équation différentielle

$$\frac{d\omega_{ij}}{dt} = \eta [S(i,j; \{\omega_{p,q}\}, \dots) - \tau \omega_{ij}],$$

où $\eta > 0$ désigne le **taux d'apprentissage** et $\tau > 0$ le **coefficient de décroissance**. Les développements théoriques des sections 2.4.1 à 2.4.5 mettent en exergue des idées telles que la **descente d'énergie** dans un paysage complexe, l'**inhibition compétitive** rappelant les mécanismes d'Amari et de Grossberg, ainsi que l'utilisation de techniques de **recuit simulé** pour éviter le piégeage dans des minima locaux. Ces concepts, tout en étant profondément ancrés dans des modèles théoriques multidisciplinaires, ouvrent la voie à une transposition vers une **implémentation opérationnelle**.

Il convient désormais de passer de cette **maquette conceptuelle** à une **ingénierie concrète**. C'est exactement l'objectif des Chapitres 3, 4 et 5, qui se proposent d'apporter une déclinaison pratique et opérationnelle de l'approche DSL. Plus précisément, le **Chapitre 3** détaillera la structuration du SCN en exposant les **structures de données** nécessaires pour gérer les matrices de pondérations $\{\omega_{ij}\}$ et les **algorithmes** de mise à jour, qu'ils soient de type **batch**, **en ligne** ou **stochastique**. On y développera notamment des stratégies pour la gestion de seuils (notion de ω_{\min}) et l'implémentation de mécanismes d'inhibition, en s'appuyant sur des formulations mathématiques précises.

Le **Chapitre 4** se concentrera quant à lui sur les aspects **logiciels** et **parallelisme computationnel**. Il exposera les environnements de développement adaptés – qu'il s'agisse de langages comme Python pour la flexibilité ou de C++/CUDA pour la performance – et décrira comment répartir le calcul sur plusieurs cœurs ou sur des architectures GPU. La modularisation des fonctions, par exemple la séparation entre le module de calcul de la synergie $S(i,j)$ et le module de mise à jour

des pondérations, sera abordée en détail, permettant ainsi d'envisager une implémentation robuste et scalable du SCN.

Enfin, le **Chapitre 5** portera sur la **validation** et l'**évaluation** des performances du SCN dans des scénarios réels. Il s'agira de définir des **critères de convergence** (par exemple, lorsque $\|\omega_{ij}(t+1) - \omega_{ij}(t)\|$ devient négligeable), d'évaluer la **cohésion** des clusters formés (via des indicateurs comme la modularité ou le ratio interne/externe) et de mener des expériences sur des jeux de données multimodaux ou issus de flux sensoriels. Ces tests permettront de vérifier la robustesse, la **résilience** face au bruit et la **scalabilité** des approches développées.

En résumé, le passage de la théorie à l'ingénierie concrète se matérialise par une série de développements progressifs. Le **Chapitre 2** a ainsi posé les **fondements** conceptuels et mathématiques du DSL, en s'appuyant sur des analogies interdisciplinaires. La transition vers les **Chapitre 3**, **Chapitre 4** et **Chapitre 5** marquera l'étape où ces idées abstraites se déclinent en algorithmes opérationnels, en architectures logicielles et en méthodologies d'évaluation pratiques, ouvrant la voie à des applications réelles dans des domaines aussi variés que la **robotique**, le **cloud computing**, l'**écologie numérique** ou les **réseaux sociaux évolutifs**. Ce cheminement vise à démontrer qu'un cadre théorique solide peut être traduit en solutions ingénierie efficaces, capables de gérer la complexité et la dynamique d'un système multi-entités en constante évolution.

2.5. Perspectives Historiques et Liens avec l'IA Moderne

Au terme de ce **Chapitre 2**, nous avons exploré les **fondements théoriques** du DSL (Deep Synergy Learning) : liens avec la théorie des systèmes dynamiques, rapprochements avec la physique statistique, mécanismes d'inhibition et de saturation, etc. Il est intéressant de **resituer** tout cela dans un **contexte historique** et de voir comment le DSL peut **converger** ou s'articuler avec les **approches** actuelles de l'IA (apprentissage profond, IA neuro-symbolique, robotique adaptative, etc.). La section (2.5) propose un **regard** à la fois **rétrospectif** (comment ces idées se sont forgées) et **prospectif** (comment elles se connectent à l'IA moderne), préparant ainsi la transition vers les chapitres plus “implémentation” et “application”.

2.5.1. Évolutions Récentes et Convergence avec le Deep Learning

Depuis la vague d'**apprentissage profond** (Deep Learning) qui domine l'IA contemporaine (vision, langage, RL, etc.), l'idée de laisser des **couches** apprendre automatiquement des **features** est devenu un paradigme central. Dans cette optique, on peut se demander si un **SCN** (Synergistic Connection Network) ne pourrait pas venir **compléter** ou **fusionner** avec des réseaux neuronaux profonds, de façon à gérer l'**auto-organisation** à un niveau plus abstrait.

2.5.1.1. Essor de l'apprentissage profond : extraire des features et laisser un SCN auto-organiser ces caractéristiques à un niveau plus abstrait ?

Dans le contexte actuel de l'**apprentissage profond**, il est devenu courant d'utiliser des réseaux neuronaux tels que les **CNN**, **RNN** ou **Transformers** afin d'extraire automatiquement des **représentations** riches à partir de données brutes telles que des images, des signaux sonores ou des textes. Ces réseaux, souvent désignés comme des **backbones**, produisent des embeddings de haute dimension, par exemple un vecteur $\mathbf{x}_i \in \mathbb{R}^{512}$ pour chaque donnée d'entrée. L'idée qui se dégage est de ne pas se contenter d'envoyer ces features directement dans un classifieur supervisé, mais d'exploiter l'**auto-organisation** d'un **Synergistic Connection Network (SCN)** pour traiter ces caractéristiques à un niveau d'abstraction supérieur.

Dans ce cadre, chaque entité \mathcal{E}_i issue du backbone représente un ensemble de features qui sont ensuite transmises au SCN. La **fonction de synergie** $S(i, j)$ quantifie la relation ou la **complémentarité** entre les représentations de deux entités \mathcal{E}_i et \mathcal{E}_j . À partir de là, les pondérations ω_{ij} évoluent selon une dynamique d'**auto-organisation** décrite par une règle de mise à jour (voir par exemple l'équation en temps discret) :

$$\omega_{ij}(t + 1) = \omega_{ij}(t) + \eta [S(i, j) - \tau \omega_{ij}(t)],$$

où $\eta > 0$ représente le **taux d'apprentissage** et $\tau > 0$ le **coefficent de décroissance**. Ce processus permet au SCN d'explorer l'espace latent des features et de faire émerger des **clusters synergiques** qui ne seraient pas immédiatement décelables via une simple classification. En d'autres termes, plutôt que de procéder à une classification rigide, le SCN se charge de **combiner** et **d'organiser** ces embeddings en regroupements cohérents qui révèlent des structures plus abstraites et potentiellement pertinentes pour l'application considérée.

Un schéma typique de pipeline illustrant ce processus s'articule ainsi :

$$\begin{aligned} \text{Backbone (ResNet, BERT, etc.)} &\rightarrow \text{Feature Embedding} \\ \rightarrow \text{SCN (pondérations } \omega_{ij} \text{ évolutives).} \end{aligned}$$

Dans ce pipeline, le module de **feature extraction** produit des représentations vectorielles de qualité, qui sont ensuite traitées par le SCN. Ce dernier module peut fonctionner de manière **complètement non supervisée**, en laissant la dynamique locale – régulée par la synergie $S(i, j)$ et éventuellement enrichie par des mécanismes d'inhibition et de recuit simulé – conduire à la formation de clusters. Alternativement, il peut coexister avec des modules supervisés qui valident la structure émergente des regroupements, offrant ainsi une approche hybride permettant de bénéficier à la fois de la puissance de l'**apprentissage profond** et de la capacité d'**auto-organisation** du SCN.

L'avantage principal de cette approche réside dans sa **flexibilité**. En effet, la capacité du SCN à se réorganiser dynamiquement permet d'extraire des **patterns** inattendus ou des regroupements latents, qui peuvent correspondre à des catégories ou des classes qui ne sont pas explicites dans les données d'origine. Par exemple, dans le traitement d'un flot d'images par un ResNet, le SCN pourrait découvrir des **proto-classes** ou des regroupements fondés sur des similarités spatiales et contextuelles que l'on ne retrouverait pas dans un classifieur standard. Cette capacité à explorer et à structurer l'espace latent des features offre une plus-value significative en termes de **représentation** et de **compréhension** des données, ouvrant la voie à des applications avancées en vision par ordinateur, en traitement du langage naturel et en robotique.

Historiquement, certains systèmes expérimentaux ont déjà exploité des **modules compétitifs** et des **couches hebbiennes** pour favoriser la spécialisation des neurones dans des architectures précurseurs du deep learning moderne. Le DSL reprend et étend ces idées en introduisant une dimension d'auto-organisation qui opère non seulement sur les activations, mais directement sur les **liaisons** entre entités. Ainsi, au lieu de modifier uniquement les sorties des neurones, le DSL permet aux poids ω_{ij} de s'ajuster pour favoriser des interactions de plus haut niveau, rendant possible l'émergence d'une **structure hiérarchisée** dans l'espace des features.

Conclusion

En conclusion, l'essor de l'**apprentissage profond** offre la possibilité de combiner les approches d'extraction de **features** à haute dimension avec un SCN capable d'**auto-organiser** ces caractéristiques en regroupements synergiques. Cette approche hybride permet de transcender la simple classification en exploitant une **dynamique** de pondérations ω_{ij} qui, grâce à des mécanismes inspirés du recuit simulé et des modèles de compétitions synaptiques, peut révéler des structures abstraites et des relations complexes au sein de l'espace latent. Les **chapitres suivants** (notamment les Chapitres 8 et 14) développeront des scénarios d'application concrets où cette fusion DSL + Deep Learning s'avère particulièrement efficace pour la classification multi-domaine, la détection de patterns temporels complexes et l'organisation de flux multimodaux, démontrant ainsi la pertinence de cette approche dans des domaines variés et en constante évolution.

2.5.1.2. Exemples précurseurs : modules compétitifs, couches “Hebbiennes” dans certaines architectures de recherche

Dans l'histoire de l'intelligence artificielle, bien avant l'avènement généralisé du **Deep Learning** tel qu'on le connaît aujourd'hui et avant la formulation explicite du **Deep Synergy Learning (DSL)**, de nombreux chercheurs ont exploré des mécanismes d'apprentissage non supervisé fondés sur la compétition et l'auto-organisation. Ces travaux précurseurs, qui incluent les **modules compétitifs** et les **couches Hebbiennes**, ont permis de poser les jalons d'une approche qui, dans sa version moderne, se retrouve incarnée dans le DSL. On peut ainsi considérer ces premières architectures comme des prototypes conceptuels ayant anticipé la logique d'un **Synergistic Connection Network (SCN)**.

Dans les années 1980 et 1990, le paradigme du **Competitive Learning** fut largement étudié. L'idée fondamentale était d'amener un ensemble de neurones, ou de prototypes, à « compétition » pour représenter des caractéristiques d'une donnée d'entrée. Les algorithmes tels que le **Learning Vector Quantization (LVQ)** reposaient sur le principe que, pour une donnée d'entrée \mathbf{x} , le prototype le plus proche (selon une métrique souvent euclidienne) recevait un renforcement de son vecteur de poids. Mathématiquement, la règle de mise à jour pour le prototype gagnant, disons \mathbf{w}_i , pouvait s'exprimer de manière simplifiée par

$$\Delta \mathbf{w}_i = \eta (\mathbf{x} - \mathbf{w}_i),$$

ce qui tend à aligner \mathbf{w}_i sur les vecteurs d'entrée similaires. Ce mécanisme, par son **aspect compétitif**, impose une différenciation entre les prototypes et favorise la formation de **clusters** d'entrées. La compétition ainsi instaurée conduit naturellement à une **auto-organisation** dans l'espace de représentation, préfigurant la dynamique d'un SCN où les pondérations entre entités s'ajustent en fonction d'une **synergie** mesurée.

Parallèlement, les **Self-Organizing Maps (SOM)** de Kohonen, introduites dans les années 1980, ont apporté une dimension topologique à cette idée. Dans un SOM, chaque neurone de la carte possède un vecteur de poids et la mise à jour se fait de manière à non seulement adapter le neurone gagnant à l'entrée, mais aussi à ajuster ses voisins dans le voisinage défini par un **noyau d'inhibition**. La règle de mise à jour pour un neurone voisin peut être formulée par

$$\Delta \mathbf{w}_j = \eta h(i, j) (\mathbf{x} - \mathbf{w}_j),$$

où $h(i, j)$ représente une fonction décroissante en fonction de la distance topologique entre le neurone gagnant i et le neurone j . Ce mécanisme permet de préserver la **cohérence spatiale** dans la carte, et il a servi de modèle pour comprendre comment un réseau peut se partitionner en régions distinctes d'activité, concept qui trouve une résonance dans la formation de clusters dans le DSL.

Le **postulat Hebbien** de Donald Hebb, formulé en 1949, constitue également une pierre angulaire dans cette évolution conceptuelle. Le célèbre adage « **Neurons that fire together, wire together** » a inspiré les premières règles d'apprentissage synaptique. La règle classique de Hebb, qui peut être écrite comme

$$\Delta w_{ij} \propto x_i x_j,$$

implique que si deux neurones sont activés simultanément, le lien synaptique entre eux se renforce. Bien que cette règle ne comporte pas de mécanisme intrinsèque de régulation (ce qui peut conduire

à une croissance indéfinie des poids), des modifications ultérieures, telles que l'ajout d'un terme de normalisation ou d'inhibition, ont permis de stabiliser ces dynamiques. Ces ajustements ont donné naissance à des **couches Hebbiennes** expérimentales dans certaines architectures de recherche, où l'idée de renforcer les connexions en fonction de la co-activation était exploitée pour générer des représentations auto-organisées des données.

Des approches connexes, telles que l'**Adaptive Resonance Theory (ART)** développée par Stephen Grossberg et Gail Carpenter, combinent des mécanismes de compétition, de vigilance et de reset pour permettre à un réseau de classifier de manière stable tout en intégrant de nouvelles informations sans oublier les anciennes. Les modèles ART illustrent comment la dynamique compétitive et les processus de **stabilisation** peuvent coexister dans des systèmes non supervisés, ce qui préfigure la logique d'un SCN dans le cadre du DSL.

En résumé, les **modules compétitifs** et les **couches Hebbiennes** ont constitué des **précurseurs** essentiels dans l'évolution de l'apprentissage non supervisé. Ces systèmes ont démontré que la **compétition** entre neurones – conduisant à la formation de clusters d'activité – ainsi que l'**apprentissage Hebbien** pouvaient engendrer une auto-organisation efficace des connexions. Le DSL s'inscrit dans cette lignée en généralisation, en permettant aux pondérations ω_{ij} d'évoluer selon une **fonction de synergie** $S(i, j)$ qui peut intégrer diverses mesures (distance, co-information, etc.) et en incluant des mécanismes de régulation tels que la **décroissance** (via un terme $\tau \omega_{ij}$) ou l'**inhibition compétitive**. Ainsi, l'approche DSL synthétise et étend les idées anciennes en les adaptant aux défis modernes de la **multimodalité** et de l'**apprentissage continu**, grâce notamment aux avancées récentes en calcul (GPU, big data) qui permettent de gérer des réseaux de grande dimension.

Conclusion

Les travaux antérieurs sur les **modules compétitifs** et les **couches Hebbiennes** offrent une base conceptuelle et expérimentale riche qui préfigure la logique du DSL. Ces architectures précurseurs, en instaurant une dynamique de renforcement des liens basée sur la co-activation et la compétition, ont démontré la faisabilité d'une auto-organisation des connexions dans un réseau. Le DSL généralise cette approche en permettant aux pondérations ω_{ij} de s'ajuster selon une **fonction de synergie** plus complexe, intégrant des aspects non linéaires et des mécanismes de régulation supplémentaires. En définitive, ces exemples historiques constituent un tremplin vers des architectures modernes où l'auto-organisation et la dynamique compétitive sont exploitées de manière plus fine, ouvrant la voie à des systèmes adaptatifs et multimodaux d'une grande robustesse et d'une forte capacité d'apprentissage.

2.5.2. Extensions Symboliques et Neuro-Symboliques

Dans la lignée des **perspectives historiques** (2.5.1), on constate que le **DSL** (Deep Synergy Learning) ne se limite pas à des entités purement numériques ou strictement sub-symboliques : le cadre théorique du DSL étant très **général**, il peut **intégrer** à la fois des représentations **logiques** (symboliques) et des composantes **sub-symboliques** (vecteurs, embeddings, etc.). La section (2.5.2) aborde ces **extensions symboliques et neuro-symboliques**, montrant que le DSL offre un terrain d'entente entre l'IA symbolique (règles, concepts, raisonnements) et l'IA sub-symbolique

(réseaux neuronaux, traitement statistique). Nous commençons (2.5.2.1) par rappeler que le DSL peut accueillir des entités logiques ou sub-symboliques en toute flexibilité.

2.5.2.1. Rappel : DSL peut accueillir des entités logiques ou des blocs sub-symboliques

Dans le cadre du **Deep Synergy Learning (DSL)**, la conception des « **entités** » n'est pas nécessairement limitée à des vecteurs continus appartenant à un espace \mathbb{R}^d . En effet, l'architecture d'un **Synergistic Connection Network (SCN)** se caractérise par sa grande **flexibilité** quant à la nature des entités qui le composent. Dès lors, il est possible d'intégrer dans un même réseau à la fois des **entités sub-symboliques** et des **entités logiques** ou des blocs **symboliques**, ce qui permet de traiter simultanément des représentations continues et des représentations discrètes.

Dans la perspective **sub-symbolique**, les entités \mathcal{E}_i sont généralement représentées par des **embeddings**, c'est-à-dire des vecteurs caractéristiques issus de réseaux profonds (tels que CNN, RNN, ou Transformers). Ainsi, une entité est formellement décrite par un vecteur

$$\mathbf{x}_i \in \mathbb{R}^d,$$

et la **synergie** entre deux entités, notée $S(i, j)$, est calculée à l'aide de mesures classiques telles que la **distance euclidienne** ou la **similarité cosinus**. Par exemple, une fonction de synergie peut être définie comme

$$S(i, j) = \exp(-\alpha \parallel \mathbf{x}_i - \mathbf{x}_j \parallel),$$

où $\alpha > 0$ est un paramètre de sensibilité. Cette formulation permet d'extraire des caractéristiques robustes issues de données brutes (images, sons, textes, etc.) et d'auto-organiser ces représentations en **clusters** par la suite.

D'un autre côté, le DSL permet également d'accueillir des **entités logiques** ou des **blocs symboliques**. Dans ce cas, une entité \mathcal{E}_i peut être définie non pas par un simple vecteur, mais par un objet conceptuel ou un ensemble de règles. Par exemple, on peut envisager que \mathcal{E}_i représente un **concept** ou une **proposition** formelle, telle que

$$\mathcal{E}_i = \text{"Si } A \text{ alors } B".$$

La fonction de synergie $S(i, j)$ dans ce contexte sera alors adaptée pour quantifier la **compatibilité** ou la **cohérence** entre deux ensembles de règles ou entre des concepts issus d'ontologies différentes. Une approche possible consiste à définir une distance sémantique ou une mesure de **similarité logique** basée sur la structure de l'ontologie. Par exemple, on pourrait utiliser une mesure de similarité définie par

$$S(i, j) = \sigma(\mathcal{E}_i, \mathcal{E}_j) \in [0, 1],$$

où σ est construite à partir d'une métrique de correspondance sémantique.

L'**intégration** de ces deux types d'entités dans un même SCN permet de bâtir un modèle **neuro-symbolique** puissant, capable de combiner les avantages du traitement continu (apprentissage à partir de données brutes, robustesse aux variations) et ceux du raisonnement symbolique (interprétabilité, manipulation explicite de concepts et règles). L'une des **forces** du

DSL réside dans sa capacité à utiliser une **fonction de synergie** $S(i, j)$ qui peut être spécifiquement adaptée en fonction de la nature des entités concernées. Ainsi, pour des entités sub-symboliques, on utilisera des mesures issues de la **géométrie de l'espace vectoriel** (distance euclidienne, similarité cosinus, etc.), tandis que pour des entités logiques, des techniques de **matching sémantique** ou des calculs de **coïncidence symbolique** seront privilégiées.

Cette approche hybride permet d'aboutir à une **auto-organisation** qui se fonde sur une **complémentarité** entre des représentations denses et continues et des représentations structurées et discrètes. Les pondérations ω_{ij} qui relient les entités dans le SCN s'ajustent ainsi en fonction d'une **synergie** multi-dimensionnelle, où chaque lien reflète à la fois une similitude perceptuelle et une compatibilité conceptuelle. L'ensemble du réseau, noté

$$\Omega = \{\omega_{ij}\},$$

se structure de manière à mettre en avant les connexions les plus **pertinentes** et à minimiser celles qui sont moins significatives, quelle que soit la nature des entités.

Conclusion

En résumé, le **DSL** se distingue par sa capacité à intégrer un **mélange** d'entités, aussi bien **sub-symboliques** que **logiques**. Cette flexibilité est précieuse car elle permet d'aborder des environnements **multimodaux** où coexistent des données continues (images, sons, textes) et des informations symboliques (catégories, règles, concepts). La **fonction de synergie** $S(i, j)$ est alors conçue pour s'adapter à ces différents types de représentations, assurant ainsi que le SCN puisse auto-organiser ses connexions en tenant compte à la fois des **aspects perceptifs** et **conceptuels**. Cette approche ouvre la voie à des applications neuro-symboliques avancées, où la **plasticité** du réseau exploite pleinement la richesse des données d'entrée pour produire des **clusters** d'entités qui reflètent des relations complexes et complémentaires.

2.5.2.2. Synergies entre IA symbolique, sous-ensembles de règles, et la plasticité d'un SCN

Dans cette section, nous explorons comment l'**IA symbolique** – qui s'appuie sur la manipulation de règles, de concepts et d'ontologies – peut interagir de manière fructueuse avec la **plasticité** inhérente à un **Synergistic Connection Network** (SCN), tel que présenté dans le cadre du **DSL** (Deep Synergy Learning). Cette synergie repose sur la capacité du SCN à adapter dynamiquement les pondérations ω_{ij} en fonction d'une fonction de synergie $S(i, j)$ adaptée aux différentes natures d'entités. Nous nous référerons ici aux idées développées dans les sections précédentes, notamment aux notions d'auto-organisation évoquées en (2.5.2.1) et aux fondements théoriques du DSL présentés dans les sections 2.3 et 2.4.

A. IA symbolique et “blocs de règles” : rappel et enjeux

Historiquement, l'**IA symbolique** s'est concentrée sur la manipulation de symboles, de règles formelles et de représentations logiques. Dans ce paradigme, une entité symbolique peut être un **bloc de règles** ou un **concept** défini par une structure logique, par exemple une proposition du type

if A then B .

Ces entités disposent d'une **force interne** basée sur la validité ou la cohérence de leurs règles et, lorsqu'elles interagissent avec d'autres blocs similaires, une mesure de **compatibilité** peut être définie. Par exemple, si deux ensembles de règles \mathcal{R}_1 et \mathcal{R}_2 se complètent, leur synergie peut être quantifiée par une fonction de similarité, que nous noterons $S_{\text{sym}}(i, j)$. Ce mécanisme est similaire aux systèmes d'inférence dans l'IA symbolique classique, où un moteur de règles applique des opérations logiques pour en déduire de nouvelles connaissances.

B. Plasticité d'un SCN : une approche dynamique de l'auto-organisation

Dans le cadre du DSL, les **entités** ne sont pas exclusivement sub-symboliques (comme les vecteurs $\mathbf{x}_i \in \mathbb{R}^d$ issus d'un réseau de neurones) ; elles peuvent également être des entités symboliques. La **fonction de synergie** $S(i, j)$ est alors conçue de manière à prendre en compte des mesures issues tant de la comparaison de vecteurs que de la compatibilité entre ensembles de règles. Par exemple, pour une entité symbolique \mathcal{E}_{sym} et une entité sub-symbolique \mathcal{E}_{sub} , on peut définir une synergie mixte

$$S(\mathcal{E}_{\text{sym}}, \mathcal{E}_{\text{sub}}) = f(\sigma(\mathcal{E}_{\text{sym}}, \mathcal{E}_{\text{sub}})),$$

où σ représente une métrique de compatibilité (telle qu'une distance sémantique ou une mesure d'inférence) et f est une fonction d'activation adaptée. L'idée est que si les données sub-symboliques confirment ou valident le contenu d'un bloc de règles, le lien ω_{ij} entre ces deux entités se renforce automatiquement, selon la règle de mise à jour DSL, telle que présentée dans la section (2.2.2) et analysée dans (2.3.1) et (2.4.1).

La **plasticité** du SCN se traduit par l'évolution dynamique des pondérations ω_{ij} qui se mettent à jour, par exemple, selon la règle additive

$$\omega_{ij}(t + 1) = \omega_{ij}(t) + \eta [S(i, j) - \tau \omega_{ij}(t)],$$

où η est le taux d'apprentissage et τ le terme de décroissance. Lorsque $S(i, j)$ est une fonction qui intègre à la fois des éléments symboliques et sub-symboliques, la dynamique de ω_{ij} permet une **auto-organisation** fine des entités en clusters, reflétant des affinités multiples.

C. Interactions entre les couches symbolique et sub-symbolique

L'une des idées centrales de l'approche neuro-symbolique est de permettre une **cohabitation** entre une couche symbolique, où résident des règles explicites et des concepts, et une couche sub-symbolique, issue des techniques d'apprentissage profond. Dans un SCN, cela se traduit par la coexistence d'entités \mathcal{E}_i qui peuvent être soit des vecteurs d'activation (issues d'un backbone de deep learning, comme un ResNet ou un Transformer), soit des blocs de règles ou des concepts logiques. Les liens entre ces entités sont modulés par une fonction de synergie $S(i, j)$ conçue pour comparer des types de données hétérogènes.

Par exemple, dans une application pratique, un bloc de règles décrivant des relations logiques (par exemple, la règle *if animal then vertebrate*) pourra interagir avec des embeddings issus d'images ou de textes. Si un grand nombre d'images correspondant à des animaux se retrouvent alignées avec ce bloc, la synergie mesurée augmentera, ce qui, selon la dynamique du DSL, renforcera la pondération ω_{ij} correspondante. Ainsi, le SCN pourra, de manière auto-organisée, former un cluster qui associe les représentations symboliques et sub-symboliques autour d'un thème commun.

Les avantages de cette approche résident notamment dans la **flexibilité** et la **richesse** de l'auto-organisation, car le réseau n'est pas contraint à une seule modalité de représentation. Comme indiqué dans la section (2.5.2.1), l'adaptation aux divers types d'entités permet d'étendre l'application du DSL à des environnements multimodaux, où coexistent des informations perceptuelles et des concepts explicites.

D. Applications et implications de la synergie neuro-symbolique

Les synergies entre IA symbolique et plasticité d'un SCN ouvrent de nombreux champs d'application :

- **Systèmes experts évolutifs** : Dans ces systèmes, les règles ne sont pas figées mais évoluent en fonction des données. Un SCN peut adapter dynamiquement les pondérations entre différentes règles en fonction de leur performance ou de leur pertinence dans un environnement changeant. Cette approche permet de résoudre le problème de la rigidité des systèmes experts classiques.
- **Raisonnement multimodal** : En intégrant des représentations issues à la fois de sources sub-symboliques (comme des images ou du texte) et symboliques (comme des ontologies ou des règles), le SCN facilite la formation de clusters hybrides capables d'exploiter des informations complémentaires. Par exemple, dans le domaine de la vision par ordinateur, un bloc de règles décrivant des relations spatiales pourrait être combiné avec des embeddings d'images pour améliorer la reconnaissance d'objets.
- **Systèmes neuro-symboliques unifiés** : L'approche permet de construire des architectures où la logique symbolique, avec ses avantages en termes d'interprétabilité et de raisonnement formel, est intégrée dans un cadre d'apprentissage profond, offrant ainsi des solutions plus robustes et adaptatives pour des tâches complexes.

Ces exemples illustrent comment la **plasticité** du SCN, combinée à une **fonction de synergie** capable de traiter des entités hétérogènes, peut conduire à une auto-organisation efficace des connaissances, permettant d'associer de manière dynamique des informations issues de différentes modalités.

Conclusion

Pour conclure cette sous-section, nous récapitulons les points essentiels en nous référant aux sections antérieures (notamment (2.5.2.1)) :

- **Auto-organisation hybride** : Le DSL permet d'accueillir simultanément des entités sub-symboliques (embeddings, vecteurs de features) et des entités symboliques (blocs de règles, concepts logiques). La fonction de synergie $S(i,j)$ est adaptée à chaque type d'entité, favorisant ainsi des liens pertinents entre elles.
- **Plasticité et adaptation** : La mise à jour des pondérations ω_{ij} via des mécanismes de descente de gradient (ou des versions stochastiques) permet au SCN de s'auto-organiser de manière dynamique. Ainsi, les blocs de règles peuvent se renforcer ou s'affaiblir en fonction de leur compatibilité avec les données sub-symboliques.
- **Perspectives neuro-symboliques** : Cette approche ouvre la voie à des systèmes d'IA capables de combiner le meilleur des deux mondes – la capacité d'apprentissage et d'adaptation des réseaux de neurones et la clarté du raisonnement symbolique – pour aboutir à des architectures plus complètes et robustes.

Nous verrons dans la section (2.5.2.3) comment ces idées seront approfondies dans des **chapitres futurs** (notamment les chapitres 5 et 13), afin de déployer des systèmes neuro-symboliques intégrés qui tirent pleinement parti de la complémentarité entre traitement symbolique et sub-symbolique dans le cadre du DSL.

2.5.2.3. Chapitres futurs (5, 13) où l'on détaillera la cohabitation symbolique-subsymbolique

Les développements présentés dans les sections (2.5.2.1) et (2.5.2.2) ont montré que le **DSL** (Deep Synergy Learning) se distingue par sa capacité à accueillir des entités de nature hétérogène, allant des représentations sub-symboliques (vecteurs, embeddings) aux entités symboliques (ensembles de règles, concepts logiques). Cette cohabitation est rendue possible par une fonction de synergie $S(i,j)S(i,j)$ capable d'évaluer, de manière adaptée, la compatibilité ou la complémentarité entre des entités aux représentations très différentes. Toutefois, afin de passer du cadre théorique à une application concrète et opérationnelle, il est nécessaire de préciser les modalités d'implémentation et d'intégration de ces idées dans un système complet. C'est exactement ce que viseront les **Chapitre 5** et **Chapitre 13** de notre ouvrage.

Dans le **Chapitre 5**, nous aborderons tout d'abord l'**architecture générale** du Synergistic Connection Network (SCN). Ce chapitre détaillera la manière dont les pondérations ω_{ij} sont structurées et stockées (par exemple sous forme de matrices denses ou creuses) ainsi que les différentes stratégies d'**initialisation** et de mise à jour. Nous y décrirons également comment les données d'entrée—qu'elles soient issues d'un réseau de neurones pré-entraîné (représentations sub-symboliques) ou de modules symboliques (ensembles de règles, ontologies)—sont intégrées dans le SCN. La mise en œuvre pratique des routines de calcul de la synergie, qui peut inclure des mesures de distance euclidienne, de similarité cosinus ou même des évaluations de compatibilité logique, sera explicitée à travers des exemples concrets et du pseudo-code. Ce chapitre servira ainsi de passerelle entre le concept théorique du DSL et sa traduction en algorithmes exécutables, en mettant particulièrement l'accent sur la cohabitation des entités symboliques et sub-symboliques au sein d'un réseau unifié.

Le **Chapitre 13** quant à lui se penchera sur la dimension plus globale et cognitive de l'approche, en explorant la perspective d'une **IA forte** ou d'un raisonnement cognitif avancé reposant sur une fusion neuro-symbolique. Ici, l'objectif sera d'étudier comment le SCN peut non seulement

organiser de manière auto-adaptative des représentations sub-symboliques mais aussi, de façon dynamique, pondérer et moduler des ensembles de règles ou des concepts logiques. Ce chapitre examinera, par exemple, comment la plasticité des pondérations $\omega_{i,j}$ peut servir à ajuster l'importance relative de certaines règles en fonction de leur confirmation par des données empiriques. Nous discuterons également de l'**ordonnancement** et de la **mise à jour** des blocs de règles en interaction avec les embeddings issus de données perceptuelles, et nous montrerons comment ces mécanismes peuvent contribuer à une meilleure cohérence globale du système. L'approche neuro-symbolique proposée vise à dépasser la dichotomie classique entre réseaux de neurones et systèmes experts, en intégrant ces deux dimensions dans une même architecture adaptative.

Pour résumer, ces chapitres futurs illustreront de manière concrète les points suivants, en référence aux concepts développés précédemment dans (2.5.2.1) et (2.5.2.2) :

- **Architecture et implémentation pratique (Chapitre 5)** : Nous détaillerons comment structurer les données Ω et les pondérations $\omega_{i,j}$, ainsi que les algorithmes de mise à jour adaptés aux scénarios hybrides où coexistent des entités symboliques et sub-symboliques. Des aspects tels que l'inhibition, le recuit simulé et les stratégies de gestion des clusters seront abordés pour permettre une auto-organisation efficace dans des contextes hétérogènes.
- **Perspectives cognitives et neuro-symboliques (Chapitre 13)** : Nous examinerons comment un SCN peut servir de colonne vertébrale à une architecture d'**IA forte**, où la logique formelle et le raisonnement symbolique se combinent avec la flexibilité et la plasticité des représentations neuronales. L'objectif sera d'illustrer, par des études de cas et des analyses théoriques, comment la fusion des deux approches peut conduire à une intelligence plus intégrée, capable de s'adapter à des environnements complexes et évolutifs.

En conclusion, le passage du cadre théorique présenté dans le **Chapitre 2** vers les chapitres ultérieurs (notamment **Chapitre 5** et **Chapitre 13**) marque une transition essentielle, de la **modélisation conceptuelle** à la **mise en œuvre opérationnelle**. Cette transition permettra de concrétiser l'idée que le DSL n'est pas seulement une abstraction mathématique ou un ensemble de principes inspirés de la physique statistique et des neurosciences, mais qu'il constitue également un outil puissant pour l'ingénierie de systèmes adaptatifs. Ces systèmes seront capables de gérer la cohabitation d'entités logiques et sub-symboliques de manière harmonieuse, ouvrant ainsi la voie à des applications dans des domaines variés tels que la robotique, les réseaux sociaux évolutifs, l'économie numérique, et bien d'autres. Les futurs chapitres détailleront ainsi les aspects pratiques de cette approche interdisciplinaire, en se référant explicitement aux travaux et concepts développés dans les sections précédentes, notamment (2.5.2.1) et (2.5.2.2).

2.5.3. Apport en Robotique et Contrôle Adaptatif

Les principes du **DSL** (Deep Synergy Learning) — basés sur la dynamique adaptative de liaisons $\{\omega_{i,j}\}$ et la formation de clusters en fonction de la **synergie** — ne se cantonnent pas aux systèmes algorithmiques ou cognitifs : ils trouvent également un **terrain d'application** dans la **robotique**. En particulier, la notion de **coordination sensorimotrice** peut être envisagée comme un **réseau**

d'entités (capteurs, actionneurs, modules décisionnels) dont les liens se renforcent ou s'affaiblissent selon la **complémentarité** et l'**efficacité** qu'ils offrent. La section (2.5.3) examine l'**apport** du DSL en robotique et en contrôle adaptatif. Nous commençons (2.5.3.1) par rapporter quelques **retours d'expériences** de laboratoires ayant exploré la synergie adaptive pour la **coordination sensorimotrice**.

2.5.3.1. Retours d'expériences dans des laboratoires : usage de la synergie adaptive pour la coordination sensorimotrice

Dans le domaine de la robotique sensorimotrice, où la coordination entre divers capteurs et effecteurs constitue un enjeu crucial pour l'adaptation en environnement dynamique, plusieurs laboratoires ont expérimenté des approches inspirées du DSL (Deep Synergy Learning). Ces études, que l'on retrouve en référence dans la section (2.5.3.1) et reliées aux concepts développés dans les sections antérieures telles que (2.5.2.1) et (2.5.2.2), mettent en lumière comment l'auto-organisation des pondérations $\omega_{i,j}$ peut être utilisée pour réguler de manière adaptive la coordination sensorimotrice.

A. Contexte général en robotique sensorimotrice

Dans une configuration robotique typique, un robot est équipé d'un ensemble hétérogène de capteurs — caméras, LiDAR, gyroscopes, microphones, etc. — ainsi que d'effecteurs (moteurs de roues, bras manipulateurs, etc.). Historiquement, la coordination de ces dispositifs s'appuyait sur des schémas centralisés ou des architectures modulaires fixes, qui ne prenaient pas toujours en compte la variabilité environnementale. Dans ce contexte, le DSL propose de considérer chaque capteur ou module comme une entité \mathcal{E}_i dont la relation avec un autre module \mathcal{E}_j est quantifiée par une pondération $\omega_{i,j}$. Cette pondération évolue selon la règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

décrise en détail dans les sections (2.3.1.1) et (2.3.1.2), et permet ainsi d'exploiter la **synergie adaptive** entre capteurs et effecteurs. Lorsque la synergie $S(i,j)$ entre deux modules est élevée—par exemple, lorsque deux caméras fournissent des informations complémentaires pour l'estimation de la profondeur ou pour la détection d'obstacles—la pondération $\omega_{i,j}$ se renforce, favorisant une fusion efficace des données. Inversement, si un capteur est perturbé ou fournit des informations moins fiables, son lien avec d'autres modules est progressivement atténué.

B. Exemples de laboratoires et de scénarios rapportés

(a) Coordination multi-capteurs (Laboratoire A)

Dans un laboratoire spécialisé en vision stéréoscopique, un SCN a été déployé pour coordonner plusieurs caméras disposées à différents angles ainsi qu'un système inertiel (IMU). Les expériences ont montré que lorsque la qualité de la caméra frontale se détériore (par exemple, à cause d'un éblouissement), le système réévalue dynamiquement les pondérations, renforçant le lien entre la

caméra latérale et l'IMU. Ainsi, la fusion visuelle se fait de manière adaptative, garantissant une estimation de la profondeur plus robuste. Ce mécanisme repose sur la même règle de mise à jour décrite en (2.3.1.1) et (2.5.3.1), où l'auto-organisation des $\omega_{i,j}$ permet de moduler la contribution de chaque capteur.

(b) Coordination bras manipulateur et retours tactiles (Laboratoire B)

Un autre laboratoire s'est intéressé à la manipulation d'objets par des robots équipés de capteurs tactiles. Dans ce contexte, la synergie entre les capteurs tactiles et le contrôle moteur du bras manipulateur a été mise en œuvre à l'aide d'un SCN. En cas de prise stable, la pondération entre le capteur tactile et le module de commande du bras augmente, tandis que des erreurs (glissement ou échec de préhension) entraînent une diminution de $\omega_{tactile,moteur}$. Ce processus adaptatif, qui permet d'auto-ajuster les stratégies de préhension, illustre bien comment la mise à jour des pondérations assure l'émergence de clusters de connexions robustes, comme évoqué dans les sections (2.5.3.1) et (2.5.2.2).

(c) Coordination dans un essaim de robots (Laboratoire C)

Dans le domaine de la **swarm robotics**, chaque robot est considéré comme une entité \mathcal{E}_i . Un SCN a été utilisé pour modéliser la coopération entre les robots, où la pondération $\omega_{i,j}$ reflète la capacité de deux robots à collaborer efficacement (par exemple, pour l'exploration ou le transport). Lorsqu'une coopération productive est détectée (par exemple, un échange réussi de signaux ou une mutualisation des ressources), les liens entre ces robots se renforcent. Inversement, des interférences ou des redondances conduisent à une diminution des pondérations. Cette approche décentralisée permet à l'essaim de s'organiser en clusters spécialisés sans qu'un contrôleur central ne soit nécessaire, ce qui renforce la flexibilité et la résilience du système.

C. Bilan des expérimentations : Impact sur l'adaptativité et la robustesse

Les expériences rapportées dans les laboratoires montrent plusieurs points essentiels :

- **Adaptation en temps réel** : Grâce à la règle de mise à jour, telle que rappelée en (2.3.1.1), le SCN ajuste continuellement les pondérations $\omega_{i,j}$ en fonction de la qualité perçue de la synergie $S(i,j)$. Ce mécanisme permet au système de se reconfigurer instantanément en réponse aux variations environnementales, qu'il s'agisse d'une dégradation du signal d'un capteur ou d'une nouvelle situation d'interaction entre modules.
- **Décentralisation** : L'approche DSL évite l'utilisation d'un contrôleur hyper-centralisé. Chaque entité, qu'elle soit un capteur, un effecteur ou un robot dans un essaim, ajuste localement ses liaisons. Cette autonomie locale favorise la formation d'un réseau d'interactions organiquement structuré, conformément aux principes développés en (2.5.2.1) et (2.5.2.2).
- **Robustesse face aux perturbations** : La capacité d'auto-organisation permet de réduire l'impact des défaillances locales. Par exemple, dans le laboratoire A, la diminution de la synergie d'un capteur dégradé se traduit par une baisse de sa pondération, limitant ainsi l'influence négative de ce capteur sur la fusion globale. De manière similaire, dans les

configurations en essaim (Laboratoire C), la redondance des liens garantit que la perte d'un robot ne perturbe pas l'organisation globale du réseau.

Ces observations, en écho aux concepts théoriques détaillés dans les sections (2.3.1) à (2.5.2.2), confirment l'intérêt d'une approche basée sur la **synergie adaptative** pour la coordination sensorimotrice en robotique.

Conclusion

Les retours d'expériences dans divers laboratoires illustrent clairement que l'**usage** du DSL pour la coordination sensorimotrice présente de nombreux atouts :

- Une **adaptation en temps réel** des liaisons $\omega_{i,j}$, assurée par la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ (cf. sections 2.3.1 et 2.5.3.1), permet de maintenir une fusion efficace des données issues de multiples capteurs.
- La **décentralisation** des décisions, où chaque entité ajuste localement ses connexions, évite une centralisation excessive et renforce la robustesse du système.
- La **robustesse** face aux perturbations, grâce à la capacité d'auto-organisation du SCN qui réoriente les synergies en fonction de la qualité des signaux, assure que même en présence de défaillances locales, le système conserve une performance satisfaisante.

Ces résultats expérimentaux confortent l'idée que la vision DSL, telle qu'exposée dans les sections précédentes (notamment 2.5.2.1 et 2.5.2.2), offre un cadre prometteur pour la coordination sensorimotrice en robotique. La suite de la discussion, dans la section (2.5.3.2) et dans des chapitres ultérieurs (comme le chapitre 9 pour l'apprentissage continu et le chapitre 11 pour la robustesse), étendra ces conclusions à des applications de robotique collaborative et de swarm robotics, démontrant ainsi la polyvalence et l'efficacité du DSL dans des contextes réels.

2.5.3.2. Potentiel pour gérer des flottes de robots, chacun se reliant localement, formant un SCN global

Dans le cadre de la robotique collaborative, le concept de **Deep Synergy Learning (DSL)** offre une perspective innovante pour orchestrer la coordination entre de nombreux agents autonomes. En effet, chaque robot, considéré comme une entité \mathcal{E}_i au sein du réseau, dispose de l'opportunité de modifier ses liaisons $\omega_{i,j}$ de manière autonome, suivant une règle de mise à jour telle que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ représente le **taux d'apprentissage** et $\tau > 0$ le **coefficent de décroissance**. La **synergie** $S(i,j)$ entre deux robots peut être définie à partir de critères variés, tels que la proximité géographique, la complémentarité des capteurs, ou encore la réussite des tâches coopératives. Cette approche, que l'on retrouve également dans la section 2.5.3.1 relative aux retours d'expériences en robotique sensorimotrice, permet à chaque robot d'ajuster localement ses connexions sans recourir à une supervision centrale.

Sur le plan théorique, l'architecture du SCN pour une flotte de robots repose sur la capacité de chaque agent à calculer sa **synergie locale** avec ses voisins immédiats, ce qui s'exprime par des mises à jour distribuées des pondérations $\omega_{i,j}$. Si la synergie est élevée entre deux robots, le poids associé se renforce, tandis qu'en cas de collaboration inefficace ou de divergence de mission, le poids se réduit. Ce mécanisme favorise l'émergence de **clusters** d'agents fortement connectés, illustrant une organisation auto-adaptative du réseau. Les aspects mathématiques de cette dynamique reposent sur des équations différentielles discrètes, et l'auto-organisation du système est comparable à une descente de gradient dans un espace de configuration défini par les pondérations, comme nous l'avons détaillé dans la section 2.5.1.1.

L'un des avantages majeurs de cette approche réside dans la **décentralisation** de la mise à jour. Chaque robot, en traitant localement l'information issue de ses interactions, contribue à la formation d'un **SCN global** sans intervention d'un contrôleur centralisé. Cette propriété est cruciale lorsque le nombre de robots est élevé, car elle permet d'éviter une surcharge computationnelle et favorise la **robustesse** du système. En effet, la défaillance d'un robot n'affecte que localement le réseau, et le système peut se réorganiser automatiquement pour compenser la perte, une caractéristique déjà évoquée dans la section 2.5.3.1.

Par ailleurs, cette méthode permet d'implémenter des mécanismes d'**apprentissage continu** et de **robustesse**, comme ceux détaillés dans les chapitres futurs, notamment le **Chapitre 9** sur l'apprentissage continu et le **Chapitre 11** sur la robustesse. En pratique, la synergie $S(i,j)$ peut être ajustée en temps réel en fonction de la performance des collaborations, de la qualité des données de capteurs ou d'autres critères d'interaction, de sorte que les pondérations $\omega_{i,j}$ évoluent de manière dynamique. Par exemple, si deux robots collaborent efficacement pour réaliser une tâche, leur pondération tend à converger vers une valeur élevée, ce qui renforce leur intégration dans un cluster. À l'inverse, une interaction moins productive entraîne une diminution de $\omega_{i,j}$, permettant ainsi au réseau de se réorganiser de façon adaptative.

L'architecture ainsi proposée se distingue par sa **scalabilité**. Chaque robot n'a besoin de gérer que ses interactions locales, souvent limitées à un voisinage restreint, ce qui réduit la complexité globale du système. De plus, en exploitant des algorithmes de mise à jour inspirés du **recuit simulé** et des **mécanismes d'inhibition** (voir sections 2.4.4 et 2.4.5.1), le SCN peut éviter de se figer dans un minimum local non optimal et favoriser l'émergence d'une organisation plus stable et cohérente. Ainsi, l'auto-organisation du réseau repose sur une dynamique où la **synergie locale** guide la formation de clusters qui, au niveau global, constituent une structure robuste et résiliente face aux variations de l'environnement ou aux perturbations.

Pour conclure, le **DSL** appliqué aux flottes de robots offre un cadre conceptuel et opérationnel permettant la gestion auto-adaptative de réseaux distribués. La mise à jour locale des pondérations, régulée par des mécanismes de **synergie**, **d'inhibition** et de **recuit**, conduit à l'émergence de clusters de coopération efficaces. Cette approche, qui se base sur les principes de la théorie des systèmes dynamiques et de l'apprentissage adaptatif, permet de coordonner des ensembles d'agents de manière décentralisée tout en garantissant une résilience et une adaptabilité indispensables dans des environnements complexes. Comme nous l'avons établi dans les sections 2.5.3.1 et 2.5.2.2, ainsi que dans les perspectives évoquées dans les chapitres futurs (**Chapitre 9** et **Chapitre 11**), cette méthode ouvre la voie à des applications robustes en robotique coopérative, tout en illustrant l'interdisciplinarité entre les approches mathématiques, physiques et biologiques dans la conception de systèmes intelligents.

2.5.4. Comparaison avec d'Autres Paradigmes (RL, GNN, etc.)

Les sections précédentes (2.5.1 à 2.5.3) ont montré que le **DSL** (Deep Synergy Learning) se prête à des **convergences** multiples : apprentissage profond, IA symbolique, robotique adaptative. Pour clore ce panorama, la section 2.5.4 compare brièvement le DSL à d'autres paradigmes très en vogue en IA moderne, notamment les **Graph Neural Networks (GNN)** et les approches de **Reinforcement Learning (RL)** ou de **clustering**. Nous commencerons (2.5.4.1) par situer le DSL vis-à-vis des GNN, avant de voir (2.5.4.2) ce qui le distingue par rapport aux algorithmes de clustering ou aux méthodes de RL, puis de conclure (2.5.4.3) en renvoyant à des analyses plus approfondies dans les prochains chapitres 6, 7, 8.

2.5.4.1. Où se situe le DSL vis-à-vis des Graph Neural Networks (GNN) ?

A. Rappel sur les GNN

Les **GNN** traitent un **graphe** d'entrée, où chaque noeud est associé à un **vecteur** (feature) et les arêtes peuvent porter un label ou un poids. On applique des “convolutions” ou des “aggregations” sur les voisins pour mettre à jour les **représentations** de chaque noeud. L'objectif est souvent de réaliser une **tâche** supervisée ou semi-supervisée (classification de noeuds, prédiction de liens, etc.). Les poids du réseau (couches GNN) sont **apris** par backpropagation, en prenant en entrée la structure du graphe et les features des noeuds.

Dans la plupart des GNN, la **topologie** du graphe est **fixée** (ou du moins pas totalement réinventée à chaque itération). On applique des couches de type GNN (GCN, GAT, etc.) pour extraire des embeddings plus riches de chaque noeud ou de l'ensemble du graphe.

B. DSL = évolution de la matrice de liaison

Le **DSL** fait évoluer la **matrice** $\omega_{i,j}$ (pondérations d'un SCN) en fonction d'une **synergie** $S(i,j)$. La structure du graphe n'est donc pas statique : les liens (et leurs intensités) peuvent se créer, se renforcer ou disparaître au fil du temps, selon la dynamique auto-organisée.

Au lieu de se baser sur un graphe fixe et d'y appliquer des “pass” de convolution (comme en GNN), le **DSL** vise la **construction** ou la **réorganisation** même de ce graphe. Il ne repose pas obligatoirement sur un objectif supervisé ; il peut fonctionner dans une **logique** auto-organisée ou faiblement supervisée.

Dans l'ensemble, on peut relever plusieurs points communs entre les GNN et le DSL. D'abord, les deux approches manipulent des entités (noeuds) ainsi que leurs liaisons (arêtes ou pondérations). Par ailleurs, il est possible d'associer des *features* spécifiques à chaque entité (ou noeud), de sorte que le réseau intègre des informations additionnelles sur ces entités.

Sur le plan des différences majeures, la première tient à la **structure** : les GNN partent d'un graphe dont la topologie est définie au préalable, tandis que dans le DSL, on fait évoluer (voire créer ou supprimer) les connexions $\omega_{i,j}$ au fil de l'exécution, selon la dynamique auto-organisée. La seconde différence concerne le **type d'apprentissage** : les GNN relèvent généralement d'un cadre supervisé (minimisation d'une perte via backpropagation), alors que le DSL met en œuvre des

mises à jour dites “locales”, le plus souvent sans supervision stricte. La troisième divergence réside dans l'**objectif** poursuivi : les GNN visent la réalisation d’une tâche précise (classification, régression, etc.), tandis que le DSL se consacre à l’auto-organisation d’un réseau d’entités en fonction de leurs synergies, sans impératif d’inférence ou de prédiction.

Conclusion

Le **DSL** et les **GNN** partagent l’idée de **modéliser** un graphe, mais s’en distinguent par la **dynamique adaptative** des liens (pondérations $\omega_{i,j}$) et la **nature** (souvent auto-organisée) du DSL. Là où un GNN suppose la structure essentiellement *donnée*, le DSL la *construit/transforme* en continu selon la synergie. On peut néanmoins imaginer des **approches hybrides** (un GNN pour la propagation de features + un DSL pour ajuster la topologie du graphe), suggérant une piste de recherche combinant le meilleur des deux mondes.

2.5.4.2. Qu’apporte-t-il de distinct par rapport aux algorithmes de clustering traditionnels ou aux approches de renforcement ?

Le **Deep Synergy Learning (DSL)** se présente comme une approche qui auto-organise un réseau de pondérations $\{\omega_{i,j}\}$ au fil d’une dynamique évolutive, en mettant à jour continuellement chaque connexion selon la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ désigne le **taux d’apprentissage**, $\tau > 0$ le **coeffcient de décroissance**, et $S(i,j)$ représente la **synergie** mesurant l’affinité locale entre les entités. Cette formulation, qui a été détaillée dans les sections précédentes telles que **2.5.1.1** et **2.5.3.1**, contraste fortement avec les approches traditionnelles de clustering et de renforcement.

D’un côté, les **algorithmes de clustering** traditionnels comme **k-means** ou **DBSCAN** procèdent généralement en mode *batch*. Par exemple, dans k-means, chaque point de données est assigné au centroïde le plus proche, puis les centroïdes sont recalculés par minimisation de la somme des distances intra-cluster. Cette procédure itérative se poursuit jusqu’à convergence, et le nombre de clusters, k , est fixé a priori. Ces méthodes reposent sur une mesure de distance statique, ce qui aboutit à une partition fixe de l’ensemble des données. En revanche, dans le DSL, les pondérations $\omega_{i,j}$ évoluent en temps réel en réponse à la synergie locale entre entités, sans imposer une structure de partition rigide. Ainsi, les clusters dans le DSL apparaissent de manière émergente et évolutive, permettant à une même entité de maintenir des connexions significatives avec plusieurs groupes au lieu d’être confinée à une unique partition déterminée à l’avance.

D’un autre côté, les approches de **Renforcement (RL)** se fondent sur un cadre dans lequel un agent, en interaction avec un environnement, reçoit un état s_t , choisit une action a_t et perçoit une récompense r_t . L’objectif est d’apprendre une politique $\pi(a | s)$ qui maximise la récompense cumulative sur le long terme. Les algorithmes de RL, tels que le Q-learning ou les méthodes de policy gradient, se concentrent donc sur la prise de décision séquentielle guidée par un signal de récompense explicite. Dans le DSL, la mise à jour des pondérations est déterminée par la **synergie** $S(i,j)$ locale et n’implique pas de signal de récompense global ni de choix d’actions dans un espace d’états. Par conséquent, le DSL s’inscrit dans une dynamique d’**auto-organisation** où chaque entité

ajuste ses connexions de façon autonome, en se basant sur des critères de compatibilité mesurés localement, plutôt que d'optimiser une fonction de récompense cumulative.

Les **avantages** du DSL résident dans sa **flexibilité** et sa **capacité adaptative**. En effet, la mise à jour continue des $\omega_{i,j}$ permet au système de s'ajuster en temps réel aux variations des données, contrairement aux méthodes de clustering traditionnelles qui opèrent sur un ensemble de données fixe. De plus, le DSL ne se contente pas de partitionner l'espace des données ; il permet à une entité d'entretenir des liens multiples, reflétant ainsi la complexité des interactions dans des environnements réels. Du côté des approches de renforcement, le DSL offre une méthode décentralisée d'auto-organisation sans nécessiter une modélisation explicite des récompenses, ce qui peut simplifier l'implémentation dans des systèmes multi-agents où la coordination se fait par des interactions locales plutôt que par des signaux de récompense centralisés.

Cependant, cette **vision énergétique** et adaptative présente également des limites. Lorsque la synergie $S(i, j)$ varie de manière significative au fil du temps ou dépend fortement des pondérations elles-mêmes, la fonction potentielle globale ne reste plus statique et l'interprétation comme une descente de gradient devient moins évidente. Par ailleurs, dans des environnements hautement dynamiques, les mises à jour continues peuvent conduire à des oscillations ou à une instabilité qui ne sont pas facilement expliquées par les méthodes de clustering classiques ou par les algorithmes de renforcement qui, eux, reposent sur des critères de performance cumulés. De plus, la fusion des deux paradigmes (clustering statique et apprentissage par renforcement) est envisageable, mais elle nécessite une conception hybride qui peut complexifier la mise en œuvre du système.

En conclusion, le **DSL** se distingue à la fois des algorithmes de clustering traditionnels et des approches de renforcement par sa capacité à produire une **auto-organisation évolutive** du réseau de pondérations. La dynamique continue de mise à jour, basée sur la synergie locale, permet de former des clusters adaptatifs sans imposer de partitions fixes, tout en se passant d'un cadre décisionnel centré sur la maximisation de récompenses cumulées. Cette approche représente une alternative robuste et flexible, particulièrement adaptée aux systèmes complexes et distribués où les interactions entre entités sont non statiques et évolutives. Nous verrons, dans les chapitres **6**, **7** et **8**, comment ces principes sont intégrés dans des architectures d'IA avancées, démontrant ainsi la complémentarité et la richesse du DSL par rapport aux méthodes classiques.

2.5.4.3. Discussion brève avant de laisser place à l'analyse plus poussée dans les chapitres **6**, **7**, **8**

Dans les sections **2.5.4.1** et **2.5.4.2**, nous avons établi des comparaisons détaillées entre le **DSL** (Deep Synergy Learning) et divers paradigmes traditionnels, tels que les **Graph Neural Networks (GNN)**, les méthodes classiques de **clustering** et les approches de **Renforcement (RL)**. Il apparaît clairement que le DSL occupe un **créneau distinct** puisqu'il se concentre sur la **dynamique adaptative** des connexions $\omega_{i,j}$ qui se mettent à jour continuellement en fonction de la **synergie** locale, sans imposer de schéma de supervision tel que requis dans les GNN, ni de partition fixe comme dans k-means, et sans recourir à un signal de récompense global comme c'est le cas en RL. Cette caractéristique intrinsèque permet, en outre, de combiner le DSL avec ces paradigmes afin de renforcer leurs capacités respectives, par exemple en ajustant dynamiquement la structure d'un

graphe dans un GNN ou en apportant une dimension d'auto-organisation dans un cadre de RL multi-agents.

La suite de notre ouvrage se poursuivra avec des développements plus approfondis dans les chapitres **6**, **7** et **8**, qui visent à clarifier et à intégrer ces notions dans des systèmes d'intelligence artificielle complexes. Le **Chapitre 6**, intitulé « **Apprentissage Synergique Multi-Échelle** », traitera de la capacité du DSL à gérer plusieurs niveaux ou granularités d'entités, en mettant en lumière la manière dont la mise à jour des $\omega_{i,j}$ peut être effectuée dans une logique multi-échelle. Dans ce contexte, certains rapprochements avec les GNN et d'autres structures hiérarchiques seront approfondis, permettant ainsi de comprendre comment une architecture en couches peut émerger de l'auto-organisation des liens.

Le **Chapitre 7**, sous le titre « **Algorithmes d'Optimisation et Méthodes d'Adaptation Dynamique** », présentera des techniques issues du recuit simulé, de la compétition inhibitrice (cf. section **2.4.4**) ainsi que des approches inspirées de la physique statistique (cf. section **2.4.5.1**) pour optimiser la configuration d'un SCN. Ce chapitre abordera également les questions de **complexité** et de **scalabilité** lorsque le nombre d'entités augmente, en proposant des solutions pour assurer la convergence et la robustesse du système dans un environnement dynamique.

Le **Chapitre 8**, intitulé « **DSL Multimodal : Fusion de la Vision, du Langage et des Sons** », se focalisera sur les aspects pratiques de la fusion de données issues de différentes modalités. Ici, le DSL sera confronté aux approches traditionnelles de clustering multimodal et de RL multimodal, et l'on examinera comment la **synergie adaptative** entre divers types de données (images, textes, signaux audio) peut être exploitée pour former des représentations communes, enrichissant ainsi les performances du système global.

En guise de **conclusion** de cette discussion préliminaire, il apparaît que la dynamique du DSL se distingue des autres méthodes par sa capacité à organiser un réseau de pondérations de manière **auto-évolutive**. Ainsi, tandis que les GNN reposent sur une structure de graphe relativement fixe et un apprentissage supervisé, le DSL met en œuvre une **mise à jour continue** des $\omega_{i,j}$ dans un contexte non supervisé. De même, les méthodes de clustering traditionnelles segmentent un ensemble de données statique, et les approches de RL se concentrent sur l'optimisation séquentielle d'actions via une récompense cumulative, alors que le DSL privilégie l'**auto-organisation** des relations sans imposer de partition ni de politique d'actions prédéfinie.

Les chapitres **6**, **7** et **8** approfondiront ces **intégrations** et complémentarités, démontrant comment le SCN peut s'insérer dans des architectures d'IA plus larges et plus riches. Ce passage de la théorie à l'ingénierie concrète ouvre ainsi la voie à des systèmes d'IA capables de combiner de manière harmonieuse les paradigmes de **clustering**, de **renforcement** et de **modélisation graphique**, contribuant à l'émergence de solutions adaptatives et évolutives dans des environnements complexes.

2.5.5. Transition vers les Chapitres Suivants

L'ensemble de la section 2.5 (Perspectives Historiques et Liens avec l'IA Moderne) a révélé comment le **DSL** (Deep Synergy Learning) s'inscrit dans la continuité de plusieurs **courants** de l'IA : l'apprentissage profond (2.5.1), la neuro-symbolique (2.5.2), la robotique sensorimotrice

(2.5.3), et la comparaison avec GNN/clustering/RL (2.5.4). Pour conclure ce chapitre 2, la section (2.5.5) annonce la **transition** vers les chapitres plus concrets de la suite — notamment le **Chapitre 3**, qui portera sur la **représentation concrète** des entités et la définition pratique de la **synergie**, ainsi que les **Chapitres 4 et 5**, où l'on traitera davantage la **dynamique d'auto-organisation** et l'**architecture SCN** dans un cadre d'ingénierie.

2.5.5.1. Le prochain chapitre (3) traitera de la représentation concrète des entités et de la façon de définir la fonction de synergie dans divers contextes (image, texte, capteurs)

Dans la suite de ce travail, nous nous attacherons à passer d'une conceptualisation théorique du **DSL** (Deep Synergy Learning) à une mise en œuvre plus opérationnelle, en particulier en ce qui concerne la représentation concrète des **entités**. Le **Chapitre 3**, qui figure dans la table des matières, abordera de manière détaillée la problématique de la formalisation de chaque entité \mathcal{E}_i . Nous examinerons notamment si ces entités doivent être considérées comme des **vecteurs**—par exemple, des embeddings issus d'un réseau de neurones convolutionnel (CNN) ou d'un Transformer appliqué à des textes—ou si elles peuvent également prendre la forme de structures symboliques, telles que des ensembles de règles ou des ontologies. Dans ce contexte, il sera essentiel d'identifier les **structures de données** appropriées pour stocker et manipuler ces représentations, qu'il s'agisse de tableaux multidimensionnels, de graphes ou d'autres structures adaptées à la nature et à la dimension des descripteurs.

Un point central de cette démarche réside dans la définition de la **fonction de synergie** $S(i, j)$. Il s'agira d'établir des formules qui quantifient la similarité ou la complémentarité entre deux entités, en fonction du contexte considéré. Par exemple, dans le cas d'images, on pourra adopter une mesure basée sur la **distance cosinus** ou la distance euclidienne entre des embeddings extraits par un CNN, alors que pour des textes, des mesures de **similarité sémantique** issues de modèles de langage pré-entraînés (tels que BERT) seront privilégiées. Pour des capteurs, la synergie pourra être définie par des fonctions mesurant la cohérence temporelle ou la corrélation des signaux, telles que $S(i, j) = \text{corr}(\mathbf{x}_i, \mathbf{x}_j)$. Dans tous ces cas, la mise en œuvre devra garantir que la dynamique de mise à jour des pondérations $\omega_{i,j}$ du **SCN** (Synergistic Connection Network) soit adaptée à la nature des données, ce qui implique la prise en compte de paramètres tels que le taux d'apprentissage η et le coefficient de décroissance τ dans la règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)].$$

Cette équation, qui sera étudiée en détail dans les chapitres ultérieurs, devra être adaptée pour prendre en compte la diversité des types d'entités présentes dans le réseau.

Nous verrons également comment intégrer la **fonction de synergie** dans un cadre multimodal, où il est nécessaire de fusionner des informations provenant de sources très hétérogènes. Par exemple, dans un scénario de fusion de données, les représentations d'images et de textes devront être combinées au sein du même SCN, ce qui impliquera de définir des mécanismes permettant de normaliser et de pondérer les différentes mesures de synergie selon leur échelle et leur pertinence relative. Ce défi technique est essentiel pour assurer que la dynamique du DSL reste cohérente et exploitable, même lorsque les entrées proviennent de domaines très différents.

Le **Chapitre 3** constituera ainsi une passerelle entre la théorie développée dans le Chapitre 2 et les applications pratiques, en présentant des *pseudo-codes* et des exemples concrets d'implémentation. Il expliquera comment les entités, qu'elles soient purement sub-symboliques ou de nature mixte (symbolique et sub-symbolique), peuvent être représentées par des structures de données adaptées et comment la fonction de synergie $S(i, j)$ est définie pour piloter la dynamique d'auto-organisation des pondérations.

Par ailleurs, la suite de la transition est explicitée dans les sections **2.5.5.2** et **2.5.5.3**, qui détailleront respectivement les aspects de la dynamique d'auto-organisation du SCN dans des contextes appliqués et la cohabitation des idées théoriques avec des principes d'ingénierie. Le **Chapitre 3** sera suivi par les **Chapitres 4 et 5** qui s'orienteront vers la mise en œuvre effective des algorithmes et la validation de la dynamique auto-organisée dans des scénarios réalistes. Cette progression permettra de passer de la conceptualisation abstraite des mécanismes du DSL à une réalisation pratique, en mettant l'accent sur l'extraction des **features** et l'adaptation de la fonction de synergie aux caractéristiques des données, qu'elles soient issues de l'imagerie, du traitement de textes ou des flux sensoriels.

En conclusion, ce prochain chapitre se propose de formaliser la représentation concrète des entités ainsi que la définition précise de la fonction de synergie, ce qui est indispensable pour assurer que le SCN opère de manière optimale dans divers contextes d'application. Les résultats présentés dans ce chapitre jettent les bases pour les développements techniques approfondis qui suivront dans les chapitres **4** et **5**, lesquels se consacreront à la dynamique d'auto-organisation et à l'implémentation logicielle du DSL dans des environnements réels, tels que la robotique sensorimotrice et les systèmes multimodaux.

Références internes : cette section s'inscrit dans la continuité des sections **2.5.5.2** et **2.5.5.3**, lesquelles orientent respectivement vers la dynamique d'auto-organisation et la transition vers les chapitres suivants.

2.5.5.2. Les chapitres 4 et 5 aborderont la dynamique d'auto-organisation plus appliquée, en tirant parti des bases mathématiques introduites ici

Dans ce passage du Chapitre 2 aux chapitres suivants, nous avons posé des fondations théoriques solides pour le **DSL** (Deep Synergy Learning) en définissant précisément ses concepts clés, tels que la dynamique des pondérations $\omega_{i,j}$, les analogies avec la physique statistique, la théorie des systèmes dynamiques et les mécanismes d'**inhibition**. Cependant, il apparaît désormais nécessaire de transposer ces principes abstraits dans un cadre d'**implémentation concrète** afin de déployer le DSL dans des applications réelles. C'est à ce titre que les **Chapitres 4 et 5** interviendront, après que le **Chapitre 3** aura défini la représentation des entités (issues d'images, de textes, de capteurs, etc.) ainsi que la manière de formaliser la fonction de **synergie** $S(i, j)$.

Dans le **Chapitre 4**, intitulé "**Synergies Émergentes et Auto-Organisation**", l'objectif sera d'approfondir la manière dont la dynamique d'auto-organisation se manifeste au sein d'un SCN (Synergistic Connection Network). Nous passerons d'équations abstraites, telles que

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

à des algorithmes précis et des schémas de mise en œuvre qui permettent de maîtriser et d'exploiter les synergies issues des interactions locales entre entités. La mise en œuvre algorithmique sera détaillée à travers des pseudo-codes qui illustreront la gestion du pas de temps, l'initialisation des pondérations et les stratégies de détection ainsi que de régulation d'oscillations (par exemple, via des mécanismes d'inhibition et de découpage en lots). Ces exemples concrets permettront de comprendre comment, itération après itération, des clusters émergents ou des micro-réseaux stables se forment naturellement à partir de mises à jour locales. Les concepts introduits dans les sections **2.3** et **2.4** – tels que la théorie des attracteurs, la fonction d'énergie potentielle définie en **2.4.3**, et les mécanismes d'inhibition présentés en **2.4.4** – seront mobilisés pour expliquer en pratique comment ajuster les paramètres η et τ pour obtenir une dynamique auto-organisée prévisible et ajustable.

Le **Chapitre 5**, intitulé "**Le Synergistic Connection Network (SCN) : Architecture Générale**", portera sur la conception complète du réseau. Ce chapitre s'attardera sur la construction de l'architecture du SCN en abordant les questions de stockage et d'organisation des pondérations $\omega_{i,j}$. Nous y discuterons des différentes structures de données susceptibles d'être utilisées (matrices denses versus structures creuses, listes d'adjacence, etc.), ainsi que de l'organisation hiérarchique du réseau, qui pourra être structurée en différentes couches ou niveaux de synergie (par exemple, un niveau local contrasté avec un niveau global). En outre, l'architecture du SCN sera décomposée en plusieurs modules fonctionnels, notamment le module de calcul de la synergie $S(i, j)$, le module de mise à jour des pondérations (intégrant les mécanismes d'inhibition et de saturation), le module de clustering pour extraire des groupes d'entités, et enfin une interface dédiée à l'ingestion de données en temps réel dans le cas d'un SCN évolutif. Des exemples d'implémentation seront fournis, avec des pseudo-codes illustrant un framework léger, susceptible d'être réalisé en Python ou en C++ couplé à CUDA pour des applications nécessitant un calcul parallèle.

Ces deux chapitres, **4** et **5**, constituent ainsi un pont essentiel entre la théorie développée dans le Chapitre 2 et la phase d'**implémentation pratique** qui sera amorcée au **Chapitre 3**. La logique de progression consiste d'abord à définir concrètement la représentation des entités et la fonction de synergie dans le Chapitre 3, puis à explorer dans le Chapitre 4 la dynamique auto-organisée par les mises à jour locales, et enfin à concevoir une architecture complète pour le SCN dans le Chapitre 5. Cette transition vers l'ingénierie concrète permet de passer des principes mathématiques et théoriques à une réalisation opérationnelle, qui sera ultérieurement enrichie par des approfondissements sur le multi-échelle, l'optimisation avancée et l'intégration multimodale dans les chapitres **6**, **7** et **8**. Par ailleurs, les chapitres **9** et **10** aborderont l'apprentissage continu et le feedback coopératif, renforçant ainsi la robustesse globale du système.

Conclusion :

Les **Chapitres 4** (synergies émergentes, auto-organisation) et **5** (architecture générale du SCN) constituent un pont fondamental dans le développement du DSL. Ils opérationnalisent les concepts abordés dans le Chapitre 2 en montrant comment implémenter et exploiter la dynamique adaptative des pondérations $\omega_{i,j}$ dans des scénarios concrets. Cette transition permet également de préparer le terrain pour les chapitres ultérieurs, notamment les sections **2.5.5.3**, **2.5.4.3** et les chapitres **6**, **7**, **8**, qui exploreront l'intégration multi-échelle, l'optimisation et les applications multimodales du SCN. Pour plus de détails sur la transition entre théorie et pratique, les références internes aux sections **2.5.5.1** et **2.5.5.3** rappellent l'orientation initiale du Chapitre 2 et la synthèse des fondements

théoriques, tandis que les sections **2.4.5.2** et **2.5.4.3** présentent les perspectives d'intégration avec d'autres paradigmes.

2.5.5.3. Conclusion générale sur la place de ce chapitre 2 : on passe désormais d'un cadre historico-théorique à une mise en œuvre plus ingénierie pour initier le SCN

Ce chapitre 2 a permis de poser les fondements théoriques du **DSL** (Deep Synergy Learning) en établissant un panorama conceptuel riche et interdisciplinaire, qui relie des idées issues de la physique statistique, de la théorie des systèmes dynamiques et des modèles biologiques (tels que l'inhibition latérale et l'apprentissage Hebbien) aux approches contemporaines de l'**IA** et de l'auto-organisation. Les sections précédentes, notamment **2.5.1** qui aborde les précurseurs et l'évolution historique du DSL, **2.5.2** qui explore les potentialités de cohabitation entre entités symboliques et sub-symboliques, **2.5.3** qui présente des retours d'expériences dans le domaine de la robotique sensorimotrice, ainsi que **2.5.4** qui compare le DSL avec les paradigmes classiques (clustering, renforcement, GNN), ont permis de dresser un réseau de concepts centré sur la **synergie adaptative**. Nous avons vu que la dynamique des pondérations $\omega_{i,j}$ peut être interprétée en termes d'énergie potentielle, de recuit simulé et de mécanismes d'inhibition, offrant ainsi une vision qui, bien que partielle dans les cas non stationnaires, constitue un cadre de référence solide.

En effet, la formulation d'une fonction d'énergie du type

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{i,j}^2 + \dots$$

a permis d'illustrer comment, dans un scénario stationnaire, la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}|_{\omega_{ij}(t)}$$

peut être assimilée à une **descente de gradient** locale, guidée par la synergie $S(i,j)$ et régulée par le terme de décroissance $\tau \omega_{i,j}$. Ces concepts ont été détaillés dans les sections **2.4.3** et **2.4.4**, qui exposent respectivement les aspects énergétiques et les mécanismes d'inhibition, et ils illustrent comment le DSL se distingue des méthodes statiques classiques.

Ce chapitre se positionne ainsi comme un **socle historico-théorique** en établissant d'une part l'origine et l'évolution des idées qui sous-tendent le DSL (voir **2.5.1** et **2.5.2**) et, d'autre part, en comparant de manière critique le DSL avec d'autres paradigmes de l'**IA**, tels que les algorithmes de clustering traditionnels et les approches de renforcement présentées dans **2.5.4**. Cette approche comparative montre que, contrairement aux modèles fixés ou purement supervisés, le DSL propose une **auto-organisation dynamique** des connexions $\omega_{i,j}$, qui s'adapte en continu aux évolutions des données et aux interactions entre entités.

La place du chapitre 2 dans l'ensemble de l'ouvrage est ainsi double. Premièrement, il offre une **vue d'ensemble** qui situe le DSL dans le contexte des développements historiques et théoriques de l'**IA**, en faisant le lien avec les principes de la **physique statistique**, de l'**auto-organisation** et des mécanismes **biologiques** (tels que l'inhibition latérale et la plasticité synaptique). Deuxièmement, il prépare le terrain pour une transition vers une **mise en œuvre ingénierie** plus concrète, où les

concepts théoriques seront traduits en structures de données, algorithmes et architectures logicielles. Cette transition est explicitée dans les sections **2.5.5.1** et **2.5.5.2**, qui annoncent respectivement la représentation concrète des entités et la dynamique d'auto-organisation, et qui servent de prélude aux chapitres suivants.

En effet, le **Chapitre 3** traitera de la **représentation et de la modélisation** des entités \mathcal{E}_i dans divers contextes (image, texte, capteurs), en définissant précisément la fonction de synergie $S(i, j)$ adaptée à chaque type de données. Ensuite, les **Chapitres 4** et **5** se concentreront sur la mise en œuvre de la **dynamique d'auto-organisation** et la construction de l'**architecture globale** du SCN, en s'appuyant sur les principes établis ici – tels que la mise à jour des poids, les mécanismes d'inhibition, et les analogies avec des modèles physiques et biologiques. Enfin, les chapitres ultérieurs (notamment **6**, **7**, **8**, **9** et **10**) approfondiront les aspects d'**optimisation multi-échelle**, d'**apprentissage continu** et de **robustesse** du système.

Conclusion

En somme, ce **Chapitre 2** se veut fondateur tant sur le plan **historico-théorique** que sur le plan **comparatif**, en établissant un réseau de concepts – tels que la fonction d'énergie potentielle, le recuit simulé, l'inhibition latérale et la plasticité adaptative – qui structure la dynamique des pondérations $\omega_{i,j}$ dans un SCN. Il prépare ainsi le lecteur à une transition vers une phase **d'implémentation pratique**, où les idées théoriques seront traduites en algorithmes, en structures de données et en architectures logicielles concrètes. Cette démarche permet de passer du “pourquoi” au “comment” et d’initier l’exploitation effective du DSL dans des domaines variés, allant de la robotique sensorimotrice à l’IA multimodale, en passant par des applications en réseaux sociaux et en écologie numérique. Les références internes aux sections **2.5.1**, **2.5.2**, **2.5.3**, **2.5.4** et **2.5.5.1** illustrent la progression logique qui mène à cette transition, ouvrant la voie à une approche d’ingénierie concrète dans les chapitres suivants.

Chapitre 3 : Représentation et Modélisation des Entités d'Information

Chapitre 3 : Représentation et Modélisation des Entités d'Information	363
3.1. Introduction	Erreur ! Signet non défini.
3.2. Principes Généraux de la Représentation dans le DSL ..	Erreur ! Signet non défini.
3.3. Représentations Sub-Symboliques	Erreur ! Signet non défini.
3.4. Représentations Symboliques	Erreur ! Signet non défini.
3.5. Représentations Hybrides (Sub-Symbolique + Symbolique)	Erreur ! Signet non défini.
3.6. Aspects Avancés et Études de Cas.....	Erreur ! Signet non défini.
3.7. Conclusion.....	Erreur ! Signet non défini.
3.1. Introduction	365
3.1.1. Contexte et Motivation	365
3.1.2. Objectifs du Chapitre	368
3.1.3. Structure du Chapitre	375
3.2. Principes Généraux de la Représentation dans le DSL	380
3.2.1. Rôle Fondamental de la Représentation	380
3.2.2. Critères de Choix	383
3.2.3. Impact sur la Synergie et la Dynamique	389
3.3. Représentations Sub-Symboliques	396
3.3.1. Vecteurs et Embeddings	396
3.3.2. Autoencodeurs, Transformers et Approches Avancées	401
3.3.3. Calcul de la Synergie entre Vecteurs.....	407
3.3.4. Gestion du Bruit et de l'Évolution des Embeddings	415
3.4. Représentations Symboliques	422
3.4.1. Logiques, Règles et Ontologies	422
3.4.2. Calcul de la Synergie Symbolique	428
3.4.3. Évolution Symbolique et Gestes des Contradictions	437
3.5. Représentations Hybrides (Sub-Symbolique + Symbolique)	446
3.5.1. Motivation de la Fusion	446
3.5.3. Avantages et Défis	458
3.6. Aspects Avancés et Études de Cas.....	464
3.6.1. Hypergraphes et Synergie n-aire	464

3.6.2. Structures Fractales de Représentation	469
3.6.3. Études de Cas Illustratives	475
Exemple	481
3.6.4. Comparaison Expérimentale	482
3.7. Conclusion	490
3.7.1. Synthèse des Contributions du Chapitre	490
3.7.2. Limites et Pistes Futures	492
3.7.3. Liens avec les Chapitres Suivants	494
3.7.4. Vision Globale	495

3.1. Introduction

Dans ce chapitre, nous nous concentrons sur la **représentation** et la **modélisation** des entités d'information au sein du DSL (Deep Synergy Learning). Tout système d'apprentissage repose sur une certaine façon de **décrire** ses données, dans le DSL, ce sont les entités et leurs propriétés qui déterminent la manière dont la synergie $\omega_{i,j}$ sera calculée et mise à jour. La présentation qui va suivre servira de **pierre angulaire** aux chapitres ultérieurs (notamment Chapitre. 4 sur la dynamique d'auto-organisation, Chapitre. 5 sur l'architecture SCN, etc.), qui tireront directement profit de la qualité de la modélisation initiale.

3.1.1. Contexte et Motivation

La représentation des entités est un **facteur crucial** dans le DSL, si elle est inadéquate, la fonction de synergie $S(i,j)$ ne peut pas refléter correctement les relations entre entités, risquant de compromettre la formation de clusters pertinents ou l'adaptation du réseau. À l'inverse, une bonne modélisation facilite la détection de liens inattendus et l'émergence de structures synergiques riches. L'enjeu est d'autant plus grand que les entités peuvent être de différentes **natures** (images, textes, signaux, concepts symboliques), obligeant à manier des représentations **hétérogènes**.

3.1.1.1. Rappel du Livre : Le DSL Repose sur des Entités d'Information Dont la Qualité de Représentation Conditionne le Calcul de la Synergie et la Dynamique $\omega_{i,j}$

Il a été souligné dans les chapitres précédents que le **Deep Synergy Learning (DSL)** organise la connaissance autour d'un réseau de pondérations $\omega_{i,j}$ ajustées par une règle d'auto-organisation, comme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Cette règle ne prend tout son sens que si la **synergie** $S(i,j)$ entre les entités \mathcal{E}_i et \mathcal{E}_j fournit un **indicateur fiable** de leur affinité ou de leur compatibilité. La structure entière du **Réseau de connexion synergétique (Synergistic Connection Network (SCN))** en anglais) repose ainsi sur la capacité à estimer correctement les liens potentiellement utiles, ce qui dépend étroitement de la **qualité des représentations** de chaque entité.

Pour comprendre l'importance de ce point, il suffit de noter que la **dynamique** de l'auto-organisation, fondée sur la formule ci-dessus, renforce ou diminue les liaisons $\omega_{i,j}$ en fonction de $S(i,j)$. Dans le cas où la description des entités serait lacunaire ou bruitée, la **mesure** $S(i,j)$ risque de **s'égarer**, renforçant des liens non pertinents ou affaiblissant des liens au contraire prometteurs. La conséquence peut être un **réseau** qui ne capture pas la structure réelle des données, des **clusters** peu significatifs ou des phénomènes oscillatoires ne reflétant plus la nature véritable du système.

À l'inverse, si chaque entité \mathcal{E}_i est modélisée sous un format riche, cohérent et représentatif de ses caractéristiques internes, la **synergie** $S(i,j)$ devient un détecteur fiable des regroupements logiques ou des similitudes profondes. Il en résultera une **auto-organisation** claire, faisant émerger des liens forts là où la complémentarité est avérée, et réduisant ceux qui n'apportent que peu de cohérence.

Dans l'optique du **DSL**, cette qualité de représentation peut être sub-symbolique (vecteurs, embeddings, distributions de probabilité) ou symbolique (règles logiques, concepts, graphes), ou encore un **mélange** de ces approches dans un cadre **hybride**.

Les **méthodes** d'apprentissage profond (transformers, autoencodeurs, réseaux génératifs, etc.) fournissent souvent des vecteurs latents puissants et expressifs pour \mathcal{E}_i , tandis que les systèmes symboliques (ontologies, logiques descriptives, axiomes) véhiculent une information formelle ou sémantique plus explicite. Chaque choix a un **impact** direct sur le calcul

$$S(i, j) = f(\mathbf{r}(i), \mathbf{r}(j)),$$

où $\mathbf{r}(i)$ désigne la représentation de \mathcal{E}_i . L'**exactitude** et la **complétude** de $\mathbf{r}(i)$ s'avèrent alors cruciales pour guider le processus d'auto-organisation vers des **clusters** ou des **groupements** factuellement cohérents, révélant de manière fiable la structure cachée des données.

Du point de vue théorique, il apparaît donc que la **représentation** constitue un *pré-requis* indispensable à l'efficacité de l'ensemble du **DSL**. Même une très bonne règle de mise à jour $\omega_{i,j}$ échouera à produire une auto-organisation pertinente si $S(i, j)$ est trompeuse, ce qui arrive lorsque $\mathbf{r}(i)$ et $\mathbf{r}(j)$ ne contiennent pas d'informations réellement discriminantes ou significatives. À l'inverse, une représentation soigneusement conçue amplifiera l'action de la synergie, rendant la **mise à jour** plus rapide et plus stable vers un **réseau** présentant des **structures** solides, qu'il s'agisse de clusters distincts ou de sous-réseaux de coopération thématique.

3.1.1.2. Besoin d'Aborder la Diversité des Entités : Sub-Symboliques (Vecteurs, Embeddings), Symboliques (Règles, Concepts), Hybrides, Multimédias...

Dans le **Deep Synergy Learning**, un **enjeu majeur** réside dans la prise en compte de **données** dont la nature dépasse le simple cadre vectoriel homogène. Les entités manipulées peuvent relever du **sub-symbolique** (des vecteurs numériques issus de l'apprentissage profond), du **symbolique** (des règles logiques ou des concepts formels) ou d'une **hybridation** de ces deux approches. Plus encore, elles peuvent provenir de sources **multimédias** (images, sons, textes, vidéos) que le **DSL** souhaite traiter de manière intégrée. Cette **diversité** impose donc de **définir** des modèles de représentation appropriés, car la fonction de **synergie** $S(i, j)$ dépend directement de la manière dont $\mathbf{r}(i)$ et $\mathbf{r}(j)$ sont encodées.

Un premier registre rassemble les entités **sub-symboliques**, généralement décrites par des **vecteurs** ou des **embeddings**. Une telle approche s'avère particulièrement efficace pour modéliser des données continues ou massives, par exemple, un extrait textuel peut être projeté dans un espace latent à l'aide d'un modèle de langue, ou une image être convertie en un vecteur de caractéristiques via un **réseau neuronal convolutif**. En pratique, la **synergie** $S(i, j)$ se ramène souvent à une **similarité vectorielle** (cosinus, euclidienne), notée

$$S(i, j) = \text{sim}(\mathbf{r}_{\text{subsym}}(i), \mathbf{r}_{\text{subsym}}(j)),$$

où $\mathbf{r}_{\text{subsym}}(i)$ est un **embedding** associé à l'entité \mathcal{E}_i . De tels vecteurs favorisent la robustesse au bruit et la flexibilité face aux évolutions futures, mais peuvent se heurter à une **interprétabilité**

limitée, puisqu'il demeure malaisé de comprendre le contenu d'un embedding hautement dimensionnel.

À l'opposé, les entités **symboliques** sont décrites via des **blocs de règles**, des **concepts** structurés en **graphes sémantiques** ou en **logiques formelles**. Le calcul de la synergie $\mathbf{S}(i, j)$ se fonde alors sur des mesures de **compatibilité** entre axiomes, sur des **distances** dans une ontologie, ou sur des **critères** de cohérence dans un espace conceptuel. On obtient, par exemple,

$$\mathbf{S}(i, j) = \text{compat}(\mathcal{C}_i, \mathcal{C}_j),$$

où \mathcal{C}_i et \mathcal{C}_j désignent des entités logiques. Cette représentation **symbolique** jouit d'une **lisibilité** élevée, car chaque entité est énoncée sous forme de règles ou de propriétés aisément contrôlables. Néanmoins, elle peut se révéler **rigide** et peu adaptée au traitement de données massives ou bruitées, nécessitant une maintenance stricte de la **consistance** logique.

Dans de nombreuses applications, un **format hybride** apparaît plus naturel, chaque entité combine des composantes **sub-symboliques** et **symboliques**. Par exemple, un document textuel peut être associé à un **embedding** qui résume le contenu statistique global, tout en étant pourvu d'un **ensemble de mots-clés** ou de relations logiques qui en précisent la structure ou la sémantique. Dès lors, la fonction de synergie $\mathbf{S}(i, j)$ doit agréger à la fois une **mesure vectorielle** et un **score symbolique**. On peut imaginer une formule

$$\mathbf{S}(i, j) = \alpha \text{sim}(\mathbf{r}_{\text{subsym}}(i), \mathbf{r}_{\text{subsym}}(j)) + (1 - \alpha) \text{compat}(\mathcal{C}_i, \mathcal{C}_j),$$

la pondération $\alpha \in [0, 1]$ modulant l'importance relative du versant sub-symbolique et du versant symbolique. L'attrait de cette solution **hybride** réside dans sa souplesse. Elle allie la **richesse expressive** des approches symboliques à la **robustesse** et à la **continuité** des embeddings profonds.

Enfin, le DSL vise fréquemment le traitement de **données multimédias**, telles que des **images**, des **sons**, ou des **clips vidéo**. Ces entités peuvent elles-mêmes être considérées comme **sub-symboliques**, au sens où chaque support est converti en un vecteur (ou un tenseur) issu d'un réseau de neurones approprié (par exemple, un **spectrogramme** pour l'audio, une **carte de caractéristiques** pour l'image). Cependant, des **métadonnées symboliques** (catégories, tags, règles de provenance) peuvent en outre accompagner ces vecteurs. De la sorte, la **synergie** inclut un terme de similarité visuelle ou auditive, et, simultanément, un terme de compatibilité sémantique. On formule alors,

$$\mathbf{S}(i, j) = \text{sim}_{\text{media}}(\mathbf{m}_i, \mathbf{m}_j) + \text{sim}_{\text{sym}}(\mathbf{r}_{\text{sym}}(i), \mathbf{r}_{\text{sym}}(j)).$$

Cette fusion multimodale encourage l'**émergence** de clusters plus riches, dotés d'une cohérence simultanément perceptive et conceptuelle.

La **réalité** des usages implique donc que le **DSL** accueille une vaste **diversité** de formats, depuis la pure vectorisation jusqu'à l'**ontologie** la plus stricte, en passant par des **mélanges** sub-symbolique et symbolique, et par des **données** clairement multimédias. Cette hétérogénéité, loin d'être un obstacle, constitue un **levier** : la synergie s'y nourrit d'interactions inattendues, et des **clusterisations** inédites peuvent apparaître. La **condition** demeure néanmoins cruciale. Chaque entité doit être **décrise** de manière à dévoiler son essence ou sa fonction, faute de quoi la mesure $\mathbf{S}(i, j)$ ne sera qu'un miroir déformant qui freinera l'**auto-organisation** synergique.

3.1.2. Objectifs du Chapitre

Dans ce chapitre, nous allons détailler la **représentation** et la **modélisation** des entités d'information, un aspect central pour la réussite du **Deep Synergy Learning** (DSL). Il s'agit de comprendre :

126. **Les principes fondamentaux** de la représentation dans le DSL :
 - Pourquoi et comment décrire une entité \mathcal{E}_i ?
 - Quels critères de qualité influent sur la synergie $S(i, j)$?
- **Les différentes formes de modélisation** (vecteurs, graphes, ontologies, symboliques, hybrides) et leur **impact** sur la dynamique d'auto-organisation :
 - Des vecteurs ou embeddings pour les données sub-symboliques,
 - Des blocs de règles ou concepts pour les entités logiques,
 - Des structures mixtes pour combiner à la fois la puissance statistique et l'interprétabilité.
- **Les concepts avancés** (embeddings profonds, transformers, représentations symboliques complexes, synergie n-aire, etc.) permettant de **pousser plus loin** la capacité d'un DSL à découvrir des patterns riches et adaptatifs.

En d'autres termes, l'objectif est de **baliser** l'espace des représentations possibles, afin d'équiper le lecteur des connaissances nécessaires pour modeler correctement chaque type d'entité (\mathcal{E}_i), en tenant compte de la diversité (images, sons, textes, règles logiques) et de la flexibilité (adaptation en temps réel, extensions multiformes). Ce faisant, on jette les bases permettant au DSL de tirer pleinement parti de la notion de **synergie** pour organiser et valoriser ces données multiples.

3.1.2.1. Rôle Fondamental de la Représentation

Dans le **Deep Synergy Learning (DSL)**, la **représentation** de chaque entité \mathcal{E}_i constitue un **pivot** essentiel, car elle détermine la **qualité** de la **synergie** $S(i, j)$ et, partant, la **dynamique** de mise à jour des pondérations $\omega_{i,j}$. L'ensemble des mécanismes d'**auto-organisation**, qui aboutissent à la formation de **clusters** cohérents ou à l'émergence de **structures** plus complexes au sein du **Synergistic Connection Network (SCN)**, dépend donc étroitement de la façon dont chaque entité est encodée ou décrite.

Impact direct sur la synergie $S(i, j)$

La fonction $S(i, j)$ joue un rôle de **filtre** en évaluant dans quelle mesure deux entités \mathcal{E}_i et \mathcal{E}_j sont jugées **compatibles** ou **similaires**. La **définition** même de S varie selon la nature des entités, mais

fait systématiquement intervenir leur **représentation**. Si celle-ci est sub-symbolique (vecteur d'**embedding**, par exemple), on peut recourir à une mesure de **similarité vectorielle** comme la cosinus ou la gaussienne :

$$S(i,j) = \exp(-\alpha \parallel \mathbf{r}(i) - \mathbf{r}(j) \parallel) \quad \text{ou} \quad \frac{\mathbf{r}(i) \cdot \mathbf{r}(j)}{\parallel \mathbf{r}(i) \parallel \parallel \mathbf{r}(j) \parallel}.$$

Si la représentation est symbolique, S peut reposer sur un **calcul** de cohérence logique, de distance ontologique ou de correspondance conceptuelle :

$$S(i,j) = f_{\text{logic}}(\mathbf{C}(i), \mathbf{C}(j)).$$

Dans tous les cas, la **pertinence** de $S(i,j)$ dépend de la **capacité** des descripteurs $\mathbf{r}(i)$ ou $\mathbf{C}(i)$ à rendre compte des attributs substantiels de l'entité \mathcal{E}_i . En d'autres termes, une **incomplétude** ou un **bruit** excessif dans ces descripteurs entraîne une **approximation** inadéquate de la synergie, ce qui peut fausser la dynamique globale du réseau.

Conséquences sur l'auto-organisation

Le **principe** d'auto-organisation dans le DSL est de laisser chaque lien $\omega_{i,j}(t)$ évoluer selon une **règle** inspirée d'une descente d'énergie, telle que :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

La convergence ou la stabilité de cette équation dépend d'une part de l'**information** captée par $S(i,j)$. Une bonne représentation garantit que la **force** $S(i,j)$ saura distinguer avec finesse les entités véritablement proches (renforcement des liens) de celles sans réelle corrélation (faible synergie et diminution progressive de $\omega_{i,j}$). À l'inverse, si $\mathbf{r}(i)$ ne contient pas les caractéristiques déterminantes, le réseau peut s'orienter vers des **clusters** peu informatifs ou s'enliser dans des **oscillations** inutiles.

Cette relation entre **représentation** et **dynamique** d'auto-organisation se révèle cruciale dans la mise en œuvre pratique du DSL. En effet, une représentation solide permet de poser un **pas d'apprentissage** η plus élevé, accélérant la consolidation des liens. Au contraire, une représentation frêle ou sujette au **bruit** impose un réglage prudent de η afin d'éviter des ajustements erratiques de $\omega_{i,j}$.

Représentation comme "passeport commun"

Le **DSL** se veut généraliste et vise à traiter des entités issues d'une **variété** de domaines (images, langage, signaux biomédicaux, règles logiques, etc.). Pour pouvoir **comparer** ces entités au sein d'un même **SCN**, il faut établir un **passeport** ou un **langage commun**, c'est-à-dire une représentation dans laquelle $S(i,j)$ devient calculable même si \mathcal{E}_i et \mathcal{E}_j proviennent de modalités différentes. Ainsi, un document texte peut être encodé par un **embedding** (BERT, GPT, etc.) tandis qu'une image l'est par un CNN, et l'on projette ces descriptions dans un espace latent partagé permettant de définir, par exemple,

$$S(i, j) = \exp(-\|\Phi(\mathbf{r}_{\text{image}}(i)) - \Phi(\mathbf{r}_{\text{text}}(j))\|^2)$$

où Φ représente la fonction d'encodage multimodale. Cette approche assure la **cohérence** des liens de synergie, puisqu'on manipule des “vecteurs” homogènes malgré l'hétérogénéité initiale.

Robustesse et sensibilité au bruit

Une représentation de **qualité** se caractérise en outre par sa capacité à **absorber** le bruit des données et à **conserver** les informations essentielles pour le calcul de S. Un embedding textuel doit par exemple demeurer stable face à de légères variations lexicales, tandis qu'une représentation d'image doit tolérer des perturbations d'éclairage ou de perspective. Si l'on ne parvient pas à extraire ces invariances, l'estimation de la synergie est parasitée par des fluctuations indues, et la mise à jour de $\omega_{i,j}$ devient chaotique.

Il s'ensuit que la **stabilité** des descripteurs est un facteur-clé pour la réussite de l'auto-organisation. Dans un cadre “online”, où les représentations peuvent elles-mêmes être affinées au cours du temps, les fluctuations trop rapides de $\mathbf{r}(i)$ risquent de semer la confusion dans les pondérations $\omega_{i,j}$, engendrant des **oscillations** durables ou une absence de convergence stable.

Conclusion et perspectives

Le **rôle fondamental** de la représentation dans le DSL se manifeste à plusieurs niveaux : elle détermine la valeur de la synergie $S(i, j)$, oriente la formation des **clusters** via la règle de mise à jour de $\omega_{i,j}$ et gouverne la **stabilité** ou les **oscillations** dans le réseau. Quelle que soit la nature (sub-symbolique, symbolique ou hybride) de la description, son adéquation aux caractéristiques du problème se révèle indispensable pour que l'auto-organisation aboutisse à des regroupements cohérents et exploitables.

Dans les sections suivantes (3.2 à 3.5), on examinera plus en détail la manière dont on peut modéliser une entité \mathcal{E}_i en recourant à des approches **sub-symboliques** (vecteurs, embeddings neuronaux), **symboliques** (règles, ontologies, structures logiques) ou **hybrides** (combinaison de descripteurs neuronaux et de blocs de connaissances expertes). On illustrera également comment cette variété de représentations se traduit dans la fonction S, influençant l'ensemble de la dynamique synergique. Ainsi, le **DSL** se dote d'un cadre flexible où le **choix** de la représentation devient un levier majeur pour révéler des **synergies** pertinentes dans des univers de données de plus en plus hétérogènes.

3.1.2.2. Détails des Formes de Modélisation (Vecteurs, Graphes, Ontologies...) et Leur Impact sur la Synergie

Le **Deep Synergy Learning (DSL)** manipule des entités \mathcal{E}_i dont la **représentation** peut relever de plusieurs paradigmes. On peut décrire chaque entité par un **vecteur** (ou embedding), par un **graphe** interne, par un **ensemble de règles symboliques** (ou tout autre formalisme logique), ou encore adopter une **approche hybride** combinant ces perspectives. Dans chacun de ces cas, la **synergie**

$S(i,j)$ — définie entre deux entités $\mathcal{E}_i, \mathcal{E}_j$ — se construit à partir d'un opérateur de **similarité** (ou de **compatibilité**) approprié. Le choix de la **forme** de modélisation influe ainsi directement sur la **nature** de $S(i,j)$, sur la **complexité** du calcul et sur la **qualité** de l'auto-organisation qui en découle.

A. Entités Vectorielles

Dans de nombreux cas, les entités se décrivent à l'aide d'**embeddings** ou de **vecteurs** $\mathbf{x}_i \in \mathbb{R}^d$. Cette approche est couramment employée pour encoder des images, des segments textuels, des extraits audio ou d'autres données sub-symboliques via un réseau neuronal ou une procédure d'extraction de caractéristiques. La fonction S peut alors prendre la forme d'une **distance** (euclidienne, Minkowski) ou d'une **similarité** (produit scalaire, cosinus) :

$$S_{\text{vectoriel}}(i,j) = \phi(\mathbf{x}_i, \mathbf{x}_j),$$

où ϕ est un opérateur de similarité comme

$$\phi(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

L'avantage principal réside dans la **simplicité** du calcul : la mise à jour des pondérations $\omega_{i,j}$ peut alors s'effectuer rapidement, favorisant une bonne **scalabilité** lorsque le **Synergistic Connection Network (SCN)** s'étend à un grand nombre d'entités. De plus, les représentations vectorielles, si elles sont correctement entraînées ou extraites, capturent des **propriétés** sémantiques ou perceptives pertinentes (ex. embeddings textuels pour la sémantique linguistique, features CNN pour l'analyse d'images).

En contrepartie, ces modèles vectoriels peuvent souffrir d'une **opacité** : ils sont moins interprétables, et leur dimension peut croître, ce qui alourdit le traitement si le SCN doit parcourir un espace vectoriel de grande taille. La **mise en œuvre** de l'auto-organisation demeure néanmoins directe, puisqu'on effectue souvent un simple calcul de **similarité** pour chaque paire (i,j) .

B. Entités Graphiques

Une autre forme de description se fonde sur des **graphes** internes. Chaque entité \mathcal{E}_i est un graphe G_i , par exemple un graphe de relations (conceptuels, biologiques, topologiques) ou un graphe de dépendances. La **synergie** $S(i,j)$ s'exprime alors par un **score** de correspondance structurale, comme

$$S_{\text{graphique}}(i,j) = \Psi(G_i, G_j).$$

On rencontre notamment la **graph edit distance** (GED), mesurant le coût minimal de transformations (insertion/suppression de nœuds ou d'arêtes) pour passer de G_i à G_j . Une fonction de similarité exponentielle, de la forme

$$\exp(-\alpha \text{GED}(G_i, G_j)),$$

peut alors servir de **synergie**.

Cette approche est plus **riche** que les simples vecteurs, puisqu'elle autorise une modélisation explicite de la structure interne, mais elle est plus **coûteuse** : la comparaison de graphes s'avère un problème algorithmique complexe (souvent NP-difficile pour l'isomorphisme de graphes généraux), et l'on doit donc gérer un surcroît de calcul. Sur le plan de la **dynamique** du SCN, l'évaluation de $S_{\text{graphique}}(i, j)$ peut s'avérer un **goulot d'étranglement** si le réseau compte de nombreuses entités structurées.

C. Entités Logiques ou Symboliques

Dans un univers **symbolique**, chaque entité \mathcal{E}_i peut être un ensemble de **règles**, un **théorème**, un fragment d'**ontologie** ou un **module logique** exprimant des propriétés formelles. Le calcul de $S(i, j)$ repose alors sur une **compatibilité** conceptuelle, par exemple un degré de “contradiction nulle” ou de “similarité sémantique” entre deux groupes d'axiomes :

$$S_{\text{symbolique}}(i, j) = \text{compat}\left(\{\text{règles}_i\}, \{\text{règles}_j\}\right).$$

On peut envisager une fonction de synergie mesurant la **proximité** de deux ontologies (recouvrement d'axiomes) ou la **cohérence** de l'union $\mathcal{E}_i \cup \mathcal{E}_j$. L'attrait principal réside dans l'**interprétabilité** : on sait **expliquer** pourquoi deux entités s'avèrent compatibles ou non, et l'on peut procéder à des **déductions** explicites. Toutefois, le formalisme symbolique exige une mise à jour **plus rigide**, et demeure peu tolérant au **bruit** ou à l'incertitude.

D. Modélisation Hybride et Multimodale

Dans de nombreux contextes, les entités \mathcal{E}_i se décrivent à la fois par des **caractéristiques sub-symboliques** (e.g. embeddings de textes, d'images) et par des **attributs symboliques** (e.g. champs sémantiques, règles associées, métadonnées). Il s'agit alors d'une **modélisation hybride**, où la synergie combine plusieurs fonctions de similarité :

$$S_{\text{hybride}}(i, j) = \lambda \phi(\mathbf{x}_i, \mathbf{x}_j) + (1 - \lambda) f_{\text{logic}}(\{\text{axiomes}_i\}, \{\text{axiomes}_j\}),$$

avec $\lambda \in [0, 1]$ un hyperparamètre de balance. Cette approche **hybride** rend possible l'exploitation simultanée d'**informations** quantitatives et qualitatives, donnant une vision plus complète de la relation entre entités. Dans un cadre multimodal (images, textes, sons, etc.), on peut adapter la fonction $\phi(\cdot, \cdot)$ à chaque modalité et agréger les scores partiels par pondération ou par un opérateur plus sophistiqué.

La **richesse** d'un tel modèle s'accompagne d'une **complexité** accrue : l'évaluation de la synergie nécessite de jongler entre plusieurs types de descripteurs, et le réglage des poids λ peut s'avérer délicat. D'un point de vue opérationnel, le SCN doit gérer des structures de données plus variées, et la formation de **clusters** engage des entités aux caractéristiques hétérogènes.

Conséquences sur la Dynamique d'Auto-Organisation

Le **choix** d'une forme de modélisation détermine **directement** le **coût** et la **précision** de la synergie $S(i, j)$. Un calcul vectoriel est **rapide**, mais peut manquer de **transparence** ; un calcul symbolique ou graphique peut être sémantiquement **puissant** mais se révèle onéreux et plus fragile au bruit. Dans un système d'**auto-organisation**, où chaque lien $\omega_{i,j}$ est actualisé de manière itérative, la **complexité** de $S(i, j)$ devient cruciale si le réseau comporte un grand nombre d'entités. Par ailleurs, la **stabilité** de la représentation, l'adéquation entre la mesure de similarité et la réalité sémantique, ainsi que la résistance au bruit déterminent la **qualité** de l'émergence de clusters ou de schémas cognitifs.

Dans la pratique, un **DSL** performant peut :

127. **S'appuyer** sur des vecteurs ou embeddings lorsque les données sont massives et qu'on souhaite une **scalabilité** haute, quitte à sacrifier l'interprétabilité.
128. **Exploiter** des structures logiques pour bénéficier d'une **lisibilité** conceptuelle, en acceptant un calcul de S plus lourd.
129. **Combiner** des descripteurs (e.g. sub-symboliques + attributs symboliques) pour dégager des synergies multidimensionnelles, au prix d'un pipeline de calcul plus complexe.

Au sein du **Synergistic Connection Network**, cette souplesse dans la définition de $S(i, j)$ autorise l'**intégration** de données de sources multiples et la **découverte** de correspondances inattendues — moyennant une **ingénierie** soignée pour élaborer la modélisation la plus appropriée aux besoins applicatifs. Les chapitres ultérieurs (notamment ceux traitant de la **fusion multimodale** ou des **architectures SCN** avancées) approfondiront comment cette diversité se reflète dans les mécanismes concrets d'auto-organisation synergique.

3.1.2.3. Aborder des Concepts Avancés : Embeddings à Haut Niveau (Transformers), Représentations Symboliques Complexes, etc.

La **représentation** des entités dans un **DSL** (Deep Synergy Learning) n'est pas limitée à de simples vecteurs statiques ou à des graphes élémentaires : pour décrire des objets ou des notions de plus en plus complexes, on peut recourir à des **embeddings contextuels** (issus de **Transformers**), à des **ontologies** riches ou à des **structures** multi-niveau voire fractales. Ces choix de modélisation, s'ils rendent la **synergie** plus riche et plus précise, posent également des défis en termes de **coût** algorithmique et de **gestion** de la dynamique. Les paragraphes qui suivent illustrent comment cette sophistication accroît le **pouvoir** du DSL au prix d'une **ingénierie** plus exigeante.

A. Embeddings à Haut Niveau (Transformers et Architectures Contextuelles)

Dans le cadre sub-symbolique, les entités \mathcal{E}_i sont souvent associées à un **embedding** $\mathbf{x}_i \in \mathbb{R}^d$. Auparavant, des modèles statiques comme Word2Vec ou GloVe généraient un vecteur unique pour chaque mot. Aujourd'hui, l'essor des **Transformers** (BERT, GPT, T5, Vision Transformers, etc.) permet d'obtenir des **embeddings contextuels**, où la représentation varie selon l'environnement

local (contexte lexical, patchs voisins dans une image, etc.). Mathématiquement, pour une entité \mathcal{E}_i , on définit une fonction

$$\mathbf{v}_i: \mathbf{C}_i \mapsto \mathbb{R}^d,$$

où \mathbf{C}_i est le **contexte** (par exemple, les tokens entourant un mot, la position dans une séquence d'image, etc.). La **synergie** $S(i, j)$ peut alors prendre la forme d'une similarité vectorielle appliquée aux **embeddings** contextuels

$$S_{\text{transf}}(i, j) = \text{sim}(\mathbf{v}_i(\mathbf{C}_i), \mathbf{v}_j(\mathbf{C}_j)).$$

Cette contextualisation renforce la **précision** de la mesure de similarité : un même mot ou un même patch peut être représenté différemment suivant les indices locaux, évitant ainsi les ambiguïtés que l'on retrouve dans les embeddings purement statiques. En contrepartie, un tel modèle implique un **coût** de calcul plus important (multiple têtes d'attention, layers de grande taille) et une **dynamique** plus complexe si le **contexte** \mathbf{C}_i se modifie continuellement durant l'apprentissage. Le **DSL** doit donc gérer des synergies potentiellement instables si les embeddings contextuels ne sont pas figés.

B. Représentations Symboliques Complexes (Ontologies Riches, Logiques Hiérarchiques)

Le raffinement ne se limite pas au sub-symbolique. Dans un contexte **symbolique**, les entités peuvent relever de **logiques descriptives**, d'**ontologies** complexes (OWL, RDF) ou de **théories** plus abouties. L'idée est alors de définir

$$S_{\text{symbolique}}(i, j) = \text{compat}(\mathcal{O}_i, \mathcal{O}_j),$$

où \mathcal{O}_i et \mathcal{O}_j désignent des sous-ensembles de l'ontologie ou des blocs logiques associées aux entités \mathcal{E}_i et \mathcal{E}_j . Le calcul de **compat** peut se baser sur le **recouvrement** de concepts, la **cohérence** (absence de contradiction), ou encore l'**intersection** sémantique. Cette approche a le **mérite** de l'**explicabilité** : on sait pourquoi deux entités sont jugées compatibles (elles partagent des axiomes, des types, des relations). D'un point de vue opératoire, cependant, l'exécution peut nécessiter des **moteurs d'inférence** plus lourds, surtout si on emploie des logiques complexes ou des algorithmes de correspondance ontologique (ontology matching). Le DSL profite ici de la grande **lisibilité** des clusters émergents, mais la mise à jour $\omega_{i,j}$ s'expose à la rigidité du formalisme symbolique, moins tolérant au **bruit** et aux incertitudes.

C. Structures Multi-Niveau ou Fractales

Dans des applications **multi-échelle** (cf. chap. 6), chaque entité \mathcal{E}_i peut se décliner en **sous-niveaux** ou exhiber des propriétés fractales. Par exemple, on peut envisager un document organisé en chapitres, sections, paragraphes, phrases, ou un graphe subdivisé hiérarchiquement. Le calcul de la synergie $S(i, j)$ s'étend alors sur plusieurs **granularités**. Une formalisation schématique pourrait recourir à un opérateur d'intégration multi-résolution :

$$S_{\text{multi}}(i, j) = \int_0^L \text{Sim}(\mathcal{E}_i^{(\ell)}, \mathcal{E}_j^{(\ell)}) d\ell,$$

où ℓ indexe le niveau d'observation, entre 0 (global) et L (local très détaillé). Cette construction donne une **vision holistique** de la synergie, dans laquelle deux entités se révèlent proches si leurs structures s'apparentent à **tous** les niveaux ou si l'on détecte un fort **matching** dans plusieurs couches. Le **DSL** exploite alors une **auto-organisation** multiforme : si la synergie est forte à certains niveaux, les liens se renforcent, même si la vue globale n'est pas identique. Le prix à payer réside dans le **coût** de comparaison de multiples niveaux et dans la gestion de la cohérence à travers ces échelles.

D. Combiner Diverses Approches

Souvent, un système DSL se trouve confronté à des données hétérogènes mêlant représentations avancées sub-symboliques, structures symboliques complexes et granularités multiples. Une **approche hybride** consiste alors à définir

$$S_{\text{avancée}}(i, j) = \lambda_1 \text{sim}_{\text{contextuel}}(\mathbf{v}_i, \mathbf{v}_j) + \lambda_2 \text{compat}_{\text{symbol}}(\mathcal{O}_i, \mathcal{O}_j) + \lambda_3 \text{SimMulti}(\mathcal{E}_i, \mathcal{E}_j),$$

avec $\lambda_1 + \lambda_2 + \lambda_3 = 1$. Cette formule intègre la **similarité contextuelle** (Transformers), la **compatibilité symbolique** (ontologies) et la **cohérence multi-niveau** (hiérarchie interne). Un tel design favorise une **syncrétisation** de l'information, permettant au **SCN** de découvrir des **clusters** sophistiqués englobant plusieurs perspectives. Cependant, la charge algorithmique et la complexité de déploiement augmentent d'autant. De plus, l'**auto-organisation** doit gérer la potentielle **instabilité** due à la variation simultanée de différentes composantes, particulièrement si l'on adapte en continu les embeddings et l'ontologie.

Conclusion

L'introduction de **concepts avancés** en représentation — embeddings contextuels, ontologies riches, structures multi-échelle — vient **enrichir la synergie** et, par conséquent, la **dynamique** du DSL. D'une part, le système gagne en **capacité** pour discerner des analogies profondes ou des compatibilités non triviales ; d'autre part, le **coût** de calcul et le **pilotage** de la mise à jour des liens $\omega_{i,j}$ se compliquent, imposant une **gestion** soigneuse des paramètres et une ingénierie algorithmique plus ambitieuse. Dans l'ensemble, ces modèles avancés représentent la voie naturelle pour étendre le DSL à des **champs** de données complexes (vision sémantique, NLP contextuel, robotique cognitive, etc.), où la **simple** similarité vectorielle ou la **simple** compatibilité symbolique ne suffisent plus à saisir toute la **richesse** du domaine.

3.1.3. Structure du Chapitre

Dans ce chapitre, nous allons progressivement bâtir une **vision complète** de la représentation et de la modélisation des entités d'information dans le DSL. Nous avons déjà rappelé (sections 3.1.1 et 3.1.2) les **enjeux** et les **objectifs** ; désormais, nous allons suivre une **progression** logique afin de couvrir toutes les approches possibles (sub-symboliques, symboliques, hybrides, etc.) et les approfondir par des illustrations concrètes.

3.1.3.1. Vue d'Ensemble : Principes, Représentations Sub-Symboliques, Symboliques, Hybrides, Aspects Avancés et Études de Cas

Ce chapitre se propose d'explorer, de manière structurée, la diversité des **représentations** qui peuvent être adoptées pour décrire les entités \mathcal{E}_i au sein d'un **Deep Synergy Learning (DSL)**, et d'en illustrer l'influence directe sur le **calcul de synergie** et la **dynamique adaptative**. Les principes généraux de la représentation, tels qu'ils s'appliquent au DSL, constituent un socle méthodologique qui sera mis à profit dans les chapitres ultérieurs portant sur l'**auto-organisation** (Chap. 4) et l'**architecture** du Synergistic Connection Network (Chap. 5).

L'idée centrale, rappelée dans ce chapitre, est que la **qualité** et la **nature** de la représentation conditionnent la **pertinence** de la fonction de synergie $S(i, j)$. Que les entités soient modélisées sous forme de **vecteurs** (apprentissage profond, embeddings neuronaux), de **structures symboliques** (règles logiques, graphes ontologiques) ou d'**architectures hybrides**, la façon dont on code l'information se répercute sur la manière dont le DSL identifie les relations entre entités, renforce ou affaiblit les pondérations $\omega_{i,j}$ et, en définitive, fait émerger des **clusters**. Les sections 3.2 à 3.7 proposent une trajectoire progressive : on part des **principes** généraux (3.2), on examine ensuite les cas **sub-symboliques** (3.3) et **symboliques** (3.4), puis on étudie les **représentations hybrides** (3.5). Enfin, on introduit certains **aspects avancés** (3.6) et l'on conclut (3.7) sur la synthèse de ces différentes approches.

Section 3.2 : Principes généraux de la représentation dans le DSL

On y clarifie le rôle clé que joue la représentation des entités dans l'élaboration de la **synergie** $S(i, j)$. Après un rappel sur les exigences de robustesse, d'expressivité et de modularité, on souligne en quoi la **cohérence** entre la forme de représentation et la dynamique d'**auto-organisation** est décisive pour éviter des clusterisations parasitaires ou des oscillations improductives. Les arguments développés forment un **cadre théorique** dans lequel s'inscriront toutes les variantes ou extensions présentées par la suite.

Section 3.3 : Représentations sub-symboliques

Cette partie met l'accent sur les vecteurs et embeddings neuronaux, abordant notamment les mécanismes usuels de **similarité vectorielle** (distance euclidienne, cosinus, noyaux gaussiens). On met en évidence la facilité avec laquelle s'opère le **calcul** de synergie dans un espace vectoriel, contribuant ainsi à la **scalabilité** du DSL quand le nombre d'entités est important. Les vecteurs issus de modèles de deep learning (CNN, Transformers, autoencodeurs) illustrent l'**efficacité** et la **souplesse** des représentations sub-symboliques, tout en rappelant les limites liées à une **opacité** conceptuelle ou à une sensibilité aux biais de l'entraînement supervisé.

Section 3.4 : Représentations symboliques

Dans un second temps, on passe en revue les **entités** décrites au moyen de **structures logiques** ou **d'ontologies**. Cette approche présente l'intérêt d'une **interprétabilité** et d'une **inférence** plus directe : deux entités partagent-elles des axiomes ? Leurs règles s'avèrent-elles compatibles, voire contradictoires ? Le **calcul** de $S(i, j)$ peut alors refléter la proportion d'axiomes communs, la distance entre classes ontologiques, ou la cohérence sémantique de leur union. Ce formalisme logique, souvent plus **rigide**, nécessite cependant des algorithmes de vérification plus lourds, et se

montre peu tolérant au **bruit**. Les implications sur la dynamique du SCN sont discutées, soulignant la nécessité de contrôler la croissance de la complexité dans un réseau de grande taille.

Section 3.5 : Représentations hybrides

Une synthèse s'impose lorsque l'on veut combiner les atouts des vecteurs sub-symboliques (robustesse, généralisation) avec la lisibilité des structures symboliques. Les **modèles hybrides** associent ainsi, pour chaque entité \mathcal{E}_i , un embedding \mathbf{v}_i et des règles \mathcal{R}_i . La synergie se conçoit alors comme une combinaison linéaire ou non linéaire de deux (ou plusieurs) mesures : l'une extrayant la **similarité** vectorielle, l'autre évaluant la **compatibilité** symbolique. Le paramètre λ dans

$$S_{\text{hybride}}(i, j) = \lambda \rho(\mathbf{v}_i, \mathbf{v}_j) + (1 - \lambda) \text{Compat}(\mathcal{R}_i, \mathcal{R}_j)$$

illustre la souplesse de ce modèle, capable de s'adapter à des applications où la **sémantique** coexiste avec des propriétés statistiques ou perceptives. Les avantages d'une telle approche sont contrebalancés par la gestion plus complexe du calcul de **synergie**, qui doit agréger plusieurs points de vue.

Section 3.6 : Aspects avancés et études de cas

Cette section présente des **concepts** ou **développements** supplémentaires visant à accroître encore la diversité ou la sophistication des représentations. On évoque notamment les **synergies n-aires**, permettant de modéliser les interactions collectives entre plus de deux entités, ainsi que l'extension vers des **structures fractales** et **hypergraphes**. De plus, on y propose des **études de cas** concrets : systèmes de vision sémantique enrichis par des ontologies, applications multimodales (audio, vidéo, textes), ou robotique cognitive où chaque entité correspond à un composant sensoriel ou à un module de planification. Ces exemples soulignent la flexibilité du DSL, mais mettent en relief la nécessité d'une **ingénierie** minutieuse pour garantir la **cohérence** et la **performance** du calcul de synergie.

Section 3.7 : Conclusion

Un dernier moment est consacré à **résumer** les points saillants de ce chapitre, tout en traçant des **passerelles** avec les chapitres suivants. On rappelle qu'une **représentation** de haute qualité (qu'elle soit vectorielle, symbolique ou hybride) conditionne la **dynamique** d'auto-organisation du DSL, à la fois dans la **descente d'énergie** qui stabilise les pondérations $\omega_{i,j}$ et dans la **formation** de clusters pertinents. Les choix retenus ici déterminent en grande partie l'efficacité, la robustesse et l'interprétabilité du **Synergistic Connection Network** dans les scénarios réels, un enjeu qu'illustreront les déploiements concrets abordés ultérieurement.

3.1.3.2. Renvoi aux Chapitres Ultérieurs

Le présent chapitre, consacré aux principes et aux méthodes de **représentation** des entités dans le **Deep Synergy Learning (DSL)**, s'inscrit dans un continuum dont les chapitres suivants approfondiront la dimension dynamique, architecturale et opérationnelle. La manière dont on encode une entité \mathcal{E}_i et dont on calcule la synergie $S(i, j)$ aura un impact direct sur les mécanismes décrits dans ces parties ultérieures. Le fait de bien saisir les choix de représentation vecteurs, règles

symboliques ou structures hybrides, et de comprendre leurs avantages et inconvénients, constitue un prérequis essentiel à l'étude des dynamiques d'**auto-organisation**, de l'**architecture** du Synergistic Connection Network et de son déploiement multi-échelle. Les liens les plus directs avec les chapitres suivants s'établissent ainsi :

Dans le **Chapitre 4**, qui détaille la **dynamique d'auto-organisation**, les pondérations $\omega_{i,j}$ évoluent selon une équation itérative qui repose fortement sur la synergie $S(i,j)$. Si cette synergie est déterminée par des embeddings sub-symboliques, le calcul de similarité sera rapide, favorisant l'implémentation de mécanismes tels que la descente d'énergie ou la stabilisation de clusters. Inversement, si la représentation est symbolique, la règle de mise à jour devra tenir compte de la compatibilité logique, ce qui influence la façon dont les clusters se forment ou se fragmentent. Les notions de robustesse de la représentation, de granularité ou de sensibilité au bruit abordées dans ce chapitre alimenteront la réflexion sur les oscillations, les convergences locales, ou l'émergence de configurations stables que le Chapitre 4 explicitera.

Le **Chapitre 5**, quant à lui, traite de l'**architecture générale** du Synergistic Connection Network. Là encore, les choix de représentation sont déterminants puisque la conception même du réseau – qu'il s'agisse de la gestion de larges embeddings vectoriels, de graphes conceptuels ou de règles logiques – impose des contraintes sur la structure interne du SCN, les protocoles de communication entre nœuds et la répartition des calculs. Ainsi, le fonctionnement d'un SCN manipulant des structures symboliques complexes doit intégrer un module de compatibilité ou d'inférence, tandis qu'un SCN vectoriel peut s'appuyer sur des algorithmes de similarité plus classiques. Ces aspects d'implémentation et de mise à l'échelle sont traités au Chapitre 5, mais prennent appui sur les fondements de la représentation établis ici.

Dans le **Chapitre 6**, consacré à l'**apprentissage synergique multi-échelle**, les représentations décrites dans le présent chapitre se verront transposées à des scénarios où la notion d'échelle ou de hiérarchie devient fondamentale. La façon dont une entité est encodée doit permettre de naviguer entre un niveau micro (par exemple, des attributs locaux d'un signal, des sous-parties d'un graphe) et un niveau macro (des clusters de haut niveau, des super-concepts). Les principes de modularité et de robustesse abordés ici guideront la construction d'architectures capables de traiter simultanément ces différentes granularités, et influenceront le mode de calcul de la synergie lorsqu'il s'agit de regrouper des "super-entités" ou d'organiser plusieurs couches de représentation.

Le **Chapitre 7**, traitant des **algorithmes d'optimisation** et des **méthodes d'adaptation dynamique**, s'appuiera également sur les fondements énoncés dans le présent chapitre. Les stratégies de recuit simulé, de heuristiques génétiques ou d'optimisation distribuée dans un DSL se conçoivent plus facilement lorsque la fonction de synergie est aisément manipulable et la représentation stable. Les embeddings sub-symboliques se prêtent par exemple à des manœuvres de projection ou de sparsification utiles pour accélérer les routines d'optimisation, alors que des formalismes symboliques imposent des approches plus subtiles pour maintenir la cohérence logique ou résoudre les contradictions.

Le **Chapitre 8**, qui se focalise sur un **DSL multimodal** intégrant par exemple la vision, le langage et les signaux auditifs, prolonge la question de la représentation dans un cadre où chaque modalité est codée différemment : CNN pour les images, embeddings BERT pour les textes, paramètres acoustiques pour l'audio. Les notions de ce chapitre portant sur la fusion sub-symbolique et symbolique, ou sur la construction de synergies hybrides, trouvent leur application concrète dans

la conception d'un SCN capable d'identifier des associations ou des regroupements cohérents à travers plusieurs modalités. La qualité de la représentation qu'on aura choisie pour chaque type de contenu détermine alors la richesse ou la précision de la synergie inter-modale.

Enfin, dans d'autres chapitres ultérieurs (Chapitres 9, 10, 11, etc.) traitant de l'**évolution en temps réel**, des mécanismes de **feedback coopératif** ou encore de la **robustesse** et de la **sécurité**, il apparaîtra qu'une bonne représentation n'est pas un simple détail : c'est un moteur de résilience et de flexibilité. Les entités décrites de manière adéquate peuvent se réorganiser ou se mettre à jour plus aisément, et la logique de l'auto-organisation est plus lisible lorsque la synergie repose sur des signatures explicites ou bien définies. De surcroît, l'ajout d'un feedback ascendant ou descendant exige parfois la capacité de retourner à des représentations interprétables, qu'on ne peut dissocier des choix opérés dans le présent chapitre. Les mécanismes de détection d'anomalies ou de perturbations exploitent eux aussi la nature de l'entité : des vecteurs robustes au bruit ou des graphes logiques doués de mécanismes de validation interne peuvent amortir les secousses et limiter les propagations d'erreurs.

En somme, le **Chapitre 3** jette les bases d'un ensemble de choix et de contraintes qui façonnent la dynamique, l'architecture et la pérennité du Deep Synergy Learning. Les représentations sub-symboliques, symboliques ou hybrides et leurs déclinaisons avancées constitueront un fil rouge qui s'étire jusqu'aux démonstrations et applications pratiques des derniers chapitres, liant la théorie de la synergie et l'implémentation à grande échelle d'un SCN apte à manipuler des données hétérogènes, évolutives et à multiples échelles de granularité.

3.2. Principes Généraux de la Représentation dans le DSL

L'étape de **représentation** d'une entité \mathcal{E}_i est un maillon essentiel de la chaîne d'apprentissage dans le **DSL** (Deep Synergy Learning). De cette modélisation initiale dépend la qualité de la fonction de synergie $S(i, j)$ — et, par conséquent, la pertinence des clusters et de la dynamique d'auto-organisation $\omega_{i,j}$. Avant d'explorer les différents types de représentation (sections 3.3, 3.4, 3.5) et leurs implications (section 3.6), nous allons poser quelques **principes généraux** essentiels à comprendre.

3.2.1. Rôle Fondamental de la Représentation

À ce stade, nous avons déjà souligné (chap. 3.1) que la **description** de chaque entité influe directement sur la manière dont on évalue la synergie. Entrons maintenant dans le détail de ce rôle fondamental, en explicitant comment la représentation oriente tout le processus d'apprentissage.

3.2.1.1. Chaque entité \mathcal{E}_i doit être “capturable” par une structure (vecteur, ensemble de règles...)

Afin de garantir la cohérence globale du **Deep Synergy Learning (DSL)** et de son mécanisme d'**auto-organisation**, il est indispensable que chaque entité \mathcal{E}_i puisse être décrite au moyen d'une **structure** qui reflète ses attributs pertinents et permette de calculer la **synergie** $S(i, j)$. L'enjeu est de choisir ou de concevoir une représentation à la fois robuste et suffisamment expressive pour mettre en évidence les similitudes ou les compatibilités recherchées au sein du **Synergistic Connection Network (SCN)**. Plusieurs facteurs interviennent alors dans la définition de ce “format” :

Dans un premier temps, il s'agit de **saisir** les attributs importants de l'entité \mathcal{E}_i . Lorsque l'objet à décrire est une image, se limiter à une statistique globale comme la moyenne de luminosité ne suffit pas à capturer des singularités plus complexes (formes, couleurs, textures). Au contraire, un **embedding** généré par un réseau de neurones profonds (CNN, Vision Transformer, etc.) peut dégager des caractéristiques discriminantes indiquant la présence de motifs ou d'objets précis. Un raisonnement analogue s'applique aux textes : un simple bag-of-words se révèle souvent insuffisant pour rendre compte des relations contextuelles et des sémantiques sous-jacentes, tandis qu'un **embedding contextualisé** (type BERT ou GPT) prend en compte l'environnement lexical et la structure syntaxique. Dans le cas de données symboliques (règles, axiomes), il faut veiller à recourir à un formalisme (logique, ontologie, graphe conceptuel) permettant de décrire toutes les relations nécessaires, sans perte d'expressivité.

Il est également primordial de **faire correspondre** la forme de la représentation au **mode de calcul** de la fonction $S(i, j)$. Si la synergie exploite une mesure de similarité vectorielle, par exemple la distance cosinus ou un noyau RBF, alors chaque entité se voit associée à un vecteur \mathbf{x}_i dans un espace de dimension raisonnable. À l'inverse, si l'on s'appuie sur un critère de **compatibilité logique** (comme la proportion d'axiomes communs ou la cohérence de deux ensembles de règles), il devient impératif d'exprimer \mathcal{E}_i sous une forme symbolique adéquate (règles logiques, graphes de connaissances, etc.), afin que les algorithmes de correspondance ou d'inférence puissent opérer.

Ainsi, il doit y avoir une **cohérence** entre le type de structure choisi et l'**opérateur** de similarité ou de compatibilité mis en œuvre dans S.

Cette logique doit être prolongée dans les contextes **multimodaux** ou hétérogènes, où les entités se répartissent entre images, textes, flux audio, signaux bruts, formules logiques ou tout autre format. Il est alors vain de tenter d'imposer un unique schéma identique pour toutes les entités. Une image pourra s'encoder sous forme d'un embedding convolutif \mathbf{x}_i , tandis qu'une base de règles symboliques restera décrite par un ensemble \mathcal{R}_i . Toutefois, le **DSL** exige qu'il existe un moyen, le cas échéant, de calculer $S(i, j)$ entre entités de types différents. Cette nécessité peut conduire à prévoir un module de “passerelle” ou de traduction conceptuelle — par exemple, via des métadonnées partagées ou un espace latent commun — garantissant la **compatibilité** des calculs de synergie.

Enfin, il est important de **préserver l'évolutivité** de la représentation. Dans un apprentissage continu (cf. chap. 9), l'embedding d'une entité peut être affiné ou ses règles symboliques peuvent être mises à jour à mesure que de nouvelles informations parviennent au réseau. La structure choisie ne doit pas **verrouiller** l'entité dans un format immuable, sous peine de contraindre la plasticité du système. Au contraire, la capacité à réviser ou à enrichir la description de \mathcal{E}_i en cours de route constitue un point essentiel pour l'**adaptation** et la **réorganisation** dynamique.

Ainsi, l'exigence centrale peut se résumer en ces termes : toute entité \mathcal{E}_i doit être munie d'une **description** ou d'un **formalisme** qui :

130. **Capture** les attributs ou les traits indispensables à la distinction et à la mise en relation (images, texte, règles, graphes, etc.).
131. **S'aligne** sur la manière de **définir** ou de **calculer** la synergie $S(i, j)$.
132. **Gère** la potentielle hétérogénéité des types de données dans un même **SCN**.
133. **Permet** une **évolution** ou un **raffinement** si la situation ou les connaissances changent.

Lorsque ce cadre est respecté, la dynamique **auto-organisée** s'appuyant sur $\omega_{i,j}$ peut alors se déployer sans **biais** majeur. À l'inverse, si la représentation d'une partie des entités échoue à décrire les informations clés — ou se trouve en complet décalage par rapport au schéma de similarité attendu —, la **synergie** en sera faussée et la **structuration** en clusters ou en liens cohérents ne pourra émerger convenablement.

Comme on le verra dans la section suivante (3.2.1.2), le calcul de $S(i, j)$ lui-même s'appuie directement sur ces considérations, établissant un continuum entre la forme de l'entité, la fonction de similarité et la mise à jour adaptative du réseau ω . De plus, la section 3.2.2 présentera divers critères (pertinence, complexité, robustesse) qui aident à **opérer** les bons choix de format selon le domaine d'application et les contraintes logicielles.

3.2.1.2. La Fonction de Synergie $S(i, j)$ Dépend Directement de ces Représentations

Le **Deep Synergy Learning (DSL)** se caractérise par l'utilisation d'une **fonction de synergie** $S(i, j)$ permettant d'évaluer à quel point deux entités \mathcal{E}_i et \mathcal{E}_j présentent une **affinité** ou une **compatibilité** mutuelle. Avant même de décider si l'on doit renforcer ou affaiblir la pondération $\omega_{i,j}$, le réseau s'appuie sur les **descripteurs** qui modélisent chacune de ces entités. Ainsi, toute **variation** dans la manière de représenter \mathcal{E}_i (ou \mathcal{E}_j) engendre un **changement** potentiel dans la valeur $S(i, j)$.

Formellement, on peut écrire :

$$S(i, j) = f(\mathbf{r}(i), \mathbf{r}(j)),$$

où $\mathbf{r}(i)$ désigne la représentation de l'entité \mathcal{E}_i (par exemple, un vecteur, un ensemble de règles symboliques, ou un graphe), et la fonction f dépend du **format** de ces représentations (similarité vectorielle, mesure de compatibilité logique, distance structurelle, etc.). Ce **schéma** induit que :

134. Pour des entités vectorielles

Si $\mathbf{r}(i) = \mathbf{x}_i$ et $\mathbf{r}(j) = \mathbf{x}_j$ sont des vecteurs en \mathbb{R}^d , le calcul de $S(i, j)$ repose souvent sur une **distance** (euclidienne, manhattan) ou une **similarité** (cosinus, gaussienne).

$$S_{\text{vect}}(i, j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad \text{ou} \quad S_{\text{vect}}(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

Le **choix** et la **qualité** de l'embedding \mathbf{x}_i influent directement sur le résultat : modifier la dimension du vecteur ou la façon dont on l'a appris (modèle de deep learning, PCA, autoencodeur, etc.) se traduit par une **variation** dans les valeurs de S .

135. Pour des entités symboliques

Si $\mathbf{r}(i) = R_i$ et $\mathbf{r}(j) = R_j$ représentent des **blocs** (ou des ensembles) de règles, d'axiomes ou de concepts, $S(i, j)$ peut alors refléter le **taux de recouvrement** (intersection d'axiomes), la **cohérence** s'il s'agit de les unir, ou toute autre **mesure** de compatibilité symbolique.

$$S_{\text{sym}}(i, j) = \text{Compat}(R_i, R_j).$$

Dans ce cas, l'ajout ou la suppression d'une règle dans R_i affecte le **score** final, pouvant drastiquement modifier la structure de l'auto-organisation : deux entités auparavant jugées proches peuvent soudain être incompatibles, ou inversement.

136. Représentations hybrides

Souvent, le DSL manipule des **descripteurs composites** alliant une partie vectorielle (embedding) et une partie symbolique (règles, concepts). On peut alors définir

$$S_{\text{hybride}}(i, j) = \alpha S_{\text{vect}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où $\alpha \in [0, 1]$ détermine l'importance relative du versant sub-symbolique ou symbolique. Toute **évolution** dans l'une ou l'autre des composantes (\mathbf{x}_i ou R_i) impacte la **valeur** de $S(i, j)$.

137. Cas n-aire ou hypergraphique

Un **DSL** avancé peut, par ailleurs, recourir à une synergie $S(i_1, \dots, i_k)$ dépendant simultanément de la représentation de k entités. Là encore, chaque entité $r(i_m)$ contribue à la **coopération** ou à la **conflictualité** du groupe.

Au plan **mathématique**, on peut décrire la représentation par une application $g: \{\mathcal{E}_i\} \rightarrow \mathcal{X}$, où \mathcal{X} est l'espace ou la famille d'espaces dans lequel vivent les **descripteurs** (vecteurs, ensembles d'axiomes, graphes, etc.). La fonction S devient alors

$$S: \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R},$$

ce qui formalise la dépendance : $S(g(\mathcal{E}_i), g(\mathcal{E}_j))$. Toute **modification** dans le contenu ou la forme de $g(\mathcal{E}_i)$ (c'est-à-dire la représentation de \mathcal{E}_i) répercute un **changement** dans $S(i, j)$. Ce phénomène se propage à la mise à jour des pondérations $\omega_{i,j}$ (voir Chap. 4), car la règle de type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$$

se fonde sur $S(i, j)$ à chaque itération.

Dans l'esprit de l'**auto-organisation**, la structure du **SCN** (existence de clusters, de liens forts ou faibles) résulte alors de la manière dont S rend compte de la **proximité** ou de la **compatibilité** entre entités. Une **représentation** trop sommaire réduit la capacité de S à distinguer des différences subtiles, tandis qu'une représentation plus riche ou évolutive affine la détection des schémas latents et favorise la formation de clusters pertinents.

3.2.2. Critères de Choix

Après avoir mis en évidence le rôle fondamental de la représentation (section 3.2.1), il convient d'examiner **quels critères** président à la sélection d'une forme de description pour les entités \mathcal{E}_i . Ces critères incluent la **pertinence** de la représentation (dimension, nature sub-symbolique vs. symbolique, etc.), la **complexité** de calcul que cela implique, et la **capacité** d'évoluer ou de s'adapter au fil du temps. Nous allons passer en revue chacun de ces aspects.

3.2.2.1. Pertinence (Faut-il un Embedding Large ? un Formalisme Logique ?)

Le premier critère d'évaluation d'une représentation, dans le cadre du **Deep Synergy Learning (DSL)**, consiste à s'interroger sur sa **pertinence** par rapport aux entités visées et aux objectifs de la **synergie**. Un format de description n'est en effet pas intrinsèquement bon ou mauvais ; il faut l'analyser à l'aune du **contenu** réel de chaque entité, du **type** de synergie que l'on souhaite dévoiler, du **niveau** de granularité requis et de la **finalité** (raisonnement symbolique, clustering statistique, etc.). En pratique, plusieurs éléments entrent en jeu :

Alignement sur le contenu réel de l'entité

La **nature** même de l'entité \mathcal{E}_i dicte souvent la **forme** de représentation la plus adéquate. Lorsqu'il s'agit d'une **image**, un vecteur d'embedding dérivé d'un **réseau de neurones** (CNN, ViT) capture des attributs visuels déterminants (formes, textures, objets), alors qu'un simple histogramme d'intensité serait trop restreint pour distinguer des classes complexes. De même, pour un **texte**, un bag-of-words statique efface les relations contextuelles et ne parvient pas à rendre la sémantique profonde, tandis qu'un **embedding contextualisé** (BERT, GPT) intègre précisément ces nuances. Dans tous les cas, la représentation choisie doit couvrir les aspects saillants ou discriminants de l'entité, faute de quoi la **synergie** $S(i, j)$ perd en expressivité et l'auto-organisation en souffre.

Adéquation au type de synergie souhaité

Si la fonction $S(i, j)$ est conçue pour évaluer une **compatibilité logique** (absence de contradiction, mise en cohérence d'axiomes, etc.), un **formalisme symbolique** (règles, ontologie) s'impose. On y gagne en **interprétabilité** et en capacité de raisonnement explicite, par exemple pour vérifier si deux ensembles de règles peuvent coexister. À l'inverse, si l'on cherche à détecter des **similitudes** majoritairement basées sur des correspondances statistiques (images, textes bruités, signaux), un **embedding sub-symbolique** est plus approprié. Le calcul de $S(i, j)$ se fera alors via une mesure de distance ou de similarité vectorielle. Ainsi, la **logique de la synergie** (symbolique vs sub-symbolique) oriente la structure de la représentation, et réciproquement.

Niveau de granularité

Au sein d'une **représentation vectorielle**, la question de la **dimension** (d) se pose : un embedding de grande taille (par exemple 768 dimensions) peut mieux cerner des subtilités, mais risquer de générer de la redondance ou de la sensibilité à la “malédiction de la dimension”. Une dimension plus réduite (type autoencodeur) rend la représentation plus stable et moins coûteuse, au risque de perdre des informations fines. Cette problématique se double d'une éventuelle **hiérarchisation** : on peut choisir un embedding global (une seule dimension du type “chat” ou “voiture”) ou des attributs plus nuancés. Un bon compromis s'efforce de maintenir la diversité descriptive tout en évitant la surcharge dimensionnelle qui nuirait au calcul de $\| \mathbf{x}_i - \mathbf{x}_j \|$.

Possibilité d'exploitation ultérieure

Selon la **finalité** visée, on peut privilégier un certain **formalisme**. Les représentations logiques offrent par exemple la **facilité** de mise en œuvre d'un raisonnement déductif, d'une détection de contradictions ou d'un enrichissement par de nouveaux axiomes. Un vecteur, en revanche, s'intègre mieux dans des **pipelines** de clustering ou de classification statistique, dispose d'un large écosystème d'algorithmes et se manipule aisément pour des calculs de distances ou de produits scalaires. La **composante** “exploitation future” est donc cruciale : veut-on raisonner, annoter, expliquer ? ou désire-t-on avant tout une similarité fluide et scalable ?

Nature des données

Au-delà du choix conceptuel, la **nature** des données elles-mêmes oriente fortement la **représentation**. Un flux **audiovisuel** ou **sensoriel** continu se prête naturellement à une description sub-symbolique (embedding, vecteur de caractéristiques), tandis qu'un ensemble de **règles expertes** s'incarne naturellement dans un formalisme d'ontologie ou de logique descriptive. Même dans des scénarios multimodaux, on veillera à ne pas forcer un encodage unique pour toutes les sources : mieux vaut conserver la spécificité des formats, au besoin en utilisant des modules passerelles (cf. Chap. 3.5 sur les représentations hybrides).

*Mesure indirecte : écart entre S et S^**

La **pertinence** se définit comme la capacité de la représentation (et de la fonction S) à **approximer** une notion “vraie” de similarité S^* . Il s’agit de minimiser la différence $|S^*(i, j) - S(i, j)|$ pour les couples (i, j) qu’on juge objectivement proches. Bien sûr, on ne dispose pas de S^* de manière explicite, mais cette considération sert de **guide**. Plus la représentation reflète les attributs réellement discriminants, plus $S(i, j)$ concorde avec le jugement “naturel” ou “souhaité”. À défaut, si la représentation omet ou brouille les caractéristiques clés, la fonction de synergie s’en voit faussée.

Conclusion

Le **choix** de la représentation ne doit pas être laissé au hasard : un embedding trop succinct ou un formalisme inadapté risque d’appauvrir la synergie et d’engendrer des clusters artificiels ou non pertinents. À l’inverse, un encodage **judicieux** maximise l’alignement entre la structure latente des données et le calcul de $S(i, j)$. Dans les sections suivantes (3.2.2.2, 3.2.3, etc.), nous approfondirons d’autres critères (complexité, robustesse, évolutivité) qui permettront d’affiner le choix de la représentation, afin de bâtir un **DSL** solide et adaptable.

3.2.2.2. Complexité Computationnelle (Dimension du Vecteur, Structure Symbolique)

Au-delà du simple **critère de pertinence**, il est primordial de considérer le **coût** (en temps et en ressources) associé au **calcul** de la synergie $S(i, j)$. Chaque forme de représentation (embeddings vectoriels, logiques symboliques, etc.) induit une **charge** algorithmique différente, qui peut s’avérer déterminante pour l’évolutivité et la stabilité du **Deep Synergy Learning (DSL)**. Lorsque le nombre d’entités n et la taille (ou la complexité) de la représentation augmentent, une mauvaise anticipation de cette complexité peut rendre la mise à jour des pondérations $\omega_{i,j}$ irréalisable dans un temps raisonnable, voire conduire à des instabilités de la dynamique auto-organisée.

1. Dimension du vecteur et coût en $O(n^2 d)$

Lorsqu’on adopte des **représentations sub-symboliques** de type vectoriel, chaque entité \mathcal{E}_i est décrite par un vecteur $\mathbf{x}_i \in \mathbb{R}^d$. La plupart des calculs de synergie (ex. cosinus, gaussienne) entre

deux vecteurs se font en $O(d)$, ce qui reste modeste si d est de taille modérée (e.g., 100–1000). Néanmoins, si le réseau manipule n entités, la comparaison **exhaustive** de toutes les paires (i, j) se chiffre en $O(n^2 d)$. Pour un large n (plusieurs millions) et un vecteur d élevé (512, 1024, etc.), ce volume de calcul peut devenir **prohibitif** :

$$S_{\text{vect}}(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, \quad \text{coût en } O(d) \text{ par comparaison, } O(n^2 d) \text{ au total.}$$

Stratégies :

- 138. **Approximate Nearest Neighbor** (ANN), qui évite de comparer toutes les paires en se concentrant sur les plus proches.
- 139. **Projection de dimension** (PCA, autoencodeurs) pour réduire d .
- 140. **Filtrage ou sparsification** où on n'évalue la synergie que pour des couples (i, j) pré-selectionnés (distants exclus).

2. Structure symbolique et complexité du matching

Pour les entités **symboliques** (règles logiques, axiomes, ontologies), la fonction $S(i, j)$ peut exiger des algorithmes de correspondance ou de vérification de cohérence, voire d'**unification** (en logique). De tels problèmes tombent parfois dans des classes de complexité élevées (NP-Complets, voire plus) :

$$\text{Compat}(R_i, R_j) \rightarrow \text{matching ou unification} \sim O\left(\exp(|R_i| + |R_j|)\right) \text{ (pire cas).}$$

Dans le cas de graphes symboliques ou d'ontologies volumineuses, la vérification d'isomorphisme, de sous-isomorphisme ou l'alignement d'ontologies peut aussi s'avérer $O(\exp(\cdot))$ en théorie. Cela se répercute sur le **calcul total** des synergies, qui peut alors freiner l'évolutivité du DSL :

Stratégies :

- 141. **Restreindre** la logique (Horn clauses, sous-langage OWL décidable) pour limiter la complexité.
- 142. **Indexation** partielle ou agrégation de règles dans des structures plus compactes.
- 143. **Heuristiques** pour unifier ou comparer les entités sans tout explorer (matching local, découpage).

3. Synergie n -aire et explosion combinatoire

Une extension de la synergie binaire $S(i, j)$ vers une synergie n -aire $S(i_1, \dots, i_k)$ (voir chap. 12) fait passer le nombre de sous-ensembles à comparer de $O(n^2)$ vers $O(n^k)$. Quand $k > 2$, l'on risque une **explosion** combinatoire des évaluations, aggravée si chaque entité \mathcal{E}_i est décrite par un graphe

symbolique volumineux. Cet **effet multiplicatif** rend vital l'introduction de mécanismes filtrant quels groupes d'entités vérifier, ou restreignant k à 2 ou 3.

4. Trade-off entre précision et coût

Plus la représentation est **détaillée** (forte dimension vectorielle, ensemble symbolique volumineux, structure fractale), plus la fonction S risque d'être précise et riche, mais au **prix** d'un coût de calcul parfois considérable. En notation abstraite, si $C(\mathbf{r})$ désigne la **complexité** liée à la représentation \mathbf{r} , alors la comparaison de $\mathbf{r}(i)$ et $\mathbf{r}(j)$ s'évalue en $O(C(\mathbf{r}))$. Multipliée par $O(n^2)$ entités, on obtient une charge globale $O(n^2 \times C(\mathbf{r}))$. L'**ingénierie** du DSL doit donc trouver un **compromis** :

$$\min(C_{\text{calcul}}(S) + \alpha \text{Err}),$$

où C_{calcul} mesure le coût de l'évaluation de S et Err l'**erreur** ou la **perte de détails** associée à une représentation plus légère (dimension réduite, règles simplifiées, etc.).

5. Distribution et stockage

Enfin, dans un **SCN** distribué (Chap. 5.7), on doit aussi considérer la **taille** du stockage nécessaire et le **trafic** induit par la récupération des descripteurs $\mathbf{r}(i)$. Conserver un embedding de dimension 512 pour 10^6 entités représente déjà $O(512 \times 10^6)$ flottants, sans parler d'un matching symbolique éventuel sur des milliers de règles. On peut répartir ces données sur plusieurs nœuds, mais il demeure crucial de **minimiser** la quantité d'informations échangées lors du calcul de synergie.

3.2.2.3. Évolutivité (Peut-on Ajouter des Attributs ?)

Un troisième critère majeur dans le choix de la **représentation** pour un système de **Deep Synergy Learning (DSL)** réside dans la **capacité** à faire **évoluer** cette description au fil du temps ou à y **intégrer** de nouveaux attributs. Dans des contextes d'**apprentissage continu**, de **données changeantes** ou de **mise à jour** de règles symboliques, il est fréquent que la représentation d'une entité \mathcal{E}_i doive être modifiée, soit pour ajouter un attribut (une nouvelle coordonnée vectorielle, un nouveau champ sémantique), soit pour réviser les informations déjà présentes. Du point de vue formel, cette **évolutivité** implique la possibilité de mettre à jour la fonction $\mathbf{r}(i)$ en cours de route, sans pour autant briser la dynamique d'auto-organisation basée sur la synergie $S(i, j)$. Différents aspects viennent éclairer ce besoin :

A. Évolutivité dans l'Espace Vectoriel

Lorsqu'on recourt à une **représentation sub-symbolique** de type vectoriel, chaque entité \mathcal{E}_i est associée à un vecteur $\mathbf{x}_i \in \mathbb{R}^d$. Il peut survenir qu'une nouvelle **dimension** doive être incorporée, reflétant un attribut nouvellement découvert. Dans ce cas, on passe d'un embedding $\mathbf{x}_i(t) \in \mathbb{R}^d$ à

un embedding $\mathbf{x}_i(t+1) \in \mathbb{R}^{d+1}$. Toutefois, l'ajout d'une coordonnée aux vecteurs de certaines entités, et pas à d'autres, risque de **déstabiliser** la fonction de similarité :

$\|\mathbf{x}_i(t+1) - \mathbf{x}_j(t+1)\|$ devient ambigu si la dimension change.

La solution la plus straightforward consiste à **initialiser** la nouvelle dimension (attribuée) à une valeur neutre (0, par exemple) pour toutes les entités concernées. Ainsi, l'ensemble des vecteurs se retrouvent à la même dimension $d + 1$. Par la suite, on laisse l'**auto-organisation** adapter ces composantes — par apprentissage supervisé ou non, ou par un mécanisme de recalibration — de sorte que la synergie puisse à nouveau être calculée de façon cohérente. Le point crucial est de **maintenir** un alignement des vecteurs : si certains sont encore en \mathbb{R}^d et d'autres en \mathbb{R}^{d+1} , la dynamique de mise à jour $\omega_{i,j}$ risque de se muer en une source de divergences.

B. Évolutivité dans une Structure Symbolique ou une Ontologie

Dans le cas d'une représentation **symbolique**, où l'entité \mathcal{E}_i se décrit par un **ensemble** de règles, d'axiomes ou de concepts R_i , on peut être amené à **ajouter** de nouvelles règles ΔR pour refléter un enrichissement de la connaissance. Il se produit alors une évolution :

$$R_i(t+1) = R_i(t) \cup \Delta R,$$

qu'il s'agisse de lois supplémentaires (ex. “si fièvre et éruption cutanée alors maladie Z”) ou d'une mise à jour dans une ontologie (nouveau concept, nouvelle relation). Cette **extension** de la description modifie la fonction de compatibilité symbolique. Deux entités antérieurement divergentes peuvent désormais partager un axiome, accroissant $S(i,j)$, ou au contraire révéler une contradiction.

Le **défi** computationnel tient à la nécessité de **recalculer** (ou de mettre à jour) S_{sym} pour toutes les paires (i,j) . Si ΔR est modeste, on peut envisager un algorithme incrémental, au lieu d'une recombinaison exhaustive. Le **DSL** doit s'assurer que la complexité — déjà évoquée en section 3.2.2.2 pour les représentations symboliques — reste gérable, sans bloquer la dynamique globale.

C. Fusion ou Scission d'Entités

Un cas plus particulier concerne la **fusion** de deux entités \mathcal{E}_i et \mathcal{E}_j en une entité unique \mathcal{E}_k , ou la **scission** inverse. Du point de vue de la représentation :

- 144. La **fusion** consiste à **combiner** $\mathbf{r}(i)$ et $\mathbf{r}(j)$ en une unique description $\mathbf{r}(k)$. Dans un cadre vectoriel, on peut, par exemple, prendre un mélange $\alpha\mathbf{x}_i + (1 - \alpha)\mathbf{x}_j$. Pour des règles symboliques, on effectue l'union $R_i \cup R_j$.
- 145. La **scission** scinde $\mathbf{r}(k)$ en deux descriptions $\mathbf{r}(k_a)$ et $\mathbf{r}(k_b)$: un vecteur peut être partitionné, ou un ensemble de règles réparti en deux sous-ensembles.

Cette adaptation modifie la **composition** même du **SCN** (des entités disparaissent, d'autres apparaissent), et entraîne une **réévaluation** de la synergie vis-à-vis de toutes les entités. Le système

doit alors gérer ce chamboulement structurel (réassiguation de liens $\omega_{i,j}$, recalcul partiel de S), et continuer son auto-organisation sans repartir entièrement de zéro.

D. Mise à Jour Incrémentale dans le Temps

De façon générale, un **DSL** opère souvent en **flux** (voir chap. 9), où de nouvelles données ou observations parviennent en continu. La représentation de l'entité \mathcal{E}_i peut alors se réviser itérativement :

$$\mathbf{r}(i, t + 1) = \mathbf{r}(i, t) + \Delta\mathbf{r}_i(t),$$

et chaque entité incorpore ainsi de l'information fraîche (données supplémentaires, règles découvertes, nouveaux attributs). La fonction $S(i, j)$ se met à jour en conséquence, puisqu'un changement dans $\mathbf{r}(i)$ implique forcément un recalcul potentiel de la synergie. La **plastique** du SCN (pondérations $\omega_{i,j}$) réagit immédiatement à ces modifications : si, par exemple, une amélioration de l'embedding rapproche $\mathbf{r}(i)$ de $\mathbf{r}(j)$, la pondération $\omega_{i,j}$ pourra se renforcer.

E. Considérations de Cohérence

Il importe de veiller à la **cohérence** du système lorsqu'on introduit ou modifie la représentation. Si l'on ajoute une composante dans un vecteur, il faut que toutes les entités possèdent cette nouvelle dimension, au moins à titre de placeholder (valeur 0). De même, dans un formalisme logique, un concept n'a de sens que si toutes les entités susceptibles de l'utiliser le reconnaissent en tant qu'attribut ou axiome possible. Faute de synchronisation dans l'évolution du format, la fonction $S(i, j)$ peut devenir impraticable pour certaines paires. Un protocole **échelonné** ou **progressif** peut être imaginé : on introduit la nouvelle dimension (ou règle) dans un sous-ensemble d'entités, tout en garantissant un mécanisme de fallback (ex. coordonnée 0, ou axiome neutre) pour éviter la perte de comparabilité. L'idée demeure d'éviter un changement brutal qui perturbe la dynamique auto-organisée ou invalide des synergies calculées précédemment.

3.2.3. Impact sur la Synergie et la Dynamique

Après avoir discuté des **critères de choix** (pertinence, complexité, évolutivité) dans la section 3.2.2, il est crucial de saisir **comment** la représentation affecte non seulement la **valeur** de la synergie $S(i, j)$, mais aussi la **dynamique** globale du DSL (mise à jour des pondérations $\omega_{i,j}$, formation de clusters, etc.). Le **rôle** de la représentation est ici déterminant : en étant plus ou moins proche entre deux entités, on module directement la **probabilité** qu'elles se renforcent mutuellement dans le réseau.

3.2.3.1. Représentation “proche” \Rightarrow Synergie Potentiellement Élevée, $\omega_{i,j}$ Renforcée

L'un des traits marquants du **Deep Synergy Learning (DSL)** réside dans la façon dont la **proximité** ou la **similarité** entre deux entités \mathcal{E}_i et \mathcal{E}_j (telle qu'elle est traduite par leur **représentation**) induit mécaniquement une **augmentation** de la pondération $\omega_{i,j}$. Cet effet peut se voir directement dans la règle de mise à jour inspirée d'une descente d'énergie, où la **synergie** $S(i,j)$ agit comme un signal de **renforcement**. Plus la représentation montre une concordance marquée, plus la **liaison** entre ces entités se consolide, favorisant l'émergence de **clusters cohérents**.

Exemple dans le cadre sub-symbolique

Lorsque la représentation de chaque entité \mathcal{E}_i est un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$, la synergie $S(i,j)$ peut se définir via une **mesure** de similarité vectorielle (cosinus, distance euclidienne inversée, etc.). Un choix courant est la **similarité cosinus** :

$$S(i,j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}, \quad \text{valeur dans } [-1, 1].$$

Si \mathbf{x}_i et \mathbf{x}_j sont presque colinéaires, c'est-à-dire $\mathbf{x}_i \cdot \mathbf{x}_j \approx \|\mathbf{x}_i\| \|\mathbf{x}_j\|$, alors la valeur de $S(i,j)$ s'approche de 1, traduisant une **forte similarité**. Dans la mise à jour adaptative :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

la présence d'un $S(i,j)$ élevé engendre un **terme positif** $\eta S(i,j)$, dépassant la “pénalisation” $\eta \tau \omega_{i,j}(t)$ si la synergie est assez grande. La **liaison** $\omega_{i,j}$ se voit donc **renforcée** et tend vers un équilibre local

$$\omega_{i,j}^* \approx \frac{S(i,j)}{\tau}.$$

Deux entités largement **proches** (au sens vectoriel) se verront ainsi attribuer une pondération stable et élevée. D'un point de vue **clustering**, elles se retrouveront **attirées** l'une vers l'autre, formant ou rejoignant un **sous-groupe** à plus large échelle dans le Synergistic Connection Network.

Exemple dans le cadre symbolique

La même idée vaut si les entités \mathcal{E}_i et \mathcal{E}_j sont décrites par des **blocs** de règles logiques ou des **concept**s dans une ontologie (R_i et R_j). La fonction de synergie S_{sym} évalue la **compatibilité** ou la **cohérence** entre ces deux ensembles :

$$S_{\text{sym}}(i,j) = \text{Compat}(R_i, R_j),$$

qui peut mesurer la proportion d'axiomes ou de règles partagées, ou l'absence de contradictions sémantiques. Plus cette **intersection** conceptuelle est forte, plus la synergie atteint des valeurs élevées, incitant la pondération $\omega_{i,j}$ à croître. Là encore, on assiste à la **formation** progressive d'un

cluster : des entités possédant des blocs de règles similaires (ou présentant une forte compatibilité) se verront reliées par des liaisons renforcées, favorisant leur regroupement.

Conséquence : du renforcement local vers le cluster global

De manière générale, la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

assure que toute **paires** (i, j) bénéficiant d'une **synergie** élevée (qu'elle soit sub-symbolique ou symbolique) subisse un **renforcement** récurrent, tant que $S(i,j)$ demeure au-dessus d'un certain point de compensation $\tau \omega_{i,j}$. Dans un environnement multi-entités, de **nombreuses** paires peuvent progressivement s'attirer, formant un **noyau** de liens forts. En se propageant, cet effet aboutit à la **coalescence** d'un ensemble d'entités partageant des attributs similaires ou des règles logiques compatibles, c'est-à-dire un **cluster** ou une **communauté** nettement identifiable dans le **SCN**.

Seuil et Trigger de Liaison

Il n'est pas rare de définir, de manière explicite ou implicite, un **seuil** θ pour la synergie $S(i,j)$. Lorsqu'elle dépasse θ , la croissance de $\omega_{i,j}$ s'accélère. Cette heuristique peut être formalisée, par exemple, en introduisant un **terme** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [F(S(i,j), \theta) - \tau \omega_{i,j}(t)],$$

où la fonction $F(S(i,j), \theta)$ vaut 0 en dessous du seuil θ , et augmente rapidement au-delà. Cette approche accentue le **contraste** : des entités proches franchissent le seuil et se relient fortement, tandis que d'autres, trop différentes, ne voient pas leur liaison progresser. La réussite de ce **filtrage** dépend à nouveau de la **qualité** de la représentation initiale : si celle-ci n'est pas assez discriminante, ce système de seuil peut mener à des associations erronées ou au contraire trop restreintes.

3.2.3.2. Risque de Confusion si la Représentation est Trop Floue ou Trop Simplifiée

En miroir de l'idée qu'une **proximité** marquée dans l'espace de représentation favorise une **synergie** élevée — laquelle renforce la pondération $\omega_{i,j}$ —, il faut souligner la possibilité d'un **effet inversé** si la représentation n'est pas assez **discriminante**. En d'autres termes, quand la description d'une entité \mathcal{E}_i est trop simplifiée, peu informative ou compressée, deux entités **objectivement** différentes risquent d'apparaître artificiellement "proches", gonflant leur synergie $S(i,j)$ sans réel fondement. Cette confusion conduit alors le **Synergistic Connection Network (SCN)** à les regrouper dans un même cluster de manière erronée.

Représentation sub-symbolique et dimensionnalité insuffisante

Un cas courant se rencontre dans les approches **sub-symboliques** : on associe à chaque entité \mathcal{E}_i un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$. Si la **dimension** d est trop faible pour couvrir la diversité des attributs, ou si l'on a appliqué une **réduction de dimension** trop agressive (PCA drastique, autoencodeur trop compact, etc.), des entités pourtant distinctes finissent par se projeter **proches** dans \mathbb{R}^d . Mathématiquement, la distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ ou la similarité cosinus $\frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}$ donne alors une valeur trompeusement élevée. La synergie $S(i,j)$ en résulte **surévaluée** et le réseau en vient à renforcer $\omega_{i,j}$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Ces rapprochements “factices” introduisent une confusion dans la **dynamique d’auto-organisation** : au lieu de séparer clairement des entités distinctes, le SCN tend à créer des liens forts entre elles, formant un **cluster** incohérent. On peut aboutir à des macro-groupes qui englobent quantité d'entités hétérogènes, ou à des groupements aléatoires privés de sens réel.

Représentation symbolique simpliste

Dans un **cadre symbolique**, un problème similaire survient si l'on ne conserve qu'un **jeu restreint** de règles ou d'axiomes pour chaque entité. Par exemple, la fonction de compatibilité $\text{Compat}(R_i, R_j)$ peut reposer sur la part d'axiomes communs entre deux ensembles R_i et R_j . Si ces ensembles se réduisent à quelques **règles** très générales (ex. “tout objet peut être rouge ou bleu”), alors de multiples entités paraîtront artificiellement “compatibles”, faute de disposer de règles plus spécialisées pour les distinguer. La synergie $S(i,j)$ demeure élevée pour bon nombre de couples, masquant les divergences réelles que des règles plus riches auraient mis en évidence.

Cette situation entraîne le même effet que dans l'espace vectoriel compressé : des **liaisons** $\omega_{i,j}$ s'en trouvent gonflées de façon injustifiée, menant à un **cluster unique** ou à des regroupements inadéquats. Dans le cas d'une ontologie appauvrie, deux concepts très différents peuvent apparaître près l'un de l'autre en raison d'un socle minimal d'axiomes partagés.

Distorsion de la synergie

D'un point de vue **mathématique**, on peut considérer qu'il existe une “**vraie**” similarité $S^*(i,j)$ que l'on souhaiterait approximer par $S(i,j)$. Une représentation **trop floue** augmente l'écart $|S^*(i,j) - S(i,j)|$. Cela se traduit par deux risques :

146. **Surévaluation** : Deux entités très différentes ($S^* \approx 0$) apparaissent “proches” en S , conduisant à un renforcement indu de $\omega_{i,j}$ et à la formation de liens illusoires.
147. **Sous-évaluation** : Inversement, des entités en réalité similaires ($S^* \approx 1$) reçoivent un score S faible, se retrouvant dispersées dans des clusters distincts.

Dans les deux cas, la **dynamique** du DSL dérive et forme des structures contraires à la **réalité** des entités (au sens de S^*). Le réseau se **désorganise** ou, pire, se retrouve dans un état quasi homogène où presque tout le monde est “lié” de façon grossière.

Clusters peu informatifs ou pseudo-homogénéité

Concrètement, une représentation simpliste ou appauvrie donne souvent lieu à des **clusters** anormalement étendus, mélangeant des entités sans vrai rapport. Sur le plan algorithmique, la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

va favoriser des liaisons multiples, lesquelles finissent par “coller” ensemble un grand sous-ensemble. Lorsqu'on visualise la matrice $\{\omega_{i,j}\}$, elle tend à se **remplir** (saturer), aboutissant à une quasi-absence de différenciation.

Comment éviter ce problème ?

D'un point de vue **ingénierie**, il convient :

- 148. D'adopter une **représentation** plus riche (augmenter la dimension vectorielle, introduire davantage d'attributs symboliques).
- 149. De vérifier que les **attributs** sélectionnés sont véritablement discriminants (ne pas se contenter d'une statistique globale).
- 150. D'envisager des **mécanismes** de filtrage ou d'inhibition (cf. chap. 4 et 7) pour ne pas laisser les synergies de faible consistance perturber la structure.
- 151. Dans certains cas, de recourir à des **tests** externes ou à un oracle pour détecter les couples (i,j) qui seraient anormalement jugés “proches”, et rétroagir sur la représentation.

3.2.3.3. Rôle de la Normalisation ou de la Calibration des Entités

Lorsque l'on conçoit un **Deep Synergy Learning (DSL)**, il ne suffit pas de sélectionner une représentation (vecteur, règles symboliques, etc.) pour chaque entité E_i . Il est également crucial de **rendre** ces différentes descriptions **comparables**, de telle sorte que la fonction de synergie $S(i,j)$ ne soit pas faussée par des échelles numériques trop divergentes ou par des attributs symboliques surpondérés. C'est ici qu'interviennent la **normalisation** et la **calibration**, qui visent à **harmoniser** la magnitude ou la pondération des descripteurs et à éviter une domination injustifiée de certaines composantes.

A. Problématique de l'Échelle (Cas Sub-Symbolique)

Lorsqu'une entité \mathcal{E}_i est décrite par un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$, on peut rencontrer le problème de **dimensions** inégales : certaines coordonnées varient dans un intervalle restreint (par exemple entre 0 et 1), tandis que d'autres couvrent un éventail bien plus large (ex. 100 à 10 000). Pour la distance euclidienne,

$$\|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{\ell=1}^d (x_{i,\ell} - x_{j,\ell})^2},$$

une composante numériquement très ample se met à **dominer** la distance, rendant négligeables les écarts sur les autres composantes. Cela peut conduire à des mesures de **synergie** $S(i,j)$ (souvent définies comme une fonction décroissante de $\|\mathbf{x}_i - \mathbf{x}_j\|$) faussées, car la “dimension la plus grande” éclipse l'information apportée par les autres.

Pour éviter cette situation, on recourt à des **méthodes de normalisation** :

- 152. **Min–max scaling** pour ramener chaque coordonnée dans [0,1],
- 153. **Z-score** où l'on retranche la moyenne et on divise par l'écart-type,
- 154. **ℓ_2 -normalisation** (projection sur la sphère unitaire) : $\mathbf{x}'_i = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}$.

Ces approches garantissent que la fonction de synergie ne soit pas **dominée** par un attribut particulier. À titre d'exemple, dans une similarité cosinus, on impose souvent $\|\mathbf{x}_i\| = 1$, alors $jS(i,j) = \mathbf{x}_i \cdot \mathbf{x}_j$. Toute différence d'échelle disparaît, et la distance ou la similarité répercute mieux les différences sémantiques réelles.

B. Homogénéité entre Entités

Au sein du **Synergistic Connection Network** (SCN), on peut avoir un grand nombre d'entités analogues (ex. plusieurs capteurs) — certains capteurs peuvent naturellement renvoyer des valeurs plus fortes (en volts ou en dB) tandis que d'autres restent dans des gammes plus faibles. Sans un processus de **calibration**, un capteur “fort” dominera la distance ou la similarité, amenant la pondération $\omega_{i,j}$ à se renforcer majoritairement pour ces entités, même si la différence n'est qu'un **artefact** d'échelle.

De même, en logique symbolique, si l'on attribue des **poids** inégaux aux règles (ex. une “règle principale” vaut 1000, les autres valent 1), alors le calcul de $\text{Compat}(R_i, R_j)$ sera le plus souvent dicté par la présence ou l'absence de cette règle “majeure”. Le réseau pourra, là encore, **exagérer** les proximités lorsque cette règle est partagée, formant ainsi des clusters artificiels.

La **calibration** consiste à imposer une **cohérence** dans la pondération : par exemple, normaliser la somme des poids (symboliques) pour que $\sum_{r \in R_i} w(r) = 1$. On peut alors définir :

$$S_{\text{sym}}(i, j) = \frac{\sum_{r \in R_i \cap R_j} w(r)}{\sum_{r \in R_i \cup R_j} w(r)},$$

de sorte qu'une règle unique n'emporte pas la totalité du calcul.

C. Éviter la “Dérive” au Fil du Temps

Dans un **environnement évolutif** (chap. 9), les entités peuvent voir leur représentation se **modifier** : un embedding peut “s’étirer” si le modèle d’apprentissage continu affine ses paramètres ; des règles supplémentaires peuvent être ajoutées à un ensemble symbolique. Si aucun mécanisme de recalibrage n’existe, on court le risque qu’**une** dimension ou **un** attribut gonfle sans limite, bouleversant progressivement la mesure de synergie. Cela se traduit par des **oscillations** dans la mise à jour de $\omega_{i,j}$ ou par l’absorption d’entités disparates dans un même cluster.

Une parade est de fixer un protocole de **renormalisation** ou de “ré-étalonnage” à intervalles réguliers, afin de maintenir l’ensemble des entités dans un même **espace** de comparaison :

$$\mathbf{r}(i, t + 1) = h(\mathbf{r}(i, t)),$$

où h effectue la recalibration nécessaire (remise à l’échelle, recentrage). On peut choisir un **pas** de recalibration adapté ou mettre en place un pilotage heuristique (ex. si la norme moyenne des vecteurs dépasse un seuil, on opère une projection globale).

D. Cas Multimodal ou Hybride

Dans des scénarios **multimodaux**, on a souvent besoin de **fusionner** un score de similarité sub-symbolique (ex. cosinus entre embeddings) et un score symbolique (ex. compatibilité de règles). Si l’on ne borne pas ou ne normalise pas ces scores, l’un peut dominer l’autre. Par exemple, un cosinus variant dans $[-1, 1]$ se retrouve dilué face à un score symbolique qui peut atteindre 10 ou 100.

Une **stratégie** consiste à normaliser chaque sous-score dans $[0, 1]$ ou $[-1, 1]$, puis à combiner linéairement ou par un noyau plus sophistiqué :

$$S_{\text{hybride}}(i, j) = \alpha S_{\text{vect}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où α est un paramètre. On évite ainsi que le bloc logique (ou le vecteur) n’occupe toute la scène. L’**harmonisation** est d’autant plus critique si le DSL doit manipuler diverses modalités (vision, audio, texte, règles expertes) dans un même SCN.

3.3. Représentations Sub-Symboliques

Les approches sub-symboliques, basées sur des **vecteurs** ou des **embeddings**, constituent l'une des manières les plus répandues de décrire des entités \mathcal{E}_i dans un contexte d'apprentissage automatique. Dans le cadre du DSL (Deep Synergy Learning), elles présentent l'avantage de se prêter à un **calcul** aisément quantifiable de la similarité (voire de la distance) entre entités, conditionnant ainsi la **synergie** $S(i, j)$. Dans cette section, nous étudierons tout d'abord les **vecteurs** et **embeddings** (3.3.1), puis nous aborderons des **méthodes avancées** (autoencodeurs, transformers, etc.) (3.3.2), avant de voir en détail le **calcul** de la synergie entre vecteurs (3.3.3) et la **gestion** du bruit ou de l'évolution de ces représentations (3.3.4).

3.3.1. Vecteurs et Embeddings

La représentation **vectorielle** est sans doute la plus intuitive dans un grand nombre d'applications : on code chaque entité par un point $\mathbf{x}_i \in \mathbb{R}^d$. Cette approche, déjà ancienne en classification ou en clustering, prend une **nouvelle dimension** avec l'émergence des **embeddings profonds**, capables de saisir des caractéristiques complexes dans les images, les textes ou l'audio.

3.3.1.1. Origine : CNN (images), Word Embeddings (textes), Spectrogrammes (audio)

Dans le cadre d'un **Deep Synergy Learning (DSL)**, une grande partie des représentations sub-symboliques est obtenue par l'extraction d'**embeddings** à l'aide de **modèles neuronaux** spécialisés. L'idée fondamentale consiste à transformer un objet initial – qu'il s'agisse d'une image, d'un texte ou d'un segment audio – en un **vecteur** ou un **tenseur** qui capture les **caractéristiques** les plus pertinentes de l'information. Ce vecteur permet ensuite de calculer la **mesure** de synergie, notée $S(i, j)$, qui reflète la proximité ou la compatibilité entre deux entités \mathcal{E}_i et \mathcal{E}_j . Diverses approches sont utilisées pour générer ces embeddings, et nous détaillons ici trois cas typiques.

Dans le domaine de l'analyse d'images, les **Convolutional Neural Networks (CNN)** ont largement popularisé l'extraction de **vecteurs** de caractéristiques. Un réseau convolutionnel, tel que VGG, ResNet, Inception ou encore Vision Transformer, procède à travers une série de couches de **convolution** et de **pooling** à l'extraction progressive de « feature maps » qui condensent l'information visuelle en attributs de plus en plus abstraits, tels que les **textures**, les **bords** ou encore les **formes**. Au terme de ce processus, on obtient un **embedding global** de dimension d (typiquement 256, 512 ou 1024), représenté par

$$\text{CNN}: \quad \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^d,$$

où $H \times W$ indique la taille de l'image, C le nombre de canaux (par exemple, 3 pour une image RGB), et d la dimension du vecteur. Ce **descripteur** capture ainsi des aspects sémantiques essentiels, et dans un DSL, deux images dont les embeddings \mathbf{x}_i et \mathbf{x}_j se montrent très similaires (par exemple, mesurées via la **similarité cosinus** ou une distance euclidienne inversée) conduisent à une synergie élevée, entraînant le renforcement de la pondération $\omega_{i,j}$ dans le SCN et favorisant l'auto-organisation en clusters d'images visuellement cohérents.

Dans le domaine du **Traitement Automatique du Langage Naturel (TALN)**, la représentation des textes s'appuie sur les **word embeddings** tels que Word2Vec et GloVe, ainsi que sur des modèles contextuels plus récents comme BERT, GPT ou T5. Plutôt que d'utiliser des représentations statiques comme les vecteurs one-hot, on projette chaque mot ou token dans un **espace** de dimension d selon la relation

$$\mathbf{w}: \{\text{mots ou tokens}\} \rightarrow \mathbb{R}^d.$$

Ainsi, deux mots ou expressions d'une **signification** similaire se trouvent naturellement proches dans cet espace, ce qui facilite le calcul de distances ou de **similarités**. De plus, pour obtenir une représentation d'une phrase ou d'un document, il est courant d'agréger les embeddings des mots, par exemple par la moyenne ou en extrayant le vecteur associé au token [CLS] dans les modèles Transformer. Cette approche permet d'assigner à chaque segment textuel un vecteur \mathbf{x}_i qui, lorsqu'il est comparé à un autre vecteur \mathbf{x}_j , donne une mesure de synergie reflétant la **parenté sémantique** entre les textes.

Pour les données **audio**, l'extraction d'embeddings se fait souvent à partir d'un **spectrogramme** ou par l'usage de modèles neuronaux spécialisés, tels que SoundNet ou des Audio Transformers. Le signal audio $a(t)$ est segmenté et transformé par un modèle spécifique, conduisant à un embedding

$$\mathbf{x}_s = \text{AudioModel}(a_s(t)) \in \mathbb{R}^d,$$

qui capture des caractéristiques acoustiques comme le **timbre** et la **prosodie**. Dans un DSL, si deux segments audio présentent des embeddings similaires, la synergie $S(i, j)$ est alors élevée, et la pondération correspondante $\omega_{i,j}$ se renforce, permettant ainsi la formation de clusters cohérents de signaux acoustiques ou la fusion d'une modalité audio avec d'autres modalités dans un cadre multimodal.

D'un point de vue **formel**, après avoir extrait les vecteurs \mathbf{x}_i pour chaque entité, on définit la **mesure** de similarité par exemple par la formule gaussienne

$$S(i, j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|)$$

ou par la formule du **produit scalaire normalisé**

$$S(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

Lorsque ces scores de similarité sont intégrés dans la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

le DSL renforce les liens entre les entités dont les embeddings sont proches dans l'espace, ce qui conduit à la formation de **clusters** auto-organisés reflétant la **cohérence** des caractéristiques sous-jacentes.

La **force** de cette approche réside dans la capacité des **embeddings** générés par ces modèles neuronaux à capturer une **richesse sémantique** et des aspects contextuels qui permettent d'identifier avec précision les similitudes entre entités issues de domaines variés (vision, langage,

acoustique). Cependant, le succès de cette méthode dépend fortement de la **qualité** des embeddings, laquelle doit être suffisamment robuste pour résister au bruit et capable de généraliser les caractéristiques essentielles des données. Ces principes, approfondis dans le Chapitre 3, soulignent l'importance de disposer de représentations vectorielles de haute qualité pour garantir une auto-organisation efficace au sein du DSL.

En conclusion, les **embeddings** extraits par des CNN, par des modèles de **word embeddings** ou par des techniques d'extraction à partir de **spectrogrammes** constituent le socle de la représentation sub-symbolique dans le DSL. Ils permettent un calcul direct de la **proximité** entre entités, facilitant ainsi la formation de clusters cohérents dans le SCN, et leur qualité détermine en grande partie le succès de l'auto-organisation, rendant indispensable une attention particulière aux techniques d'extraction et à la robustesse des représentations.

3.3.1.2. Avantages : Calcul Aisé de Similarité (Cosinus, Distance Euclidienne)

Dans le cadre du **Deep Synergy Learning (DSL)**, chaque entité \mathcal{E}_i est représentée par un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$ qui est issu d'un modèle d'apprentissage profond tel qu'un CNN, un transformeur pour le langage ou encore un modèle audio pour les spectrogrammes. L'un des atouts majeurs de cette représentation sub-symbolique réside dans la **simplicité algorithmique** avec laquelle il est possible de calculer la **similarité** ou la **distance** entre ces vecteurs. Cette facilité de calcul joue un rôle central dans la formation et l'auto-organisation du **Synergistic Connection Network (SCN)**, puisque la **synergie** $S(i, j)$ entre deux entités est directement fonction de la proximité mesurée dans l'espace vectoriel, et plus cette proximité est élevée, plus la **pondération** $\omega_{i,j}$ tend à être renforcée selon la règle de mise à jour du SCN.

D'un point de vue mathématique, la **similarité cosinus** est définie par

$$S_{\cos}(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

cette mesure se concentrant sur l'**angle** entre \mathbf{x}_i et \mathbf{x}_j plutôt que sur leur norme, ce qui est particulièrement pertinent dans des contextes où la magnitude du vecteur n'est pas un indicateur de la **pertinence** ou de la **similarité**. En outre, une transformation linéaire de l'intervalle $[-1, 1]$ permet d'obtenir un score de similarité dans $[0, 1]$, par exemple

$$S(i, j) = \frac{1 + S_{\cos}(i, j)}{2},$$

ce qui garantit une interprétation directe dans le cadre de la mise à jour des pondérations. Par ailleurs, la **distance euclidienne** est définie par

$$d_{\text{eucl}}(i, j) = \|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{k=1}^d (x_{i,k} - x_{j,k})^2},$$

et cette distance, en quantifiant la longueur du segment reliant \mathbf{x}_i à \mathbf{x}_j , peut être convertie en un score de similarité par l'application d'une fonction de décroissance exponentielle, comme le montre la formule

$$S_{\text{gauss}}(i, j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|^2),$$

où $\alpha > 0$ est un paramètre réglable. Ces deux formules sont linéaires en la dimension d et reposent sur des opérations de multiplication–accumulation, qui sont bien adaptées aux architectures de calcul modernes telles que les GPU ou les bibliothèques optimisées en BLAS, permettant ainsi un **calcul** rapide même pour de grands ensembles d'entités. La **simplicité** de ces mesures permet une implémentation efficace de la **mise à jour** des pondérations dans le SCN selon la formule

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

ce qui conduit à un renforcement des liens entre les entités qui se rapprochent dans l'espace vectoriel. L'avantage de cette approche réside également dans le fait qu'elle permet une **interopérabilité** naturelle avec un grand nombre d'algorithmes existants, notamment ceux de **clustering** (tels que K-means, DBSCAN) et des techniques d'**approximate nearest neighbor** (comme FAISS ou Annoy), ce qui facilite l'analyse et la visualisation des regroupements obtenus via des méthodes de réduction de dimension telles que PCA, t-SNE ou UMAP. De plus, cette méthode se prête aisément à des scénarios **multimodaux** : en projetant des données provenant de différentes sources dans un espace commun, on peut appliquer de manière uniforme la logique de calcul de la similarité, assurant ainsi une intégration homogène et une gestion efficace de la **variabilité** des données. Finalement, en ajustant les paramètres des kernels (comme α dans le cas du noyau gaussien), il est possible d'optimiser la **sensibilité** du DSL aux distances entre vecteurs, garantissant ainsi que la **dynamique** de mise à jour des pondérations reflète de manière fine les structures sous-jacentes des données.

En conclusion, l'utilisation de **calculs** basés sur la **similarité cosinus** ou la **distance euclidienne** permet au DSL de bénéficier d'un calcul aisé et rapide de la **proximité** entre entités, facilitant la formation de clusters auto-organisés dans le SCN. La simplicité algorithmique de ces méthodes, combinée à leur capacité d'être parallélisées sur des architectures modernes, confère au système une efficacité notable en termes de **scalabilité** et d'**interopérabilité** avec d'autres algorithmes de traitement de données, tout en offrant une flexibilité adaptée aux environnements multimodaux.

3.5.3.3. Inconvénients : Dimension Élevée, Parfois Peu Interprétables

L'utilisation de **représentations sub-symboliques** sous forme de vecteurs d'embeddings, générés par des modèles neuronaux tels que les CNN ou les Transformers, offre indéniablement des avantages en termes de simplicité et d'efficacité du calcul de la synergie dans \mathbb{R}^d . Toutefois, ces méthodes présentent plusieurs inconvénients qui peuvent poser des problèmes de **scalabilité**, de **stabilité** et d'**explicabilité**. La discussion qui suit s'articule autour de quatre axes majeurs.

A. Dimension Élevée : Risques et Limitations

Les modèles neuronaux modernes produisent des embeddings dont la dimension peut atteindre plusieurs centaines voire milliers (par exemple, 512, 768 ou 1024 dimensions). Lorsque la fonction

de synergie $\mathbf{S}(i, j)$ est calculée de manière naïve pour toutes les paires (i, j) au sein d'un Synergistic Connection Network (SCN) composé de n entités, le coût de calcul devient de l'ordre de $O(n^2 \times d)$. Ainsi, si n est très grand – par exemple, plusieurs millions d'entités – et d élevé, le temps de calcul peut devenir prohibitif, comme l'illustre l'estimation

$$\text{Coût} \approx n^2 \times d.$$

De plus, la malédiction de la dimension peut rendre les distances moins discriminantes, puisque dans des espaces de très haute dimension, les distances entre la plupart des points tendent à converger vers une valeur similaire. Cette uniformisation des distances peut aboutir à une dynamique d'auto-organisation moins stable, avec des clusters qui se forment de manière peu distincte ou qui présentent des frontières floues. Face à ces difficultés, des approches telles que l'utilisation d'algorithmes d'**approximate nearest neighbors**, de techniques de **sparsification** ou de mécanismes de **sélection** ne traitant que les paires prometteuses sont souvent nécessaires afin de réduire la charge de calcul.

B. Opacité et Faible Interprétabilité

Les embeddings issus de réseaux neuronaux sont le produit d'opérations complexes et hiérarchiques réparties sur de multiples couches (convolutions, mécanismes d'attention, etc.). Cette complexité rend souvent l'interprétation de chaque coordonnée de l'embedding difficile. Il n'est pas aisément de déterminer si une dimension particulière de \mathbf{x}_i correspond à une caractéristique précise, telle qu'une texture, une forme ou un concept lexical. De surcroît, lorsqu'on constate qu'un score de similarité élevé existe entre deux vecteurs \mathbf{x}_i et \mathbf{x}_j , il est ardu d'attribuer de manière explicite la cause de cette proximité – qu'il s'agisse de la couleur, de la catégorie ou d'une co-occurrence textuelle. Cette opacité pose un réel problème dans des domaines exigeant une forte **explicabilité** ; ainsi, pour des applications critiques (médicales, industrielles, etc.), il est impératif de pouvoir justifier la formation d'un cluster ou la liaison entre deux entités. Même si des techniques de réduction de dimension, telles que PCA, UMAP ou t-SNE, permettent d'obtenir une vue globale des distributions, elles n'améliorent guère l'interprétation des attributs à l'échelle individuelle.

C. Instabilité des Embeddings et Variations Contextuelles

Les embeddings neuronaux peuvent varier sensiblement en fonction du **contexte** ou des modifications apportées lors de la phase d' entraînement. Dans le domaine du traitement du langage naturel, par exemple, une légère modification du corpus d' entraînement ou un nouveau fine-tuning peut entraîner des décalages dans les positions des mots ou des tokens dans l'espace vectoriel. De même, en vision par ordinateur, l'adaptation d'un réseau pour intégrer de nouvelles classes d'images peut modifier la distribution finale des embeddings \mathbf{x}_i . De telles variations contextuelles impactent directement la mesure de la synergie, que ce soit par la modification de la distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ ou par le produit scalaire $\mathbf{x}_i \cdot \mathbf{x}_j$. Cette instabilité peut ainsi perturber la dynamique du SCN, induisant des réorganisations brutales ou imprévues dans l'auto-organisation, et posant un défi pour la stabilité à long terme du DSL.

D. Nécessité d'un Prétraitement Potentiellement Coûteux

L'obtention de ces embeddings repose sur l'utilisation de modèles préentraînés, tels que des réseaux CNN pour les images ou des Transformers pour les textes, qui nécessitent une infrastructure de calcul robuste (GPU/TPU) et des ressources importantes pour l'entraînement ou

l’inférence. Si l’objectif est de concevoir un DSL autonome fonctionnant sur des ressources matérielles limitées, le coût en termes de calcul et de temps associé au prétraitement des données peut s’avérer prohibitif. De plus, le déploiement d’un tel système requiert la mise en place d’un pipeline complet, capable de fournir en continu des embeddings de haute qualité, ce qui représente une contrainte supplémentaire en termes d’ingénierie et d’infrastructure.

Conclusion

L’utilisation de représentations sub-symboliques sous forme d’embeddings dans un DSL présente indéniablement des avantages, notamment en facilitant le calcul de la synergie dans \mathbb{R}^d et en permettant l’auto-organisation du réseau par des opérations vectorielles efficaces. Cependant, ces avantages s’accompagnent de plusieurs inconvénients majeurs. La dimension élevée des embeddings conduit à une complexité algorithmique accrue, en particulier lorsque le nombre d’entités est très important, et la malédiction de la dimension peut nuire à la discrimination des distances. De plus, l’opacité inhérente à ces représentations rend difficile l’attribution de causes explicites aux similarités observées, ce qui limite leur interprétabilité. Par ailleurs, les variations contextuelles et l’instabilité potentielle des embeddings peuvent perturber la dynamique du réseau, et le prétraitement nécessaire pour obtenir ces représentations impose des contraintes matérielles et techniques non négligeables. Dans la pratique, il est essentiel de mettre en œuvre des techniques de réduction de dimension, des heuristiques d’approximate nearest neighbors et des stratégies de normalisation afin d’atténuer ces écarts et d’assurer une robustesse globale du DSL.

3.3.2. Autoencodeurs, Transformers et Approches Avancées

Les vecteurs et embeddings présentés en section 3.3.1 constituent déjà une base solide pour représenter des entités de nature variée (images, textes, sons). Toutefois, les **techniques** d’apprentissage ne cessant d’évoluer, il est possible d’aller **plus loin** en recourant à des méthodes avancées qui extraient des embeddings encore plus **expressifs** ou plus **compacts**. Dans cette section, nous verrons d’abord (3.3.2.1) comment les **autoencodeurs** peuvent servir à la **réduction dimensionnelle** ou à l’**extraction de features**, puis (3.3.2.2) comment les **Transformers** (BERT, GPT, ViT) produisent des embeddings **contextuels**, et enfin (3.3.2.3) nous soulignerons **l’intérêt** de ces méthodes avancées pour le DSL, au prix toutefois d’une complexité de calcul plus élevée.

3.3.2.1. Autoencodeurs pour Réduction Dimensionnelle ou Extraction de Features

Dans le cadre d’un **Deep Synergy Learning (DSL)**, l’extraction d’informations pertinentes à partir de représentations sub-symboliques constitue un enjeu majeur pour l’efficacité du processus d’auto-organisation. Les **autoencodeurs (AE)** se présentent comme un outil fondamental permettant d’apprendre, de manière non supervisée, un **code latent \mathbf{z}** qui capture l’essence d’un vecteur d’entrée \mathbf{x} . Cette démarche offre deux avantages essentiels pour le DSL : la **réduction de dimension** et l’**extraction de features discriminantes**.

A. Principe et Architecture d’un Autoencodeur

Un autoencodeur se compose typiquement de deux modules complémentaires. Le premier, l’**encodeur E** , est une fonction $E: \mathbb{R}^d \rightarrow \mathbb{R}^k$ qui reçoit un vecteur d’entrée $\mathbf{x} \in \mathbb{R}^d$ et produit un **code latent $\mathbf{z} \in \mathbb{R}^k$** , où $k < d$ dans le but de compresser l’information. Le second module, le **décodeur D** , opère la transformation inverse, $D: \mathbb{R}^k \rightarrow \mathbb{R}^d$, et tente de reconstruire le vecteur

d'entrée à partir de ce code latent, produisant ainsi $\hat{\mathbf{x}} = D(E(\mathbf{x}))$. L'apprentissage se réalise en minimisant une **fonction de perte de reconstruction** $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}})$, souvent formulée comme la moyenne des erreurs quadratiques, ce qui s'exprime par

$$\min_{E, D} \frac{1}{N} \sum_{i=1}^N \mathcal{L}(\mathbf{x}_i, D(E(\mathbf{x}_i))).$$

Cette optimisation force l'encodeur à extraire les facteurs de variation les plus pertinents de \mathbf{x} , tout en éliminant le bruit et les détails superflus, aboutissant ainsi à une représentation compacte et significative.

B. Avantages : Réduction de Dimension et Extraction de Traits

L'imposition d'un goulot d'étranglement, où la dimension du code latent k est inférieure à celle de l'entrée d , engendre deux bénéfices notables pour le DSL. D'une part, la **réduction dimensionnelle** permet de diminuer significativement le coût de calcul lors de l'évaluation des distances ou similarités entre entités. En effet, le calcul de la distance entre deux codes latents, par exemple $\|\mathbf{z}_i - \mathbf{z}_j\|$, se réalise en $O(k)$ plutôt qu'en $O(d)$, ce qui est particulièrement avantageux lorsque le réseau traite un grand nombre d'entités. D'autre part, en apprenant à reconstruire \mathbf{x} à partir de \mathbf{z} , l'autoencodeur extrait des **features discriminantes** qui résument les aspects sémantiques les plus importants du vecteur d'origine, facilitant ainsi la formation de clusters cohérents et la détection de similarités véritables dans le SCN.

C. Variantes : Denoising, Sparse et Variational Autoencoder

Il existe plusieurs variantes d'autoencodeurs qui visent à améliorer la robustesse et la qualité des représentations latentes.

Premièrement, le **Denoising Autoencoder** introduit du bruit dans \mathbf{x} durant l'entraînement et apprend à reconstruire la version nettoyée de la donnée. Ce procédé confère au code latent \mathbf{z} une meilleure tolérance aux perturbations, ce qui est particulièrement utile dans des environnements où les données sont bruitées.

Ensuite, le **Sparse Autoencoder** incorpore une contrainte de sparsité sur \mathbf{z} – par exemple, via une pénalisation de la norme L_1 – de sorte que seule une fraction limitée des neurones s'active pour chaque donnée, ce qui peut améliorer l'interprétabilité des caractéristiques extraites et favoriser une séparation plus nette des classes.

Enfin, le **Variational Autoencoder (VAE)** adopte une approche probabiliste en imposant que le code latent suive une distribution prédéfinie, généralement gaussienne, c'est-à-dire $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \sigma^2 I)$. Cette formulation permet non seulement de générer de nouvelles données, mais aussi d'organiser l'espace latent de manière plus lisse, facilitant ainsi la continuité et la robustesse des clusters formés lors de l'auto-organisation.

D. Intégration au DSL : La Synergie entre Codes Latents

Une fois l'autoencodeur entraîné, la partie **encodeur** E est utilisée pour transformer chaque vecteur d'entrée \mathbf{x}_i en un code latent $\mathbf{z}_i = E(\mathbf{x}_i)$ dans un espace de dimension réduite \mathbb{R}^k . Ce code latent remplace alors \mathbf{x}_i dans la phase de calcul de la synergie au sein du DSL. La mesure de synergie peut être définie par des formules classiques telles que

$$S(i, j) = \exp(-\alpha \parallel \mathbf{z}_i - \mathbf{z}_j \parallel^2)$$

ou par une mesure basée sur la similarité cosinus

$$S(i, j) = \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\parallel \mathbf{z}_i \parallel \parallel \mathbf{z}_j \parallel}.$$

Ces scores de synergie, calculés dans un espace de dimension réduite, facilitent le renforcement des pondérations $\omega_{i,j}$ et la formation de clusters dans le Synergistic Connection Network. L'optimisation de la dimension k joue alors un rôle critique : si k est trop petit, l'autoencodeur risque de perdre des informations essentielles, tandis qu'un k trop élevé n'apporte pas le gain de complexité recherché.

E. Limites et Points de Vigilance

Malgré leurs avantages, les autoencodeurs présentent quelques inconvénients notables. Premièrement, la minimisation de la **perte de reconstruction** n'est pas nécessairement alignée avec l'objectif de séparer distinctement les classes ou de maximiser la discrimination entre les entités. Par conséquent, les codes latents \mathbf{z}_i peuvent parfois ne pas être suffisamment discriminants pour la tâche d'auto-organisation.

Deuxièmement, l'entraînement d'un autoencodeur sur des datasets volumineux peut être coûteux en termes de ressources computationnelles et de temps, notamment lorsqu'il s'agit d'architectures profondes. Troisièmement, le choix de la dimension latente k est crucial : une valeur inappropriée peut soit sous-estimer les facteurs de variation essentiels, soit ne pas fournir une véritable réduction de complexité. Enfin, dans un contexte d'apprentissage continu, la mise à jour de l'encodeur E doit être soigneusement gérée pour éviter le phénomène de **catastrophic forgetting**, qui pourrait altérer l'organisation de l'espace latent déjà établi.

Conclusion

Les autoencodeurs constituent un pilier classique des méthodes sub-symboliques dans un DSL, permettant de réduire la dimensionnalité des données tout en extrayant des features discriminantes qui facilitent la formation de clusters cohérents au sein du Synergistic Connection Network. Leur architecture, fondée sur un encodeur et un décodeur, permet d'apprendre un code latent \mathbf{z} qui condense l'information essentielle d'un vecteur d'entrée \mathbf{x} . Cette réduction de dimension se traduit par un coût de calcul moindre lors de l'évaluation des similarités, ainsi qu'une meilleure capacité à éliminer le bruit. Néanmoins, il convient de rester vigilant quant aux limites inhérentes à la reconstruction, aux coûts d'entraînement, au choix de la dimension latente et aux défis de l'apprentissage continu. En intégrant judicieusement ces techniques, le DSL peut bénéficier d'une représentation sub-symbolique à la fois compacte, robuste et plus interprétable, améliorant ainsi la dynamique d'auto-organisation globale.

3.3.2.2. Transformers (BERT, GPT, ViT) : Embeddings Contextuels plus Riches

L'émergence des **Transformers** – tels que BERT, GPT, T5 en NLP et Vision Transformer (ViT) en vision – a profondément transformé la manière dont sont générés les **embeddings** dans divers domaines. Contrairement aux représentations plus statiques (comme Word2Vec) ou aux approches purement convolutionnelles (CNN), les Transformers produisent des **embeddings contextuels**.

Ces représentations intègrent non seulement l'information d'un token ou d'un patch de manière isolée, mais elles capturent également le **contexte global** par le biais d'un mécanisme de *self-attention*. Dans le cadre d'un **Deep Synergy Learning (DSL)**, cette capacité à incorporer le contexte se traduit par des mesures de synergie plus fines et expressives, où la proximité entre deux entités reflète non seulement des similarités de caractéristiques isolées, mais aussi des interactions complexes au sein de l'ensemble des données.

A. Principe des Transformers et Self-Attention

L'architecture Transformer repose sur un mécanisme de *self-attention* qui permet à chaque position d'une séquence de « regarder » toutes les autres positions. Soit une séquence d'entrées représentées par la matrice $\mathbf{x} \in \mathbb{R}^{n \times d}$, où chaque ligne correspond à un embedding initial d'un token ou d'un patch. Les Transformers calculent trois matrices essentielles par multiplication avec des poids appris :

$$\mathbf{Q} = \mathbf{x} W_Q, \quad \mathbf{K} = \mathbf{x} W_K, \quad \mathbf{V} = \mathbf{x} W_V,$$

où W_Q , W_K , et W_V sont des matrices de projection de dimensions appropriées (souvent $d \times d_k$). Le mécanisme de self-attention se définit alors par l'opération

$$\text{Att}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q} \mathbf{K}^T}{\sqrt{d_k}}\right) \mathbf{V}.$$

Cette opération permet à chaque token (ou patch) de pondérer les contributions de tous les autres éléments en fonction de la similitude entre ses **requêtes** et les **clés** associées aux autres tokens, produisant ainsi des embeddings qui intègrent l'information contextuelle de l'ensemble de la séquence.

B. BERT, GPT : Embeddings Contextuels en NLP

Dans le domaine du **Traitements Automatique du Langage Naturel (NLP)**, des modèles tels que **BERT** et **GPT** exploitent pleinement le mécanisme de self-attention pour produire des **embeddings contextuels**. Concrètement, après plusieurs couches de self-attention et de transformations non linéaires, chaque token i de la séquence obtient une représentation finale $\mathbf{h}_i^{(L)}$. Deux types d'extractions sont couramment utilisées :

- **Token-level** : Chaque mot ou sous-mot se voit attribuer un embedding \mathbf{h}_i qui capture ses nuances contextuelles.
- **Embedding global** : Un vecteur représentatif de l'ensemble de la séquence est généré, souvent en utilisant le token spécial $[CLS]$ (dans le cas de BERT) ou par agrégation (moyenne ou max) sur tous les tokens.

Dans un DSL, ces embeddings globaux servent de « signature » vectorielle pour une entité textuelle. La synergie entre deux entités \mathcal{E}_i et \mathcal{E}_j peut ainsi être évaluée par des mesures telles que la similarité cosinus :

$$S(i, j) = \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|},$$

où \mathbf{z}_i et \mathbf{z}_j sont les embeddings finaux issus du Transformer. Grâce à la prise en compte du contexte, ces représentations permettent de discerner des relations sémantiques fines, telles que la synonymie contextuelle ou les paraphrases, améliorant ainsi la qualité de l'auto-organisation du réseau.

C. Vision Transformer (ViT) : Patches et Self-Attention en Vision

Les principes du Transformer se sont également appliqués au domaine de la vision par ordinateur via le **Vision Transformer (ViT)**. Dans ce cas, une image est divisée en petits **patches** (par exemple, de 16×16 pixels), qui sont linéarisés et projetés dans un espace de dimension d . Pour conserver l'information spatiale, des **positional embeddings** sont ajoutés à chaque patch. Le Transformer traite alors l'ensemble des patches de manière similaire aux tokens en NLP, utilisant le mécanisme de self-attention pour capturer des relations globales dans l'image. La sortie finale peut être constituée d'un token spécial $[CLS]$ agissant comme représentation globale ou d'une agrégation des embeddings individuels. Cette approche permet de générer des embeddings d'images qui intègrent non seulement des caractéristiques locales, mais aussi des dépendances à longue portée entre différentes régions de l'image, améliorant ainsi la mesure de la synergie dans le DSL.

D. Enjeux et Coûts : Dimension et Ressources

Bien que les Transformers produisent des embeddings contextuels particulièrement riches, ils impliquent également des coûts importants en termes de **dimension** et de **ressources**. Par exemple, les modèles tels que BERT ou GPT génèrent des vecteurs de dimension 768, 1024 ou plus, ce qui peut amplifier la complexité des calculs de similarité dans un SCN, souvent de l'ordre de $O(n^2 \times d)$ pour n entités. De plus, le poids des modèles (parfois plusieurs centaines de millions de paramètres) requiert une infrastructure matérielle (GPU, TPU) conséquente pour l'inférence et le fine-tuning. Enfin, la sensibilité des Transformers aux ajustements fins – un léger changement de paramètres peut modifier la distribution des embeddings – nécessite des techniques d'alignement ou de versionnage pour préserver la stabilité des représentations dans un système d'apprentissage continu.

E. Intégration dans un DSL

L'intégration des Transformers dans un DSL se réalise en extrayant, pour chaque entité \mathcal{E}_i (qu'il s'agisse d'un document, d'une image ou d'un segment audio), un embedding contextuel \mathbf{z}_i via une opération telle que

$$\mathbf{z}_i = \text{TransformerEncode}(\mathbf{x}_i),$$

où \mathbf{x}_i représente l'objet brut initial et `TransformerEncode` désigne le processus de passage par plusieurs couches de self-attention et de feed-forward. Une fois obtenu, l'embedding peut être normalisé afin de faciliter le calcul de similarités, par exemple en utilisant le cosinus. La synergie entre deux entités est alors définie par une fonction f (comme le cosinus ou un noyau RBF) :

$$S(i, j) = f(\mathbf{z}_i, \mathbf{z}_j).$$

Dans le processus de mise à jour des pondérations du SCN,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

une synergie élevée entre des embeddings contextuels conduira au renforcement des liens, favorisant ainsi la formation de clusters d'entités sémantiquement cohérents, tout en capturant des relations complexes que seules des approches traditionnelles auraient du mal à discerner.

Conclusion

Les Transformers, à travers des modèles tels que BERT, GPT et ViT, offrent une nouvelle génération d'**embeddings contextuels** qui intègrent l'information globale par le mécanisme de self-attention. Ces représentations, bien que souvent de dimension élevée et exigeantes en ressources, permettent une mesure de la synergie plus fine et expressive dans un DSL, en tenant compte du contexte complet d'un token ou d'un patch. En intégrant ces embeddings dans le processus d'auto-organisation du Synergistic Connection Network, le DSL parvient à regrouper des entités en fonction de leur similarité contextuelle, améliorant ainsi la précision des clusters et la qualité globale du système. Malgré les défis liés aux coûts de calcul et à la stabilité des représentations, la richesse contextuelle offerte par les Transformers constitue un atout majeur pour des applications nécessitant une compréhension fine des interactions sémantiques dans des domaines variés.

3.3.2.3. Intérêt pour le DSL : un Embedding plus Robuste, mais plus Complexé à Calculer ou à Mettre à Jour

Dans le cadre d'un **Deep Synergy Learning (DSL)**, l'emploi de techniques avancées telles que les **Transformers** ou les **autoencodeurs profonds** permet d'obtenir des **embeddings** d'une qualité remarquable, caractérisés par une richesse contextuelle et une capacité de discrimination supérieure. Ces représentations, issues de modèles sophistiqués, se distinguent par leur aptitude à capter des nuances sémantiques fines et à structurer l'espace latent de manière à favoriser la formation de clusters d'entités plus robustes. On définit notamment, par une fonction g , l'embedding contextuel d'une entité initiale \mathbf{x}_i par

$$\mathbf{z}_i = g(\mathbf{x}_i),$$

ce qui permet de mesurer la **synergie** entre deux entités \mathcal{E}_i et \mathcal{E}_j à l'aide d'une distance ou d'un produit scalaire, par exemple

$$S(i, j) = \exp(-\alpha \|\mathbf{z}_i - \mathbf{z}_j\|^2) \quad \text{ou} \quad S(i, j) = \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|},$$

ce qui permet de regrouper les entités ayant des caractéristiques latentes similaires au sein du **Synergistic Connection Network (SCN)**. Ces méthodes confèrent au DSL une **robustesse** accrue dans la capture des variations contextuelles et une capacité à distinguer subtilement des différences sémantiques que des approches plus simples auraient tendance à négliger.

Cependant, la qualité supérieure des embeddings contextuels se traduit par une **complexité** notable tant dans le calcul de l'inférence que dans la gestion des mises à jour de ces représentations. Le modèle g , en intégrant plusieurs couches de self-attention et de transformations non linéaires, nécessite en général une infrastructure matérielle puissante, telle que des GPU ou TPU, en raison du nombre élevé de paramètres et de la dimension des vecteurs, souvent de l'ordre de 768, 1024 ou plus. Par conséquent, le coût de calcul d'une opération d'inférence, qui peut être évalué en

$O(n^2 \times d_{\text{latent}})$ pour un ensemble de n entités, augmente considérablement, en particulier lorsque le DSL doit traiter de grands volumes de données en temps réel. Cette complexité algorithmique se trouve également exacerbée par le coût associé à la mise à jour des embeddings dans un contexte d'apprentissage continu. En effet, si le modèle g subit un fine-tuning ou une révision partielle, il devient nécessaire de recalculer les embeddings pour toutes les entités, c'est-à-dire

$$\mathbf{z}_i(t+1) = g_{t+1}(\mathbf{x}_i) \quad \text{pour } i = 1, \dots, n,$$

ce qui, pour des ensembles de données de grande taille, peut engendrer une latence non négligeable et provoquer des fluctuations dans les valeurs de $S(i, j)$ entre les cycles de mise à jour. Ces variations, même modestes, peuvent induire des réorganisations brutales au sein du SCN, compromettant ainsi la stabilité de l'auto-organisation.

L'ingénierie d'un DSL doit donc établir un compromis entre la **richesse** des embeddings, qui améliore la précision des regroupements en capturant des informations contextuelles complexes, et l'**efficacité computationnelle** requise pour leur calcul et leur mise à jour régulière. Dans certains contextes critiques, où la qualité de la représentation est primordiale – par exemple en médecine ou en analyse de données sensibles – il peut être acceptable de supporter des coûts de calcul élevés pour garantir une **précision** et une **robustesse** accrues. À l'inverse, dans des applications massives où la rapidité d'inférence est essentielle, des modèles plus légers ou des embeddings de dimension réduite pourraient être privilégiés afin d'optimiser la scalabilité du système.

En somme, l'utilisation d'embeddings plus riches, issus de techniques avancées, permet d'améliorer significativement la **discrimination** et la **robustesse** des synergies dans un DSL, mais se fait au prix d'une complexité computationnelle accrue et d'un défi supplémentaire en termes de mise à jour des représentations dans un cadre d'apprentissage continu. Ce compromis entre la richesse des représentations et l'efficacité opérationnelle constitue un enjeu central dans la conception et l'implémentation de systèmes de Deep Synergy Learning.

3.3.3. Calcul de la Synergie entre Vecteurs

Dans les sections précédentes (3.3.1 et 3.3.2), nous avons mis en évidence l'usage de **vecteurs** (embeddings) pour représenter des entités dans un contexte sub-symbolique, ainsi que l'apport de méthodes avancées (autoencodeurs, Transformers) pour produire des embeddings plus riches. Il reste à voir **comment**, à partir de ces vecteurs, on définit la **synergie** $S(i, j)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j . En pratique, le DSL (Deep Synergy Learning) recourt souvent à des **fonctions** de distance ou de similarité géométriques. Cette section (3.3.3) se divise en trois points :

- (3.3.3.1) présentation des **distances usuelles** (euclidienne, manhattan) et de la **similarité cosinus**,
- (3.3.3.2) usage de **kernels** pour une similarité non linéaire,
- (3.3.3.3) **ajustements** (normalisation, calibration) pour mieux gérer le bruit ou l'échelle des vecteurs.

3.3.3.1. Distances Usuelles : Euclidienne, Manhattan, Cosinus

Dans le cadre du **Deep Synergy Learning (DSL)**, chaque entité \mathcal{E}_i est représentée par un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$. L'évaluation de la **synergie** entre deux entités repose alors sur la comparaison de ces représentations via une fonction de **distance** ou de **similarité**. Ces mesures jouent un rôle fondamental dans le processus d'**auto-organisation** du **Synergistic Connection Network (SCN)**, puisque plus deux vecteurs sont « proches » dans l'espace, plus la **pondération** $\omega_{i,j}$ tend à être renforcée. Parmi les approches classiques pour mesurer cette proximité, on trouve la **distance euclidienne**, la **distance manhattan** et la **similarité cosinus**, chacune offrant une interprétation géométrique différente et répondant à des exigences spécifiques selon le type de données traitées.

A. Distance Euclidienne

La **distance euclidienne** constitue la mesure la plus intuitive, puisqu'elle correspond à la longueur du segment reliant les points représentés par \mathbf{x}_i et \mathbf{x}_j dans \mathbb{R}^d . Mathématiquement, elle se définit par

$$d_{\text{eucl}}(i, j) = \| \mathbf{x}_i - \mathbf{x}_j \| = \sqrt{\sum_{k=1}^d (x_{i,k} - x_{j,k})^2}.$$

La valeur ainsi obtenue représente une notion de **distance** physique dans l'espace, et dans le DSL, cette distance est souvent convertie en un **score de similarité** afin de faciliter son intégration dans la dynamique du SCN. Une fonction d'agrégation classique consiste à appliquer une décroissance exponentielle, ce qui donne

$$\mathbf{S}(i, j) = \exp(-\alpha \| \mathbf{x}_i - \mathbf{x}_j \|^2),$$

où $\alpha > 0$ est un **paramètre** qui ajuste la pente de décroissance. Dans ce contexte, une distance faible entre \mathbf{x}_i et \mathbf{x}_j conduit à un score proche de 1, indiquant une **affinité élevée** entre les deux entités et justifiant ainsi le renforcement de la pondération $\omega_{i,j}$.

B. Distance Manhattan (ou ℓ_1)

La **distance manhattan** ou **distance taxicab** s'exprime en prenant la somme des différences absolues entre les composantes correspondantes des vecteurs. La formule est donnée par

$$d_{\text{manh}}(i, j) = \sum_{k=1}^d |x_{i,k} - x_{j,k}|.$$

Cette mesure, qui évalue la différence cumulée par dimension, est particulièrement utile lorsque les données se combinent de façon additive, et elle présente l'avantage de se montrer souvent plus **robuste** face aux valeurs extrêmes de certaines coordonnées. Pour intégrer cette distance dans un DSL, on peut transformer la mesure en un score de similarité en appliquant par exemple la fonction exponentielle

$$\mathbf{S}(i, j) = \exp(-\beta d_{\text{manh}}(i, j)),$$

ou encore en utilisant la forme rationnelle

$$\mathbf{S}(i, j) = \frac{1}{1 + d_{\text{manh}}(i, j)},$$

où $\beta > 0$ est un **paramètre** qui ajuste le taux de décroissance. De cette manière, une distance manhattan réduite se traduit par un score de similarité élevé, indiquant une forte **affinité** entre les entités.

C. Similarité Cosinus

La **similarité cosinus** se distingue des mesures de distance classiques en se focalisant sur l'**angle** entre deux vecteurs plutôt que sur leur distance absolue. Elle est définie par

$$\mathbf{S}_{\cos}(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

et prend des valeurs dans l'intervalle $[-1, 1]$. Cette mesure atteint la valeur 1 lorsque les vecteurs sont colinéaires dans la même direction, -1 lorsqu'ils sont opposés et 0 lorsque les vecteurs sont orthogonaux. Dans le cadre du DSL, il est souvent souhaitable d'obtenir un score strictement positif, et pour ce faire, on peut transformer l'intervalle en appliquant par exemple la formule

$$\mathbf{S}(i, j) = \frac{1 + \mathbf{S}_{\cos}(i, j)}{2},$$

qui convertit les valeurs de $[-1, 1]$ en $[0, 1]$. Cette méthode permet de neutraliser l'effet de la norme des vecteurs, en se concentrant uniquement sur leur **direction**, ce qui est particulièrement utile pour des données textuelles ou pour des représentations où la magnitude peut varier de manière significative.

D. Liaison avec la Synergie dans le DSL

L'ensemble des mesures présentées permet de définir la **synergie** entre deux entités, laquelle est utilisée pour ajuster la **pondération** $\omega_{i,j}$ dans le SCN. La règle de mise à jour se formalise par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}(i, j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le **taux d'apprentissage** et $\tau > 0$ représente le **coefficent de décroissance**. Ce mécanisme d'auto-organisation implique que plus deux entités sont évaluées comme étant proches (ou similaires) selon la mesure choisie, plus la pondération $\omega_{i,j}$ tend à augmenter, favorisant ainsi la formation de **clusters** cohérents. Le choix entre la **distance euclidienne**, la **distance manhattan** ou la **similarité cosinus** dépend du type de données et des propriétés spécifiques recherchées, car chacune offre des avantages distincts. La distance euclidienne fournit une mesure classique de la proximité, la distance manhattan peut mieux gérer certaines disparités dimensionnelles et la similarité cosinus permet de se focaliser sur l'orientation des vecteurs indépendamment de leur norme.

Conclusion

La formulation mathématique des **distances usuelles** joue un rôle central dans la définition de la **synergie** au sein du DSL. En utilisant des formules telles que

$$d_{\text{eucl}}(i, j) = \sqrt{\sum_{k=1}^d (x_{i,k} - x_{j,k})^2}, \quad d_{\text{manh}}(i, j) = \sum_{k=1}^d |x_{i,k} - x_{j,k}|,$$

et

$$\mathbf{S}_{\cos}(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

puis en convertissant ces distances en scores de similarité par des fonctions exponentielles ou des transformations linéaires, le DSL parvient à quantifier de manière rigoureuse la **proximité** entre entités. Ces scores, intégrés dans la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}(i, j) - \tau \omega_{i,j}(t)],$$

permettent de renforcer les liens entre entités proches dans l'espace vectoriel et d'aboutir à une structure auto-organisée cohérente. Le choix de la mesure de distance ou de similarité dépendra du contexte d'application et des caractéristiques spécifiques des données traitées, ce qui confère au DSL une flexibilité essentielle dans l'exploitation de la **géométrie** de \mathbb{R}^d .

3.3.3.2. Kernels (RBF, Polynomial) pour une Similarité non Linéaire

Les **distances** usuelles (euclidienne, manhattan) ou la **similarité** cosinus (section 3.3.3.1) considèrent des relations souvent **linéaires** ou du moins directes dans l'espace vectoriel. Cependant, il arrive que la **relation** réelle entre deux entités \mathcal{E}_i et \mathcal{E}_j soit plus subtile, nécessitant des méthodes **non linéaires** pour en rendre compte. Dans un **Deep Synergy Learning (DSL)**, recourir à des **kernels** (noyaux) permet d'introduire des notions de **similarité** bien plus riches que les simples distances ℓ_p . Les **RBF kernels** (noyaux gaussiens) et **kernels polynomiaux** figurent parmi les plus répandus et se traduisent en un **score** $\mathbf{S}(i, j)$ apte à révéler des liens complexes cachés dans l'espace initial \mathbb{R}^d .

A. Principe Général des Kernels

Un **kernel** k entre deux vecteurs $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ se définit comme une **fonction** :

$$k: \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R},$$

renvoyant un score (positif ou non) quantifiant leur **affinité**. La **particularité** du kernel trick, issu de la théorie des **noyaux de Mercer**, réside dans le fait que $k(\mathbf{x}_i, \mathbf{x}_j)$ peut être interprété comme un **produit scalaire** dans un espace de dimension éventuellement infinie :

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}},$$

où $\phi: \mathbb{R}^d \rightarrow \mathcal{H}$ est une **transformation** non linéaire, et \mathcal{H} un **espace** (souvent de Hilbert) de dimension potentiellement énorme. La **magie** du kernel trick est qu'on n'a pas besoin de **calculer** explicitement ϕ . Il suffit d'évaluer $k(\mathbf{x}_i, \mathbf{x}_j)$. D'un point de vue **DSL**, cela équivaut à définir la **synergie** $S(i, j)$:

$$S(i, j) = k(\mathbf{x}_i, \mathbf{x}_j).$$

Deux entités peuvent se trouver “lointaines” selon une métrique linéaire, tout en étant **fortement similaires** en termes de transformations non linéaires. Dans un **Synergistic Connection Network**, on peut alors capter des formes de **ressemblance** plus complexes, sans explicitement projeter les entités dans un espace étendu.

B. RBF Kernel (Radial Basis Function)

Le **kernel RBF** (aussi appelé “noyau gaussien”) constitue un choix très répandu. Il s’écrit :

$$k_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad (\gamma > 0).$$

Il s’agit d’une **exponentielle négative** de la distance euclidienne au carré. On peut l’interpréter comme une “**gaussienne**” centrée sur \mathbf{x}_j . Plus \mathbf{x}_i est proche de \mathbf{x}_j , plus la valeur se rapproche de 1 ; plus ils sont éloignés, plus elle tend vers 0. Le **paramètre** γ détermine la “largeur” du noyau :

- Un γ **grand** induit une décroissance rapide de $k_{\text{RBF}}(\mathbf{x}_i, \mathbf{x}_j)$ dès que $\|\mathbf{x}_i - \mathbf{x}_j\|$ grandit, ce qui focalise la similarité sur une zone très **locale**.
- Un γ **petit** rend la fonction plus “plate”, donnant une vue plus **globale** où des entités modérément éloignées conservent une similarité notable.

Pour un **DSL**, un **RBF kernel** donne une **synergie** :

$$S(i, j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2).$$

Cette forme **non linéaire** permet à des entités ayant un écart euclidien significatif (dans \mathbb{R}^d) de tout de même afficher une similarité mesurable si elles partagent certaines **propriétés** sous-jacentes. En outre, si γ est bien paramétré, la synergie reflète de subtiles variations dans les données — plus subtiles, parfois, qu’une simple distance euclidienne.

C. Polynomial Kernel

Un autre kernel usuel est le **kernel polynomial** :

$$k_{\text{poly}}(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i \cdot \mathbf{x}_j + c)^p,$$

où p est le **degré** (typiquement 2, 3, 4...) et $c \geq 0$ un terme de décalage. Cette définition incorpore toutes les **composantes** polynomiales du produit $\mathbf{x}_i \cdot \mathbf{x}_j$, allant jusqu’au degré p . Ainsi, pour $p = 2$, on inclut toutes les interactions bilinéaires $(x_{i,k}x_{j,\ell})$. Cela peut capturer des **corrélations** plus complexes qu’un simple “angle” ou qu’une distance linéaire.

Dans un **SCN**, si on adopte $S(i, j) = k_{\text{poly}}(\mathbf{x}_i, \mathbf{x}_j)$, on se retrouve avec une **synergie** qui peut s’avérer très sensible à des variations dans certaines dimensions, surtout pour un p élevé. Cela peut amplifier des **propriétés** souhaitées (s’il y a de fortes corrélations polynomiales), mais risque aussi de **surexposer** le bruit ou les valeurs extrêmes. Le choix de p et c réclame donc un réglage attentif.

D. Non-Linéarité et Espace Implicit : Kernel Trick

Le **kernel trick** signifie que l'on n'a **pas** besoin de définir explicitement une transformation $\phi(\cdot)$ vers un espace de dimension (parfois énorme) \mathcal{H} . Il suffit de calculer $k(\mathbf{x}_i, \mathbf{x}_j)$. Dans un **DSL**, cela se traduit par une **synergie** :

$$S(i, j) = k(\mathbf{x}_i, \mathbf{x}_j) \quad \text{où } k \text{ est RBF, polynomial, ou autre.}$$

Mathématiquement, deux vecteurs lointains en \mathbb{R}^d peuvent devenir "proches" dans l'espace \mathcal{H} , si la transformation ϕ induite par le kernel met en évidence des attributs partagés. Ainsi, la **mise à jour** $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$ tient compte de propriétés non linéaires. Dans la **formation** de clusters, des entités qui ne semblaient pas se ressembler au sens euclidien peuvent se rapprocher si leur **synergie** kernel s'avère élevée.

E. Exemples d'Implémentation dans un SCN

Pour un **SCN** gérant n entités, on définit $S(i, j) = k(\mathbf{x}_i, \mathbf{x}_j)$. À chaque itération, calculer $\omega_{i,j}(t+1)$ exige d'évaluer ces kernels. Sur le plan **coût** :

- 155. **RBF** : $O(d)$ multiplications pour $\|\mathbf{x}_i - \mathbf{x}_j\|^2$, puis une exponentiation.
- 156. **Polynomial** : $O(d)$ pour le produit scalaire $\mathbf{x}_i \cdot \mathbf{x}_j$, puis l'élévation au degré p .

Pour comparer toutes les paires (i, j) , on fait face à $O(n^2)$ évaluations, chaque en $O(d)$. Si n est grand, cela devient lourd (voir chap. 3.2.2.2). En pratique, on peut élaguer en ne calculant la synergie que sur un **voisinage** restreint (approximate nearest neighbor, etc.) ou recourir à d'autres heuristiques d'échantillonnage pour gérer la **scalabilité**.

F. Limites et Paramétrage Sensible

Le **RBF kernel** nécessite le choix d'un paramètre γ . Un γ trop grand produit une décroissance exponentielle ultra-rapide, rendant la synergie presque nulle pour la majorité des paires, ce qui peut isoler excessivement les entités. Un γ trop petit, en revanche, aboutit à un noyau très plat, de sorte que presque tout le monde se retrouve "similaire" : on aboutit à un **mégacluster**. Le **polynomial kernel** demande aussi de fixer p et c . Un p trop élevé peut amplifier les bruits, alors qu'un p trop faible (ex. 1) redescend à un comportement linéaire classique.

L'ingénieur d'un **DSL** doit donc **régler** ces hyperparamètres en fonction des données, par validation croisée ou d'autres heuristiques. Les kernels offrent une flexibilité précieuse, mais nécessitent une **phase** d'ajustement pour qu'ils produisent des **synergies** stables et cohérentes dans le SCN.

3.3.3.3. Ajustements : Normalisation, Calibration, Prise en Compte du Bruit

Dans un **Deep Synergy Learning (DSL)** où les entités \mathcal{E}_i sont décrivées par des **vecteurs** ou des attributs numériques, l'évaluation de la **synergie** $S(i, j)$ repose sur des **distances** (euclidienne, manhattan) ou des **similarités** (cosinus, kernels). Toutefois, dans la **pratique**, il ne suffit pas de choisir une métrique : il faut aussi **calibrer** et **ajuster** la façon dont on compare les vecteurs. Les

questions de normalisation, d'échelle, de saturation, ou encore de **prise en compte** du bruit sont essentielles pour garantir la **cohérence** du calcul de $S(i, j)$ et la stabilité de la mise à jour $\omega_{i,j}$. Les sections qui suivent décrivent en détail comment on gère ces ajustements pour un **Synergistic Connection Network (SCN)** robuste et efficace.

A. Normalisation et Équilibrage des Vecteurs

Lorsqu'on manipule des vecteurs $\mathbf{x}_i \in \mathbb{R}^d$ pour calculer la synergie $S(i, j)$, il arrive souvent que certaines **dimensions** varient sur des plages beaucoup plus larges que d'autres, ou que différentes entités \mathcal{E}_i n'utilisent pas le même **ordonnancement** de valeurs. Sans précautions, la mesure de distance ou de similarité peut s'en trouver **biaisée**. Deux mécanismes de base s'appliquent :

157. Normalisation en norme

Un classique consiste à normaliser chaque vecteur à la **norme 1**, c'est-à-dire :

$$\mathbf{x}'_i = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}, \quad \|\mathbf{x}'_i\| = 1.$$

Ainsi, la **similarité cosinus** entre \mathbf{x}'_i et \mathbf{x}'_j devient un **produit scalaire** direct. Cela annule l'influence de la **magnitude** brute et met l'accent sur l'**angle**. Dans un DSL multimodal, cette démarche évite qu'une modalité sortant des valeurs très élevées ne domine la mesure de proximité.

158. Normalisation par composante

On peut aussi appliquer un **z-score** (soustraction de la moyenne et division par l'écart-type), ou un min–max scaling dans l'intervalle [0,1], par dimension. Cette approche assure que chaque coordonnée contribue équitablement. Dans un **SCN**, elle prévient la situation où une composante “géante” écrase toutes les autres dans le calcul de $\|\mathbf{x}_i - \mathbf{x}_j\|$ ou $\mathbf{x}_i \cdot \mathbf{x}_j$.

En pratique, la **choix** entre ces normalisations dépend de la distribution des données. L'objectif reste de **préserver** le signal discriminant et de **limiter** l'influence de différences d'échelle, améliorant ainsi la comparabilité et la **qualité** du calcul de $S(i, j)$.

B. Calibration du Score de Synergie

Une fois les vecteurs normalisés, on définit la synergie par :

$$S(i, j) = \phi(d(\mathbf{x}_i, \mathbf{x}_j)) \quad \text{ou} \quad \psi(\text{sim}(\mathbf{x}_i, \mathbf{x}_j)),$$

où ϕ et ψ sont des transformations (éventuellement exponentielles, sigmoïdes, inverses, etc.). Il se peut qu'on désire **calibrer** la sortie pour moduler l'amplitude. Par exemple :

159. Noyau gaussien

$$S(i, j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|^2),$$

où $\alpha > 0$ gouverne la rapidité de décroissance. Si α est trop grand, la synergie chute quasi immédiatement pour tout écart modéré, regroupant uniquement les entités quasi identiques. Si α est trop petit, elle reste trop élevée pour un large spectre de distances, fusionnant potentiellement tout en un seul méga-cluster.

160. Distance inversée

$$S(i,j) = \frac{1}{1 + \beta d(\mathbf{x}_i, \mathbf{x}_j)},$$

où $\beta > 0$. Ici, plus la distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ grandit, plus S tend vers 0 mais jamais exactement 0. Le réglage de β dicte le “taux” d’atténuation.

161. Troncature ou saturation

On peut saturer la sortie pour ne pas dépasser 1 (ou un certain seuil), ou imposer un plancher minimal. Ainsi, la synergie reste dans $[\varepsilon, 1 - \varepsilon]$ afin d’éviter un renforcement ou un affaiblissement trop extrême. Ceci peut stabiliser la mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Le point crucial est de **calibrer** S de façon à refléter *juste assez* la disparité entre entités, sans aboutir à des valeurs saturées pour la plupart des paires (synergie trop extrême) ou trop faibles (peu de différenciation).

C. Prise en Compte du Bruit

Dans bien des environnements (capteurs, images à faible qualité, textes “sales” ou hétérogènes), les **données** sont bruitées. Ce bruit peut fausser le calcul de distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ ou de similarité $\mathbf{x}_i \cdot \mathbf{x}_j$. Plusieurs stratégies se dégagent :

162. Filtrage ou prétraitement

Avant tout calcul de distance, on peut nettoyer ou lisser \mathbf{x}_i , par exemple via du **smoothing** ou un **autoencodeur denoising**. Ce faisant, on réduit la variabilité de bas niveau, améliorant la robustesse de la synergie.

163. Troncature ou clipping des valeurs extrêmes

Si certaines composantes subissent des pics aberrants, on peut fixer un **cap** sur l’amplitude. Cela évite qu’une seule dimension bruitée ne domine la distance.

164. Injection de bruit contrôlé

Parfois, on ajoute soi-même un bruit gaussien au vecteur \mathbf{x}_i (ou pendant l’apprentissage) pour **stabiliser** l’auto-organisation, un peu à la manière d’un recuit simulé. Cette démarche évite de se figer dans des minima locaux, et autorise une **exploration** plus large du SCN. Au fil du temps, on diminue ce bruit pour **affiner** la convergence.

165. Robustesse des fonctions de distance

Au lieu de la distance euclidienne ℓ_2 , on peut employer ℓ_1 (manhattan), réputée moins sensible aux outliers sur une coordonnée. On peut également introduire des fonctions qui saturent au-delà d’une certaine différence, réduisant l’influence des valeurs extrêmes.

3.3.4. Gestion du Bruit et de l'Évolution des Embeddings

Même si les vecteurs et embeddings (section 3.3.3) constituent une base pratique pour représenter les entités dans un DSL (Deep Synergy Learning), ils ne sont pas figés. Dans la **réalité** d'un système en évolution (capteurs changeants, nouvelles données, amélioration continue du modèle), il faut prendre en compte :

- 166. Le **bruit** potentiel dans les données ou dans les embeddings, qui peut entraîner des distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ artificiellement élevées ou basses,
- 167. La **mise à jour** ou le re-entraînement des modèles produisant les embeddings (réseaux neuronaux, Transformers, etc.), qui peut modifier la représentation \mathbf{x}_i au fil du temps,
- 168. Les **stratégies** de filtrage, de clipping, d'ajustement local pour éviter la propagation d'erreurs ou la dérive progressive.

Ainsi, dans cette section, nous verrons comment le DSL peut gérer ces aspects (bruit, évolution) et préserver la **cohérence** du calcul de synergie $S(i, j)$.

3.3.4.1. Embeddings Potentiellement Réentraînés, Fine-Tuned (Chap. 9 sur Apprentissage Continu)

Dans le cadre du **Deep Synergy Learning (DSL)**, les **embeddings** \mathbf{x}_i qui caractérisent les entités \mathcal{E}_i ne sont pas considérés comme des représentations statiques et définitives. En effet, ces vecteurs, générés par des modèles neuronaux tels que des **CNN**, des **Transformers** ou des **autoencodeurs**, peuvent évoluer au fil du temps lorsque le modèle générateur est soumis à un processus de **fine-tuning** ou à un schéma d'**apprentissage continu**. Ce phénomène de mise à jour dynamique des embeddings influence directement la **synergie** $S(i, j)$ entre entités ainsi que la manière dont les **pondérations** $\omega_{i,j}$ sont ajustées dans le **Synergistic Connection Network (SCN)**.

A. Fine-Tuning et Mise à Jour des Embeddings

Considérons qu'à un instant t , chaque entité \mathcal{E}_i est décrite par un **embedding** $\mathbf{x}_i(t)$ obtenu via un modèle g_t , qui s'exprime par la relation

$$\mathbf{x}_i(t) = g_t(\mathbf{d}_i),$$

où \mathbf{d}_i représente les données brutes associées à l'entité, telles qu'une image, un texte ou un signal audio. Lorsqu'un processus de **fine-tuning** est initié – c'est-à-dire que le modèle g_t est réentraîné ou ajusté en réponse à de nouvelles données ou à un objectif révisé – le modèle évolue vers une nouvelle version g_{t+1} et l'embedding correspondant se met à jour selon

$$\mathbf{x}_i(t + 1) = g_{t+1}(\mathbf{d}_i).$$

Même une modification modeste des poids du modèle peut entraîner une transformation significative de l'espace vectoriel, modifiant ainsi les distances $\|\mathbf{x}_i(t) - \mathbf{x}_j(t)\|$ entre entités et, par conséquent, la valeur de la **synergie** $S(i, j)$. Ce réajustement induit une réorganisation du SCN, puisque la règle de mise à jour des pondérations

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j,t) - \tau \omega_{i,j}(t)],$$

dépend directement des représentations actuelles $\mathbf{x}_i(t)$ et $\mathbf{x}_j(t)$. Une telle dynamique impose une attention particulière dans la gestion des mises à jour pour ne pas perturber brusquement la structure des clusters déjà établis.

B. Apprentissage Continu et Flux Dynamique

Dans un scénario d'**apprentissage continu**, le modèle g_t est régulièrement mis à jour pour intégrer un nouveau flux de données, noté $\Delta\mathcal{D}$. La relation de mise à jour du modèle s'exprime alors par

$$g_{t+1} = \text{Train}(g_t, \Delta\mathcal{D}),$$

ce qui conduit à une évolution progressive des embeddings selon

$$\mathbf{x}_i(t+1) = g_{t+1}(\mathbf{d}_i).$$

Ce mécanisme est particulièrement précieux dans des environnements non stationnaires où de nouvelles classes ou de nouvelles variations de données apparaissent de manière régulière. Toutefois, cette flexibilité s'accompagne d'une instabilité potentielle, puisque la transformation du modèle peut modifier de façon substantielle la géométrie de l'espace latent. Les distances entre les embeddings – et donc les scores de synergie – peuvent varier brutalement, induisant des réorganisations dans le SCN qui se traduisent par des fluctuations des pondérations $\omega_{i,j}$.

C. Impact sur la Mise à Jour des Pondérations

La dynamique d'auto-organisation du SCN est directement affectée par l'évolution des embeddings. La règle de mise à jour des pondérations

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j,t) - \tau \omega_{i,j}(t)]$$

utilise la **synergie** $S(i,j,t)$, qui dépend des embeddings à l'instant t . Si, suite à un fine-tuning ou à un apprentissage continu, les embeddings se modifient de manière significative, la valeur de $S(i,j,t+1)$ peut être très différente de celle précédemment obtenue. Ces variations peuvent provoquer des **sauts** ou des oscillations dans les pondérations, affectant la stabilité du regroupement des entités dans le réseau. Pour limiter ce phénomène, il est souvent recommandé d'adopter des stratégies de **lissage** dans la transition entre $\mathbf{x}_i(t)$ et $\mathbf{x}_i(t+1)$, par exemple en définissant une mise à jour pondérée telle que

$$\mathbf{x}_i(t+1) \leftarrow \alpha \mathbf{x}_i(t) + (1 - \alpha) g_{t+1}(\mathbf{d}_i),$$

où le paramètre $\alpha \in [0,1]$ contrôle l'ampleur de la transition, permettant ainsi d'introduire progressivement les changements et de préserver la cohérence de la dynamique du SCN.

D. Équilibre entre Adaptation et Stabilité

L'adaptabilité des embeddings grâce au fine-tuning et à l'apprentissage continu offre une capacité essentielle à un DSL, puisqu'elle permet au système de rester pertinent face à l'évolution des données. Toutefois, cette adaptabilité se heurte à la nécessité de maintenir une stabilité dans l'organisation des entités. Un modèle qui se modifie trop fréquemment risque de déstabiliser les **clusters** existants, car les pondérations $\omega_{i,j}$ basées sur d'anciennes représentations ne seront plus

en adéquation avec la nouvelle configuration de l'espace latent. Un compromis se trouve souvent dans la planification de mises à jour épisodiques, suivies de périodes de stabilisation, ou dans l'application de techniques d'adaptation incrémentale qui permettent un ajustement progressif. L'ingénieur se doit de régler la fréquence et l'ampleur des mises à jour de g_t afin de concilier la **richesse contextuelle** des nouvelles représentations et la nécessité d'une **stabilité** du SCN.

E. Gestion Concrète : Références au Chapitre 9 sur l'Apprentissage Continu

Le **chapitre 9** du présent ouvrage se penche en profondeur sur les techniques d'**apprentissage continu** et les stratégies permettant de gérer les mises à jour des embeddings sans compromettre la cohérence globale du DSL. Il y est présenté des protocoles d'adaptation incrémentale, des heuristiques de recalibrage des pondérations, ainsi que des mécanismes visant à atténuer le phénomène de **catastrophic forgetting**. Ces approches incluent notamment l'actualisation progressive des représentations, le verrouillage temporaire de certaines couches du modèle, et l'indexation régulière des embeddings afin d'assurer que la transition entre différentes versions du modèle reste la plus fluide possible.

Conclusion

L'évolution des **embeddings** dans un DSL, que ce soit par fine-tuning ou par apprentissage continu, constitue une dimension temporelle essentielle qui confère au système une **adaptabilité** remarquable face à des données non stationnaires. Toutefois, cette dynamique introduit des défis non négligeables quant à la stabilité de la synergie $S(i, j)$ et à la mise à jour des **pondérations** $\omega_{i,j}$ dans le SCN. Les fluctuations induites par des modifications du modèle générateur g_t peuvent perturber la structure auto-organisée, d'où l'importance de stratégies de lissage, de verrouillage ou de recalibrage, comme le décrit le **chapitre 9**. En trouvant un équilibre judicieux entre **adaptation** et **stabilité**, le DSL peut ainsi exploiter pleinement la richesse contextuelle des nouvelles représentations tout en assurant la continuité et la robustesse de l'auto-organisation.

3.3.4.2. Filtrage Local (k-NN) ou Clipping pour Limiter la Propagation d'Erreurs

Dans le cadre d'un **Deep Synergy Learning (DSL)**, la qualité et la stabilité de la **synergie** entre entités reposent sur la précision des **embeddings** et des **scores de similarité** calculés à partir de ceux-ci. Cependant, en présence de **perturbations** telles que le bruit, des valeurs aberrantes ou des variations imprévues dans les données, il devient indispensable de mettre en place des **mécanismes** visant à limiter la propagation des erreurs qui pourraient déstabiliser l'ensemble du **Synergistic Connection Network (SCN)**. Deux approches complémentaires se distinguent dans cette optique : le **filtrage local** par l'algorithme des **k-plus-proches-voisins (k-NN)** et le **clipping** des vecteurs ou des scores de similarité. Ces stratégies, en restreignant l'influence d'entités extrêmes ou aberrantes, contribuent à préserver la stabilité globale du système.

A. Filtrage Local via k-Plus-Proches-Voisins (k-NN)

Lorsqu'une entité \mathcal{E}_i est caractérisée par un **embedding** $\mathbf{x}_i \in \mathbb{R}^d$, il est fréquent, dans un SCN, de calculer la **synergie** $S(i, j)$ entre toutes les paires (i, j) . Cette approche, qui aboutit à un graphe complet, peut introduire des liaisons peu informatives ou même trompeuses, notamment lorsque des perturbations locales affectent certains embeddings. Afin de remédier à ce problème, il est judicieux de restreindre le calcul de la synergie à un **voisinage** immédiat, en considérant

uniquement les k entités les plus proches de \mathbf{x}_i . Formellement, le voisinage local d'une entité se définit par

$$\text{NN}_k(i) = \{j \in \{1, \dots, n\} \mid j \text{ fait partie des } k \text{ plus proches entités de } \mathbf{x}_i\}.$$

En conséquence, la fonction de synergie est tronquée de manière à ce que

$$S'(i, j) = \begin{cases} S(i, j), & \text{si } j \in \text{NN}_k(i) \text{ ou } i \in \text{NN}_k(j), \\ 0, & \text{sinon,} \end{cases}$$

ce qui signifie que seules les entités jugées suffisamment proches contribuent à la mise à jour des **pondérations** $\omega_{i,j}$. L'**avantage** de cette approche réside dans la réduction de la complexité, passant d'un nombre de comparaisons de l'ordre de $O(n^2)$ à $O(n k)$, tout en **limitant** l'impact des **outliers** qui, étant isolés, ne figurent pas dans le voisinage local. En outre, cette stratégie contribue à renforcer la **robustesse** du DSL en ne construisant des liaisons significatives qu'entre des entités réellement compatibles dans l'espace \mathbb{R}^d .

B. Clipping des Vecteurs et des Scores de Similarité

En complément du filtrage local, le **clipping** constitue une méthode efficace pour restreindre l'influence d'**valeurs extrêmes**. Cette technique peut s'appliquer à différents niveaux du processus d'inférence. Lorsqu'il s'agit des **embeddings** eux-mêmes, il est souvent pertinent de contraindre chaque coordonnée du vecteur \mathbf{x}_i afin d'éviter que des perturbations locales ne provoquent une domination de certaines dimensions. Pour ce faire, on impose un seuil $T > 0$ et on définit le vecteur « clippé » par

$$x_{i,k}^{\text{clip}} = \min\{\max\{x_{i,k}, -T\}, T\}.$$

Cette opération garantit que les valeurs de chaque dimension restent contenues dans l'intervalle $[-T, T]$, ce qui permet de préserver une **stabilité** géométrique dans \mathbb{R}^d . Par ailleurs, il est également possible d'appliquer le clipping directement au **score de synergie**. Si, par exemple, la fonction $S(i, j)$ calcule une similarité qui pourrait parfois dépasser une borne raisonnable à cause d'un outlier, on peut définir une version clipée de cette synergie par

$$S'(i, j) = \min\{S(i, j), S_{\max}\},$$

où S_{\max} est une borne supérieure prédéfinie. Cette approche empêche que la **synergie** n'atteigne des valeurs trop élevées qui pourraient induire un renforcement disproportionné des pondérations $\omega_{i,j}$, préservant ainsi l'équilibre global du SCN.

C. Combinaison des Approches pour une Robustesse Accrue

Dans la pratique, l'application conjointe du **filtrage local** par k-NN et du **clipping** constitue une stratégie robuste pour limiter la propagation d'erreurs. En effet, le filtrage local restreint d'abord le calcul de la synergie aux voisins les plus proches, réduisant la densité du graphe et la probabilité d'étendre l'influence d'entités aberrantes. Par la suite, le clipping, appliqué sur les vecteurs ou directement sur les scores de similarité, permet de modérer l'impact de toute valeur extrême qui pourrait encore se glisser dans le processus. Le SCN met ainsi à jour les pondérations selon la formule

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S'(i,j) - \tau \omega_{i,j}(t)],$$

où $S'(i,j)$ représente la synergie obtenue après application des mécanismes de filtrage et de clipping. Cette approche intégrée permet de réduire la complexité computationnelle, en limitant le nombre de comparaisons nécessaires, tout en préservant la stabilité de l'auto-organisation en atténuant l'impact des perturbations locales.

Conclusion

L'implémentation de techniques telles que le **filtrage local** par l'algorithme des **k-plus-proches-voisins** et le **clipping** des valeurs constitue un élément crucial dans le maintien de la **stabilité** du DSL. En restreignant le calcul de la synergie aux voisinages pertinents et en limitant l'influence des outliers, ces mécanismes permettent de préserver une dynamique d'auto-organisation robuste et efficace au sein du **Synergistic Connection Network (SCN)**. La combinaison de ces méthodes permet d'obtenir un système plus clairsemé et moins sujet aux fluctuations induites par des perturbations locales, garantissant ainsi une évolution cohérente des pondérations $\omega_{i,j}$. Ces stratégies sont essentielles pour le déploiement de DSL dans des environnements réels, où la qualité des données est variable et où la stabilité du modèle est primordiale pour assurer des performances optimales.

3.3.4.3. Exemples de Scénarios Audio-Visuels

Dans le cadre du **Deep Synergy Learning (DSL)**, les **applications audio-visuelles** offrent un terrain d'expérimentation particulièrement riche pour illustrer la capacité du système à **auto-organiser** des entités issues de modalités diverses telles que l'image, le son et le texte. En effet, le **Synergistic Connection Network (SCN)** se doit de prendre en compte des **embeddings** extraits de flux multimodaux qui, de par leur nature, peuvent être affectés par le **bruit**, des changements de **contexte** ou des ajustements successifs induits par un fine-tuning continu. Dans cette section, nous exposons en détail deux scénarios qui démontrent comment le DSL exploite des mécanismes de limitation d'erreurs, tels que le filtrage local et le clipping, ainsi que des stratégies d'adaptation dynamique, afin de préserver une organisation robuste et cohérente dans un environnement audio-visuel.

A. Détection d'Événements Audio-Visuels

Dans un premier exemple, nous considérons la détection d'un événement, tel qu'un applaudissement ou un klaxon, qui se manifeste simultanément dans la composante visuelle et acoustique d'un flux multimodal. Pour ce faire, le flux audio-visuel est segmenté en une série de **frames vidéo** $\{\mathcal{E}_i^{(vid)}\}$ et en une série de **segments audio** $\{\mathcal{E}_j^{(aud)}\}$. Chaque frame est alors transformée en un **embedding** $\mathbf{x}_i^{(vid)} \in \mathbb{R}^{d_{vid}}$ par un modèle de type CNN (tel que ResNet) ou par un Vision Transformer, tandis que chaque segment audio se voit attribuer un vecteur $\mathbf{x}_j^{(aud)} \in \mathbb{R}^{d_{aud}}$ issu d'un modèle d'autoencodeur appliqué sur un spectrogramme ou d'un transformeur audio tel que wav2vec. La **synergie** entre une frame vidéo et un segment audio peut être quantifiée à l'aide d'un noyau gaussien, défini par

$$S(i,j) = \exp(-\gamma \|\mathbf{x}_i^{(vid)} - \mathbf{x}_j^{(aud)}\|^2),$$

où $\gamma > 0$ ajuste la sensibilité de la mesure. Toutefois, en raison des perturbations inhérentes à ce type de flux – telles que la saturation audio ou les variations lumineuses dans les images –, des valeurs aberrantes peuvent fausser le calcul de la synergie. Pour limiter ces effets, le DSL intègre des mécanismes de **filtrage local** en restreignant le calcul de la similarité aux paires d’entités se chevauchant temporellement, ainsi que des techniques de **clipping** qui borner les valeurs extrêmes du score $S(i, j)$. Ce filtrage permet de constituer des **clusters** robustes reliant les frames et segments audio fortement synchronisés, ce qui facilite la détection d’événements multimodaux avec une grande précision.

B. Sous-Titrage Automatique à Partir d'un Flux Audio-Visuel

Dans un second scénario, le DSL est appliqué à la tâche de sous-titrage automatique, qui implique la fusion d’informations issues du flux vidéo et du flux textuel dérivé d’une transcription audio. Dans ce contexte, chaque frame vidéo $\mathcal{E}_i^{(\text{vid})}$ se voit attribuer un embedding $\mathbf{x}_i^{(\text{vid})}$ généré par un Vision Transformer ou un CNN, tandis que la transcription issue d’un module speech-to-text est convertie en un **embedding textuel** $\mathbf{x}_j^{(\text{text})} \in \mathbb{R}^{d_{\text{text}}}$ via un modèle de type BERT ou GPT. La synergie entre la représentation visuelle et celle textuelle peut être évaluée à l’aide de la **similarité cosinus**, exprimée par

$$S_{\cos}(i, j) = \frac{\mathbf{x}_i^{(\text{vid})} \cdot \mathbf{x}_j^{(\text{text})}}{\|\mathbf{x}_i^{(\text{vid})}\| \|\mathbf{x}_j^{(\text{text})}\|}.$$

Cette mesure permet de quantifier la proximité sémantique entre le contenu visuel et le texte associé. Cependant, des erreurs peuvent survenir dans la transcription, notamment en cas de bruit, d’accents ou de débit de parole rapide, tandis que la qualité des embeddings vidéo peut être affectée par des mouvements brusques ou des changements de scène. Afin de pallier ces imprécisions, le DSL met en place des mécanismes de **limitation temporelle** (en ne comparant que les frames proches du segment textuel) ainsi que des procédures de **clipping** pour éviter que des valeurs extrêmes ne conduisent à des scores de similarité erronés. De cette manière, le système parvient à associer de manière cohérente les segments vidéo et les extraits textuels, facilitant ainsi la génération de sous-titres ou de résumés contextuels de haute qualité.

C. Ajustement Dynamique et Fine-Tuning

Dans les deux scénarios précédents, le DSL peut être soumis à des processus de **fine-tuning** ou à des mises à jour continues, de sorte que les modèles générateurs d’embeddings (qu’il s’agisse d’un CNN, d’un Vision Transformer ou d’un modèle speech-to-text) évoluent en fonction des nouvelles données. Ainsi, les représentations \mathbf{x}_i et \mathbf{x}_j peuvent être réévaluées périodiquement, modifiant la synergie $S(i, j)$ calculée et, par conséquent, les pondérations $\omega_{i,j}$ dans le SCN. Afin d’éviter que ces changements brusques ne perturbent la dynamique globale du réseau, des techniques de **lissage** des transitions et de **verrouillage** partiel des modèles sont mises en œuvre. Ces stratégies visent à adapter les embeddings de manière progressive, garantissant ainsi une réorganisation continue mais stable des clusters d’entités multimodales.

Conclusion

Les scénarios audio-visuels présentés illustrent de manière concrète la capacité du DSL à intégrer plusieurs modalités, en utilisant des mécanismes tels que le **filtrage local** et le **clipping** pour limiter l'impact des perturbations et des valeurs aberrantes. Dans le cas de la **détection d'événements audio-visuels**, la synergie entre des frames vidéo et des segments audio est exploitée pour former des clusters cohérents qui permettent d'identifier des événements spécifiques avec une grande robustesse face aux bruits et aux variations contextuelles. De même, pour le **sous-titrage automatique**, l'association des embeddings vidéo et textuels via la similarité cosinus conduit à une correspondance précise entre les contenus visuels et les transcriptions, malgré les éventuelles erreurs ou instabilités dans les modules de reconnaissance vocale et d'analyse d'image. L'ensemble de ces mécanismes, associé à une adaptation dynamique des modèles par des processus de fine-tuning, garantit que le DSL maintienne une organisation cohérente et robuste, capable de traiter des flux multimodaux hétérogènes tout en assurant une **explicabilité** et une **stabilité** dans les réponses générées par le système.

3.4. Représentations Symboliques

Alors que les **représentations sub-symboliques** (vecteurs, embeddings) dominent de nombreuses applications d'apprentissage (sections 3.2 et 3.3), il existe un autre univers de modélisation : celui des **représentations symboliques**. Dans ces approches, chaque entité \mathcal{E}_i est décrite non plus par un vecteur, mais par un **ensemble de règles**, de **concepts** ou d'**axiomes** pouvant relever de la **logique** (règles “if... then...”) ou de la **sémantique ontologique** (graphe RDF, OWL, etc.). Cette manière de représenter l'information favorise l'**interprétabilité** et le **raisonnement formel**, deux propriétés parfois difficiles à obtenir avec des embeddings “boîte noire”. Toutefois, elle pose également des **défis** : la nécessité de maintenir la cohérence logique, l'incapacité à gérer aisément le bruit, ou encore la rigidité face aux nouveaux attributs.

Dans cette section (3.4), nous étudierons d'abord (3.4.1) les **logiques, règles et ontologies** (une palette symbolique de base), puis (3.4.2) nous verrons **comment** calculer la synergie symbolique via des mesures de **compatibilité** ou de **cohérence**, et enfin (3.4.3) nous aborderons l'**évolution** d'un ensemble symbolique et la gestion des contradictions dans le temps.

3.4.1. Logiques, Règles et Ontologies

Les **représentations symboliques** couvrent un vaste champ : de la simple règle conditionnelle (“si A alors B”) jusqu’aux **grandes ontologies** hiérarchisées (OWL, RDF) décrivant un univers de concepts et de relations. L’objectif est de permettre à la machine de **raisonner** ou de **valider** certaines assertions de manière déclarative.

3.4.1.1. Approche Symbolique : Ensembles de Règles (if A then B), Graphes Sémantiques, Ontologies (OWL, RDF)

Dans le cadre de l’analyse des systèmes d'**auto-organisation** et d'**apprentissage** tels que le **Deep Synergy Learning (DSL)**, il est essentiel de distinguer les approches **symboliques** des méthodes **sub-symboliques**. Alors que ces dernières représentent les entités \mathcal{E}_i par des **embeddings** (obtenus par des réseaux convolutionnels, transformers, autoencodeurs, etc.) – c'est-à-dire par des vecteurs dans un espace continu \mathbb{R}^d –, l’approche symbolique se fonde sur des structures explicites et logiques. Dans ce contexte, les entités sont décrites à l'aide d'**ensembles de règles** du type « *if A then B* », de **graphes sémantiques** structurés selon des triplets, ou encore d'**ontologies** formelles telles que celles définies par OWL et RDF. Ces différentes modalités de représentation offrent chacune des avantages spécifiques en termes d'**interprétabilité**, de **raisonnement formel** et de **tracabilité** de l'information.

A. Ensembles de Règles “If A then B”

L’approche la plus intuitive dans le domaine symbolique consiste à modéliser les connaissances sous forme d’implications logiques, telles que

$$\text{if } A \text{ then } B,$$

où A représente une hypothèse ou une condition initiale et B une conclusion ou action associée. Mathématiquement, une telle règle peut être exprimée par la proposition

$$A(\mathcal{E}_i) \Rightarrow B(\mathcal{E}_i),$$

indiquant que si l'entité \mathcal{E}_i satisfait l'ensemble de conditions A , alors elle doit logiquement satisfaire la conclusion B . Par exemple, dans un contexte médical, une règle du type

if fièvre et toux alors suspect infection respiratoire

permet d'associer un ensemble d'attributs booléens à une conclusion diagnostic. L'**avantage** de cette représentation est sa grande **interprétabilité** : l'humain peut aisément comprendre la justification d'une déduction, ce qui est primordial dans des domaines nécessitant une forte **traçabilité** comme la médecine ou le droit. Cependant, l'**inconvénient** majeur réside dans la **rigidité** des règles. Une légère variation dans les conditions A peut rendre la règle inapplicable, et, à grande échelle, la gestion d'un vaste ensemble de règles peut conduire à des conflits et à une explosion combinatoire difficilement maîtrisable.

B. Graphes Sémantiques

Les **graphes sémantiques** offrent une représentation plus structurée des connaissances en modélisant chaque entité \mathcal{E}_i par un sous-graphe composé de nœuds (représentant des concepts ou attributs) et d'arcs (indiquant les relations entre ces concepts). Dans le standard **RDF** (Resource Description Framework), toute information est codée sous la forme de triplets (s, p, o) où s est le sujet, p le prédicat, et o l'objet. Par exemple, les triplets

(Brutus, type, Chien), (Chien, subclass-of, Mammifère), (Mammifère, subclass-of, Animal),

permettent d'établir une hiérarchie de classes. L'**avantage** des graphes sémantiques réside dans leur capacité à représenter explicitement des **hiérarchies** et à permettre des **requêtes complexes** via des langages tels que **SPARQL**. Ils facilitent ainsi la navigation et l'inférence sur les relations entre entités. Toutefois, ils présentent également des **inconvénients**, notamment la difficulté à maintenir la **consistance** de grandes bases de connaissances et leur sensibilité aux erreurs : un triplet erroné peut perturber l'ensemble de la structure.

C. Ontologies (OWL, RDFSchema)

Les **ontologies** représentent une extension des graphes sémantiques en imposant un formalisme rigoureux pour décrire des classes, des propriétés, et des relations hiérarchiques ou restrictions (telles que des cardinalités ou des disjonctions). Par exemple, dans une ontologie OWL, on pourra formuler des axiomes tels que

Chien \sqsubseteq Mammifère et hasOwner:Chien \rightarrow Humain.

Dans ce cadre, un individu (par exemple, un chien nommé Brutus) sera déclaré comme une instance de la classe **Chien**, et par transitivité, comme instance de **Mammifère** et d'**Animal**. L'**avantage** majeur de cette approche est la **précision** sémantique qu'elle apporte et la capacité d'effectuer des **raisonnements automatiques** grâce à des moteurs d'inférence tels que **HermiT** ou **Fact++**. Cependant, l'**inconvénient** réside dans la complexité de la maintenance d'une ontologie de grande échelle, qui peut devenir difficile à gérer dans des environnements où les connaissances évoluent rapidement ou sont sujettes à des **bruits** et des imprécisions.

Conclusion

Les **méthodes symboliques** – qu'il s'agisse d'ensembles de règles implicatives, de graphes sémantiques ou d'ontologies formelles – offrent une représentation hautement **explicable** et **logiquement manipulable** des entités. Elles contrastent avec les **embeddings** sub-symboliques, qui fournissent une représentation dense et opaque en termes d'informations numériques. Dans le cadre d'un **DSL** (Deep Synergy Learning), l'utilisation d'approches symboliques présente l'avantage de rendre la **tracabilité** des relations explicite et de permettre un **raisonnement formel** (détection de contradictions, inférences automatiques), ce qui est particulièrement utile dans des secteurs nécessitant une forte **transparence**. Néanmoins, ces méthodes nécessitent souvent une intervention humaine pour la mise à jour et le maintien de la cohérence de la base de connaissances, et elles sont moins tolérantes aux **bruits** ou aux variations légères que les représentations sub-symboliques.

Dans les sections ultérieures, notamment en **3.4.2.2** et **3.4.3**, nous approfondirons les méthodes de calcul de la **synergie** $S(i,j)$ entre des blocs symboliques (par exemple, via la distance sémantique ou l'intersection d'axiomes) ainsi que les mécanismes de mise à jour pour gérer l'évolution des règles et des structures ontologiques au sein d'un **SCN**. Ces techniques se révèlent essentielles pour déployer un **DSL** dans des environnements où l'explicabilité, la **raison** et la **validation** formelle sont requises, que ce soit en médecine, en finance, ou dans d'autres domaines critiques.

Ainsi, la **représentation symbolique** dans le **DSL** permet d'englober des modèles tels que les **ensembles de règles** (if A then B), les **graphes sémantiques** (RDF) et les **ontologies** (OWL, RDFSchema), offrant ainsi une alternative complémentaire aux approches sub-symboliques. Cette diversité de méthodes enrichit le cadre du **DSL** et prépare le terrain pour une intégration fluide entre la **logique formelle** et les **représentations continues**, contribuant ainsi à la création de systèmes d'**intelligence** plus robustes et interopérables.

3.4.1.2. Avantages : Interprétabilité, Raisonnement Formel

Dans l'analyse des systèmes d'**auto-organisation** et d'**apprentissage** tels que le **Deep Synergy Learning (DSL)**, les approches symboliques – fondées sur des **ensembles de règles**, des **graphes sémantiques** et des **ontologies** – se distinguent nettement des méthodes sub-symboliques qui s'appuient sur des **embeddings** et des **vecteurs** numériques. Ces approches symboliques apportent une **interprétabilité** et une **capacité de raisonnement formel** qui facilitent l'explication, la vérification et l'audit des décisions prises par le système, qualités indispensables dans des domaines où la **tracabilité** et la **cohérence** des conclusions sont primordiales, tels que la médecine, le droit ou l'ingénierie.

A. Interprétabilité Explicite

Dans une représentation symbolique, chaque entité \mathcal{E}_i est décrite par un ensemble explicite de **règles logiques**, **d'axiomes** ou **d'assertions** qui prennent la forme d'implications telles que

$$\text{if } A \text{ then } B,$$

où A représente l'hypothèse et B la conclusion. Par exemple, une règle de diagnostic médical peut être formulée comme suit :

if Toux and Fièvre then InfectionRespiratoire.

Ce type de formulation permet à un expert – qu'il soit médecin ou ingénieur – de **lire** facilement la logique sous-jacente et de comprendre de manière transparente **pourquoi** un ensemble de conditions mène à une conclusion particulière. Contrairement aux **embeddings** sub-symboliques, où une entité est représentée par un vecteur $\mathbf{x}_i \in \mathbb{R}^d$ dont les composantes sont souvent difficiles à interpréter, les règles symboliques exposent directement les **relations** et les **attributs** qui justifient la déduction. Dans le contexte d'un DSL, cette **lisibilité** se traduit par une **träcabilité** accrue des liens établis entre les entités. Ainsi, il est possible de lister l'ensemble des règles $\{R_i\}$ associées à chaque entité \mathcal{E}_i et d'observer, de manière explicite, comment ces règles favorisent la formation ou la dissolution d'un **cluster**. Ce mécanisme offre une **confiance** renforcée dans les décisions du système, particulièrement dans des environnements critiques où il est essentiel de pouvoir expliquer les raisons d'un diagnostic ou d'une classification.

B. Raisonnement Formel et Inférences

Un autre avantage majeur des approches symboliques réside dans leur capacité à faciliter le **raisonnement formel** à l'aide de moteurs logiques tels que Prolog ou des reasoners OWL. Ces outils permettent d'effectuer des **inférences** automatiques et de vérifier la **consistance** d'un ensemble de règles. Par exemple, si l'on sait qu'« tout Mammifère a du sang chaud » et que l'on déclare que « Chien est Mammifère », il est possible de déduire automatiquement que « Chien a du sang chaud ». Dans un DSL, deux entités \mathcal{E}_i et \mathcal{E}_j , dotées respectivement de blocs de règles $\{R_i\}$ et $\{R_j\}$, peuvent ainsi afficher une **synergie** élevée, notée

$$\mathbf{S}_{\text{sym}}(i, j) = f(R_i, R_j),$$

si leurs ensembles de règles sont logiquement compatibles ou se complètent. La fonction f peut mesurer, par exemple, la proportion des règles communes, la non-contradiction entre les axiomes, ou encore la conséquence logique partagée par les deux entités. De ce fait, le DSL n'est pas uniquement un système de **clusterisation** basé sur des distances numériques, mais également un outil de **raisonnement**, capable de renforcer les liens entre des entités en fonction de leur **cohérence logique**.

D. Audit et Justification des Décisions

L'un des défis majeurs dans les systèmes d'auto-organisation basés sur des embeddings est le manque de transparence quant aux raisons sous-jacentes à la formation d'un cluster. Dans un DSL symbolique, il est possible d'**auditer** les décisions prises par le système en listant explicitement les règles ou axiomes partagés entre deux entités \mathcal{E}_i et \mathcal{E}_j . Par exemple, pour justifier pourquoi ces deux entités ont été regroupées, on peut présenter l'ensemble des attributs et des conditions qui se recoupent, telles que « elles appartiennent toutes deux à la classe Mammifère » ou « elles possèdent la même propriété hasSymptom('Toux') ». Ce mécanisme de **justification** renforce la crédibilité du système et facilite la prise de décision dans des contextes où il est impératif de pouvoir expliquer les processus de classification ou de diagnostic.

E. Réutilisation et Interopérabilité des Bases de Connaissances

Enfin, les approches symboliques bénéficient d'un important **avantage** en termes de **réutilisation** des ressources existantes. De nombreux domaines disposent déjà d'**ontologies** ou de **répertoires**

de règles validés par des experts – tels que **Snomed-CT** et **UMLS** en médecine, **FIBO** dans la finance, ou **ISO 15926** en ingénierie. Dans un DSL symbolique, chaque entité \mathcal{E}_i peut s'appuyer sur ces bases de connaissances pour hériter de classes et de propriétés prédefinies, ce qui facilite l'intégration et l'interopérabilité avec des systèmes existants. La synergie entre les entités, notée $\mathbf{S}(i,j)$, peut ainsi bénéficier d'une sémantique partagée et validée, renforçant la cohérence globale du réseau et accélérant l'émergence de clusters pertinents.

Conclusion

Les approches symboliques dans le cadre du **Deep Synergy Learning** offrent des avantages considérables en termes d'**interprétabilité** et de **raisonnement formel**. En permettant une représentation explicite des entités sous forme de règles, graphes sémantiques ou ontologies, elles assurent une **tracabilité** claire et une **justification** des décisions prises par le système. La capacité à utiliser des moteurs logiques pour inférer de nouvelles relations et vérifier la cohérence des ensembles de règles constitue un atout majeur, surtout dans des domaines où la **transparence** et la **validation** formelle sont indispensables. Par ailleurs, l'interopérabilité avec des bases de connaissances existantes renforce la robustesse et la crédibilité du DSL. Ces avantages sont particulièrement mis en lumière dans les sections **3.4.2.2** et **3.4.3**, qui approfondiront respectivement la définition de la synergie symbolique et les mécanismes d'évolution des structures logiques dans un SCN. Ainsi, la représentation symbolique enrichit le DSL en offrant une alternative complémentaire aux méthodes sub-symboliques, ouvrant la voie à des systèmes d'intelligence plus transparents, explicables et vérifiables dans des environnements critiques.

3.4.1.3. Limites : Rigidité, Besoin de Maintenir la Cohérence Logique

Dans le cadre de la représentation symbolique au sein d'un **Deep Synergy Learning (DSL)**, l'utilisation de règles logiques, de graphes sémantiques et d'ontologies offre une grande **interprétabilité** et permet un raisonnement formel détaillé (voir section **3.4.1.2**). Cependant, ces avantages s'accompagnent de limites intrinsèques, notamment la **rigidité** inhérente aux systèmes symboliques et le besoin impératif de maintenir une **cohérence logique** à l'échelle du système. Dans cette section, nous examinons en détail ces contraintes, en explicitant leurs implications tant du point de vue théorique que pratique.

D'une part, la **rigidité** des représentations symboliques découle de leur caractère souvent **binaire** et catégorique. En effet, dans un système purement symbolique, une règle ou un axiome est formulé sous la forme d'une implication logique, par exemple :

if A then B ,

où A et B représentent respectivement des conditions et des conclusions exprimées à l'aide de prédicats ou de variables booléennes. Cette formulation impose une discontinuité : une petite variation dans la valeur de A (par exemple, une température mesurée à 37,9 °C au lieu de 38 °C) peut conduire à l'échec complet de la condition, ce qui invalide la déduction de B . Mathématiquement, si l'on définit une fonction caractéristique $\chi_A(x)$ associée à la condition A de telle sorte que

$$\chi_A(x) = \begin{cases} 1, & \text{si } x \in A, \\ 0, & \text{sinon,} \end{cases}$$

alors une légère déviation qui ferait basculer $\chi_A(x)$ de 1 à 0 engendre une discontinuité de la fonction de déduction. Cette rigidité ne permet pas de modéliser des phénomènes progressifs ou graduels, comme le ferait une mesure sub-symbolique où les distances ou similarités se modélisent par des fonctions continues telles que la distance euclidienne ou le cosinus d'angle. Par conséquent, dans des environnements où les données sont bruitées ou incertaines, une approche symbolique stricte peut conduire à des ruptures soudaines de synergie, rendant la dynamique du DSL moins robuste.

D'autre part, la **nécessité de maintenir la cohérence logique** impose une contrainte supplémentaire dans l'architecture d'un DSL symbolique. Lorsqu'un ensemble d'entités $\{\mathcal{E}_i\}$ est caractérisé par des blocs de règles ou des ontologies, toute modification apportée à l'un de ces blocs doit être vérifiée globalement afin de préserver la consistance du système. Par exemple, considérons les axiomes suivants :

- (i) $\forall x, \text{if } x \in \text{Mammifère} \text{ then } x \text{ a du sang chaud},$
- (ii) \mathcal{E}_i est un Mammifère,
- (iii) \mathcal{E}_i a du sang froid.

L'introduction simultanée des axiomes (i) et (ii) conduit inévitablement à la conclusion x a du sang chaud pour \mathcal{E}_i , ce qui entre en contradiction avec (iii). Dès lors, la fonction de **synergie** entre les entités, que l'on peut formuler symboliquement par une fonction f telle que

$$\mathbf{S}_{\text{sym}}(i, j) = f(R_i, R_j),$$

doit être continuellement recalculée et validée afin de s'assurer que les modifications locales (par exemple, l'ajout, la suppression ou la modification d'un axiome dans R_i) ne conduisent pas à des contradictions globales. Ce processus de vérification peut être formulé à l'aide de systèmes de contraintes ou d'algorithmes d'inférence logiques, mais il ajoute une complexité algorithmique non négligeable, en particulier lorsque le nombre d'entités et de règles est très élevé. Du point de vue de la **théorie de la complexité**, la vérification de la cohérence d'une ontologie expressive relève souvent d'un problème NP-complet, voire de classes plus difficiles comme PSPACE ou EXPTIME dans certains cas (cf. OWL DL). Cela limite la **scalabilité** de l'approche symbolique dans un environnement dynamique et massivement distribué.

En outre, la mise à jour incrémentale d'un grand nombre de règles dans un DSL exige une **maintenance** continue de la base de connaissances. Chaque modification locale, par exemple l'ajout d'un nouvel axiome dans le bloc de règles R_i , peut avoir un effet domino sur l'ensemble du système. En effet, une telle modification nécessite non seulement de recalculer la synergie $\mathbf{S}_{\text{sym}}(i, j)$ entre \mathcal{E}_i et toutes les autres entités \mathcal{E}_j , mais aussi de vérifier que cette modification ne viole pas la cohérence globale du système. En d'autres termes, la dynamique du DSL symbolique peut être ralentie par le temps nécessaire aux opérations de validation, ce qui représente un inconvénient majeur lorsqu'on cherche à opérer des mises à jour en temps réel ou en continu.

Enfin, il faut noter que les **systèmes symboliques** purs, en raison de leur caractère binaire, ne gèrent pas naturellement un **degré de confiance** ou un **niveau de flou** dans la représentation des faits. Pour pallier ce manque, il est parfois nécessaire d'introduire des logiques floues ou probabilistes, telles que les réseaux bayésiens ou les Markov Logic Networks, ce qui complexifie encore la

modélisation et l'implémentation. La **conversion** d'un problème symbolique strict en un problème flou nécessite souvent une transformation non triviale de la fonction de synergie, et par conséquent une réévaluation de toute la dynamique d'auto-organisation.

Conclusion

Les limites inhérentes aux représentations symboliques dans le cadre d'un **Deep Synergy Learning** se résument principalement à deux points : la **rigidité** des règles, qui ne permet pas de gérer efficacement le bruit et l'incertitude, et le **besoin constant de maintenir la cohérence logique** dans un ensemble de règles potentiellement très étendu. Ces contraintes se traduisent par une moindre tolérance aux variations et une complexité accrue lors de la mise à jour ou de l'inférence, ce qui peut freiner l'auto-organisation dynamique dans des environnements où les données évoluent rapidement. Dans des domaines critiques, cette approche reste indispensable en raison de sa **tracerabilité** et de sa capacité à fournir des justifications formelles, mais son application à grande échelle nécessite souvent des solutions hybrides qui combinent la souplesse des méthodes sub-symboliques avec la robustesse du raisonnement logique. Les sections ultérieures (3.4.2 et 3.4.3) exploreront davantage ces compromis et présenteront des stratégies pour intégrer des méthodes de **logique floue** et des mécanismes d'inférence optimisés afin d'améliorer l'adaptabilité et la scalabilité du DSL symbolique.

3.4.2. Calcul de la Synergie Symbolique

Lorsque deux entités \mathcal{E}_i et \mathcal{E}_j sont décrites par des **représentations symboliques** (ensembles de règles, ontologies, graphes sémantiques), la notion de synergie $S(i, j)$ ne repose plus sur une simple distance ou similarité vectorielle. Il s'agit plutôt d'évaluer la **compatibilité**, la **non-contradiction** ou la **cohérence** entre leurs blocs de connaissances. Dans ce paragraphe, nous évoquerons successivement :

169. (3.4.2.1) la **compatibilité de règles**, le **degré de contradiction** ou la **co-occurrence** des axiomes,
170. (3.4.2.2) la **similarité** entre ontologies (ou sous-ontologies),
171. (3.4.2.3) des approches de **distance sémantique** plus générales, voire d'**information mutuelle symbolique**.

3.4.2.1. Compatibilité de Règles, Degré de Contradiction ou Co-Occurrence

Dans le cadre du **Deep Synergy Learning (DSL)**, lorsque les entités \mathcal{E}_i sont décrites de manière symbolique – par exemple, à l'aide d'ensembles de règles, d'axiomes ou de sous-graphes ontologiques – la **synergie** $S(i, j)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j dépend essentiellement de la **compatibilité** de leurs ensembles de règles, notés respectivement R_i et R_j . La compatibilité peut être vue comme le degré de recouvrement entre les deux ensembles, mais aussi comme l'absence de contradictions logiques entre eux. Nous exposerons ci-après les différents aspects de cette démarche, en intégrant les formules mathématiques nécessaires pour formaliser ces notions.

A. Définition de la Compatibilité ou Co-Occurrence

Une première approche pour quantifier la proximité symbolique entre R_i et R_j consiste à considérer leurs ensembles comme des collections d'éléments discrets et à mesurer leur **intersection** par rapport à leur **union**. On définit ainsi la mesure de recouvrement ou de **co-occurrence** par la formule :

$$S_{\text{overlap}}(i, j) = \frac{|R_i \cap R_j|}{|R_i \cup R_j|}.$$

Cette expression renvoie une valeur comprise entre 0 et 1, où un score proche de 1 indique que les deux ensembles partagent une grande partie de leurs règles (ce qui traduit une forte **similarité conceptuelle**), tandis qu'un score proche de 0 indique un faible recouvrement. Dans le contexte d'un **Synergistic Connection Network (SCN)**, il est alors possible de relier la pondération $\omega_{i,j}$ entre deux entités à cette mesure de recouvrement. Autrement dit, lorsque R_i et R_j partagent un nombre important de règles, le système renforcera la connexion entre \mathcal{E}_i et \mathcal{E}_j .

Cas pratique :
 Prenons l'exemple de deux entités médicales. Supposons que l'entité \mathcal{E}_i possède l'ensemble de règles R_i incluant des assertions telles que « if Toux and Fièvre then InfectionRespiratoire », et que \mathcal{E}_j comporte un ensemble R_j similaire, par exemple incluant aussi « if Toux and Fièvre then InfectionRespiratoire » et d'autres règles liées au diagnostic d'infections respiratoires. Si $|R_i \cap R_j|$ est élevé, cela indique que les deux entités appartiennent au même domaine diagnostique, et le score $S_{\text{overlap}}(i, j)$ sera alors élevé, conduisant à un renforcement de la pondération $\omega_{i,j}$.

B. Degré de Contradiction

Outre le recouvrement, il est crucial de mesurer si les ensembles R_i et R_j présentent des **contradictions** logiques. En logique formelle, on dit qu'il y a contradiction lorsque la conjonction de certaines formules $a \in R_i$ et $b \in R_j$ conduit à une fausseté, symbolisée par \perp . Pour formaliser ce concept, on définit la fonction de contradiction de manière binaire par :

$$\text{Contradiction}(R_i, R_j) = \begin{cases} 1, & \text{si } \exists (a \in R_i, b \in R_j) \text{ tels que } a \wedge b \models \perp, \\ 0, & \text{sinon.} \end{cases}$$

Cette définition peut être affinée en comptant le nombre de conflits entre les règles, ce qui se traduit par :

$$\text{ContradictionCount}(R_i, R_j) = \sum_{(a,b) \in R_i \times R_j} \mathbf{1}[a \wedge b \models \perp],$$

où $\mathbf{1}[\cdot]$ est la fonction indicatrice qui vaut 1 si la condition est satisfaite et 0 sinon. Un nombre élevé de contradictions, c'est-à-dire une grande valeur de $\text{ContradictionCount}(R_i, R_j)$, devrait entraîner une diminution de la synergie $\mathbf{S}(i, j)$ entre les deux entités, car il n'est pas souhaitable de regrouper des entités dont les règles sont logiquement incompatibles.

C. Combinaison de Co-Occurrence et de Contradiction : Synergie Symbolique

Pour tenir compte simultanément du recouvrement et des contradictions, il convient de définir une fonction de **synergie symbolique** qui intègre à la fois un terme positif relatif à l'intersection des ensembles et un terme négatif qui pénalise les contradictions. On peut définir cette fonction par :

$$S_{\text{sym}}(i, j) = \alpha \frac{|R_i \cap R_j|}{|R_i \cup R_j|} - \beta \text{ContradictionCount}(R_i, R_j),$$

où α et β sont des constantes strictement positives qui pondèrent respectivement l'importance du recouvrement et celle du degré de contradiction. Ce score, que l'on peut ensuite contraindre à être non négatif (par exemple, en le plafonnant à zéro si nécessaire), guide la dynamique du DSL en influençant la mise à jour des pondérations selon la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{sym}}(i, j) - \tau \omega_{i,j}(t)],$$

où η est le taux d'apprentissage et τ le coefficient de décroissance. Dans cette dynamique, si $\text{ContradictionCount}(R_i, R_j)$ est élevé, le terme $S_{\text{sym}}(i, j)$ diminue, conduisant à un affaiblissement de $\omega_{i,j}$.

D. Complexité d'Inférence Logique

La mise en œuvre pratique de ces concepts pose toutefois des défis liés à la **complexité** du raisonnement logique. La vérification de la contradiction, c'est-à-dire déterminer si une conjonction $a \wedge b$ déduit \perp , nécessite l'emploi d'un **raisonneur** ou d'un module de déduction automatique. Dans des ontologies ou des systèmes logiques expressifs (par exemple, OWL avec une expressivité complète), la vérification de la cohérence d'un ensemble de règles peut être un problème de complexité exponentielle. Pour atténuer ce problème, il est souvent nécessaire de :

- Limiter l'expressivité du système logique en passant à des versions simplifiées telles que RDF basique ou OWL Lite.
- Utiliser des structures d'indexation ou des techniques de hashage sémantique pour repérer rapidement les conflits potentiels.
- Se concentrer sur des sous-ensembles critiques des ensembles R_i et R_j lorsqu'un recouvrement minimal est déjà détecté, afin de réduire le nombre de vérifications nécessaires.

Ces heuristiques permettent d'obtenir une approximation de la synergie symbolique qui reste **scalable** pour de grandes bases de connaissances.

E. Application dans la Mise à Jour $\omega_{i,j}$

L'ensemble des considérations ci-dessus se traduit directement dans la règle d'auto-organisation du **SCN**. En effet, la synergie symbolique $S_{\text{sym}}(i, j)$ influence la dynamique des pondérations $\omega_{i,j}$ par la mise à jour suivante :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{sym}}(i, j) - \tau \omega_{i,j}(t)].$$

Ce schéma permet d'ajuster les liens entre entités en fonction de la compatibilité de leurs ensembles de règles. Ainsi, si deux entités partagent un grand nombre d'axiomes sans contradictions (c'est-à-dire si $S_{\text{overlap}}(i, j)$ est élevé et $\text{ContradictionCount}(R_i, R_j)$ est faible), la pondération $\omega_{i,j}$ sera renforcée. À l'inverse, la présence de contradictions entraînera une réduction de $\omega_{i,j}$, contribuant à la formation de clusters de règles cohérentes.

Conclusion

La mesure de synergie symbolique dans un **DSL** s'appuie sur la combinaison du **recouvrement** entre les ensembles de règles et de la détection des **contradictions** logiques. On définit d'abord la proximité par :

$$S_{\text{overlap}}(i, j) = \frac{|R_i \cap R_j|}{|R_i \cup R_j|},$$

puis on introduit une pénalité pour les conflits par :

$$\text{ContradictionCount}(R_i, R_j) = \sum_{(a,b) \in R_i \times R_j} \mathbf{1}[a \wedge b \models \perp].$$

En combinant ces deux mesures, la synergie symbolique est définie comme :

$$S_{\text{sym}}(i, j) = \alpha \frac{|R_i \cap R_j|}{|R_i \cup R_j|} - \beta \text{ContradictionCount}(R_i, R_j),$$

où α et β sont des coefficients de pondération positifs. Cette fonction intervient dans la dynamique de mise à jour des pondérations :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{sym}}(i, j) - \tau \omega_{i,j}(t)].$$

Ce schéma permet au **SCN** de favoriser les liaisons entre entités logiquement cohérentes tout en pénalisant celles qui présentent des incohérences. Néanmoins, la complexité inhérente au raisonnement logique, qui peut être exponentielle dans des systèmes très expressifs, impose le recours à des heuristiques pour garantir la **scalabilité** du système. Ces considérations sont essentielles pour maintenir une auto-organisation efficace dans un **DSL** symbolique, tout en assurant une cohérence globale des règles. Les sections **3.4.2** et **3.4.3** approfondiront les compromis et les stratégies d'intégration de logiques floues, afin d'améliorer l'adaptabilité et la robustesse des systèmes symboliques dans un cadre de Deep Synergy Learning.

3.4.2.2. Similarité d'Ontologies, Mesure d'Overlap Conceptuel

Dans le cadre du **Deep Synergy Learning (DSL)**, lorsque les entités \mathcal{E}_i sont décrites par des représentations symboliques, leur caractérisation repose souvent sur des ontologies formalisées selon des standards tels que OWL ou RDF. Ces ontologies définissent de manière explicite des **classes**, des **sous-classes**, des **relations** et des **axiomes** qui structurent la connaissance d'un domaine. Ainsi, chaque entité \mathcal{E}_i se voit associée à un **sous-graphe ontologique** \mathcal{G}_i , lequel regroupe

l'ensemble des triplets, assertions et axiomes pertinents à cette entité. Pour évaluer la **synergie** $S(i,j)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j , il convient de mesurer la similarité entre leurs sous-graphes respectifs, c'est-à-dire d'estimer l'**overlap conceptuel** entre \mathcal{G}_i et \mathcal{G}_j . Cette section présente une approche formelle et détaillée de la mesure d'overlap conceptuel, en intégrant des formules mathématiques précises pour quantifier cette similarité.

A. Notion d'Ontologie et de Sous-Graphe

Une **ontologie** se définit comme un ensemble structuré de concepts, de relations et d'axiomes qui permettent de représenter la connaissance d'un domaine de manière formelle. Dans ce contexte, chaque entité \mathcal{E}_i est associée à un sous-graphe ontologique \mathcal{G}_i qui contient les triplets RDF et autres assertions telles que

$$\mathcal{E}_i \text{ rdf:type } \mathcal{C}, \quad \mathcal{C}_1 \text{ rdfs:subClassOf } \mathcal{C}_2,$$

ainsi que d'autres relations définies en OWL, comme « equivalentClass » ou « disjointWith ». Cette représentation permet d'organiser la connaissance en hiérarchies et en réseaux de relations sémantiques. Elle offre l'avantage de rendre explicite la structure de la connaissance et de faciliter le raisonnement automatique grâce à des reasoners tels que HermiT ou Fact++.

B. Mesure d'Overlap Conceptuel

1. Overlap de Triplets

La méthode la plus directe pour quantifier la similarité entre deux sous-graphes \mathcal{G}_i et \mathcal{G}_j consiste à mesurer le **recouvrement** de leurs triplets. Si l'on considère chaque triplet comme un élément discret, une mesure simple de co-occurrence peut être définie par l'opérateur suivant :

$$S_{\text{ont}}(i,j) = \frac{|\mathcal{G}_i \cap \mathcal{G}_j|}{|\mathcal{G}_i \cup \mathcal{G}_j|}.$$

Cette expression attribue une valeur comprise entre 0 et 1. Un score proche de 1 signifie que les deux sous-graphes partagent presque tous leurs triplets, indiquant une forte **cohérence sémantique**. À l'inverse, un score proche de 0 indique que les deux entités se définissent par des concepts et relations très différents. Ainsi, dans le cadre d'un **SCN**, la pondération $\omega_{i,j}$ peut être renforcée lorsque $S_{\text{ont}}(i,j)$ est élevé, favorisant la formation de clusters d'entités ontologiquement similaires.

2. Isomorphisme Partiel et Distance Hiérarchique

Bien que le recouvrement direct des triplets offre une première approximation de la similarité, il peut arriver que deux sous-graphes ne se recouvrent pas textuellement tout en représentant des concepts équivalents ou proches. Par exemple, deux entités peuvent utiliser des terminologies différentes pour désigner un même concept (par ex. "Flu" et "Influenza"). Pour traiter ce cas, il est nécessaire d'introduire des mécanismes d'**isomorphisme partiel**. L'idée est d'identifier un **maximum common subgraph (MCS)** entre \mathcal{G}_i et \mathcal{G}_j qui tient compte des synonymes et des relations de hiérarchie telles que « rdfs:subClassOf » ou « owl:equivalentClass ». On peut alors définir un score de similarité par le rapport :

$$S_{\text{MCS}}(i, j) = \frac{|\text{MCS}(\mathcal{G}_i, \mathcal{G}_j)|}{\max(|\mathcal{G}_i|, |\mathcal{G}_j|)}.$$

Une autre approche consiste à mesurer la **distance hiérarchique** entre les concepts correspondants. Par exemple, si deux classes se trouvent à des niveaux différents dans la hiérarchie ontologique, une fonction de distance $d(\mathcal{C}_i, \mathcal{C}_j)$ peut être utilisée pour ajuster la similarité, de sorte qu'un décalage important dans la hiérarchie diminue la valeur du score de similarité.

3. Combinaison en une Fonction de Synergie Ontologique

Pour intégrer de manière globale ces notions, on peut définir la **synergie ontologique** $\mathbf{S}_{\text{ont}}(i, j)$ comme une combinaison d'un terme de recouvrement et d'un terme ajusté par la distance ou par une évaluation plus fine de l'isomorphisme partiel. Par exemple, une formule possible est :

$$\mathbf{S}_{\text{ont}}(i, j) = \alpha S_{\text{ont}}(i, j) + \beta S_{\text{MCS}}(i, j),$$

où α et β sont des coefficients de pondération qui reflètent l'importance relative du recouvrement direct et de la similarité structurelle. Dans le cas où la distance hiérarchique est également prise en compte, on peut ajuster le score par une fonction décroissante de cette distance, $g(d)$, de sorte que :

$$\mathbf{S}_{\text{ont}}(i, j) = \alpha S_{\text{ont}}(i, j) + \beta S_{\text{MCS}}(i, j) \cdot g(d(\mathcal{G}_i, \mathcal{G}_j)).$$

Cette approche permet d'obtenir une mesure de synergie qui valorise à la fois la **co-occurrence** des triplets et la similarité structurelle au sein des sous-graphes ontologiques.

C. Application dans la Dynamique du DSL

Dans un **Synergistic Connection Network**, la synergie ontologique $\mathbf{S}_{\text{ont}}(i, j)$ ainsi définie peut être intégrée directement dans la règle de mise à jour des pondérations. La mise à jour s'exprime alors par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}_{\text{ont}}(i, j) - \tau \omega_{i,j}(t)],$$

où η représente le taux d'apprentissage et τ le coefficient de décroissance. Cette dynamique permet d'ajuster les liaisons entre entités en fonction de leur **similarité conceptuelle**. Lorsque deux entités partagent un grand nombre de triplets ou présentent une forte similitude structurelle dans leurs sous-graphes ontologiques, la pondération $\omega_{i,j}$ est renforcée, favorisant ainsi l'émergence de clusters d'entités ayant des liens sémantiques forts. À l'inverse, un faible recouvrement ou une grande distance hiérarchique conduira à une diminution de $\omega_{i,j}$.

D. Problèmes de Complexité et Approximations

Il est important de noter que le calcul précis de la similarité ontologique peut devenir très coûteux en termes de complexité algorithmique, notamment lorsqu'il s'agit de trouver le maximum common subgraph (MCS) ou d'évaluer des distances hiérarchiques dans des ontologies très expressives. En pratique, pour des systèmes à grande échelle, il est souvent nécessaire de recourir à des heuristiques ou à des méthodes d'**approximate graph matching** afin de rendre le calcul de $\mathbf{S}_{\text{ont}}(i, j)$ faisable dans un temps raisonnable. Des techniques d'**indexation sémantique** ou des

simplifications de l'ontologie (par exemple, en utilisant OWL Lite plutôt qu'OWL DL) peuvent également être employées pour réduire la charge computationnelle tout en conservant une précision acceptable.

Conclusion

La **mesure d'overlap conceptuel** entre deux sous-graphes ontologiques se fonde sur la quantification du recouvrement des triplets et la prise en compte d'une similarité structurelle, ajustée par la distance hiérarchique ou l'isomorphisme partiel. La synergie ontologique est ainsi définie par une fonction combinant ces aspects :

$$\mathbf{S}_{\text{ont}}(i, j) = \alpha \frac{|\mathcal{G}_i \cap \mathcal{G}_j|}{|\mathcal{G}_i \cup \mathcal{G}_j|} + \beta S_{\text{MCS}}(i, j) \cdot g(d(\mathcal{G}_i, \mathcal{G}_j)),$$

et intervient dans la dynamique d'auto-organisation du SCN via la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}_{\text{ont}}(i, j) - \tau \omega_{i,j}(t)].$$

Cette approche permet de renforcer les liens entre entités dont les représentations ontologiques se recouvrent fortement et de diminuer ceux présentant des divergences structurelles, aboutissant ainsi à des clusters sémantiquement cohérents. Les défis principaux résident dans la complexité des calculs d'inférence logique et dans la nécessité d'employer des méthodes approximatives pour assurer la **scalabilité** du système. Les sections **3.4.2** et **3.4.3** approfondiront ces compromis et présenteront des solutions hybrides pour intégrer ces mesures dans un cadre de Deep Synergy Learning.

3.4.2.3. Approches “Distance Sémantique” ou “Information Mutuelle Symbolique”

Dans le cadre du **Deep Synergy Learning (DSL)**, la capacité à mesurer la synergie entre deux entités symboliques repose non seulement sur la simple co-occurrence ou le recouvrement de règles, mais aussi sur des méthodes quantitatives permettant d'estimer de façon plus fine le degré de proximité conceptuelle. Deux approches complémentaires se distinguent ainsi : la **distance sémantique** et l'**information mutuelle symbolique**. Ces deux outils visent à quantifier la richesse informative partagée ou l'éloignement conceptuel entre les représentations symboliques des entités, et sont ensuite intégrés dans la dynamique d'auto-organisation du SCN (Synergistic Connection Network).

A. Approche par Distance Sémantique

Lorsqu'une entité \mathcal{E}_i est représentée par un sous-graphe ontologique \mathcal{G}_i , l'**ontologie** fournit une structure hiérarchique reliant des concepts par des relations telles que rdfs:subClassOf ou owl:equivalentClass. Pour évaluer la **proximité sémantique** entre deux entités \mathcal{E}_i et \mathcal{E}_j , on peut définir une fonction de **distance sémantique** $d_{\text{sem}}(\mathcal{E}_i, \mathcal{E}_j)$ qui exploite la structure hiérarchique de l'ontologie. Par exemple, en identifiant pour deux concepts C_i et C_j leur **plus bas ancêtre commun** (LCS, *lowest common subsumer*), on peut mesurer la distance en nombre d'arêtes parcourues pour atteindre ce LCS. Les travaux issus de WordNet ont introduit des mesures classiques telles que la **mesure de Resnik** et la **mesure de Lin**, définies respectivement par :

$$\text{sim}_{\text{Resnik}}(C_i, C_j) = \text{IC}(\text{lcs}(C_i, C_j)),$$

$$\text{sim}_{\text{Lin}}(C_i, C_j) = \frac{2 \text{IC}(\text{lcs}(C_i, C_j))}{\text{IC}(C_i) + \text{IC}(C_j)},$$

où $\text{IC}(C)$ représente le **contenu informationnel** du concept C , une mesure qui augmente avec la spécificité du concept. Dans le contexte du DSL, on peut associer à chaque entité \mathcal{E}_i un ou plusieurs concepts dominants extraits de son sous-graphe ontologique et définir la synergie sémantique entre \mathcal{E}_i et \mathcal{E}_j par :

$$\mathbf{S}_{\text{sem}}(i, j) = \text{sim}_{\text{Lin}}(C_i, C_j) \quad \text{ou} \quad \text{sim}_{\text{Resnik}}(C_i, C_j),$$

selon le choix de la mesure, de sorte que deux entités seront considérées comme sémantiquement proches lorsque leurs concepts associés présentent un fort degré d'information partagée. Une autre variante consiste à appliquer une transformation exponentielle afin de convertir la distance sémantique en une similitude, par exemple :

$$\mathbf{S}_{\text{sem}}(i, j) = \exp(-\alpha d_{\text{sem}}(\mathcal{E}_i, \mathcal{E}_j)),$$

où $\alpha > 0$ est un paramètre de réglage. Ce type d'approche permet une tolérance aux variations et aux synonymes, en tirant parti de la structure hiérarchique de l'ontologie.

B. Approche par Information Mutuelle Symbolique

Une alternative quant à elle consiste à adapter le concept d'**information mutuelle (MI)**, largement utilisé en théorie de l'information pour mesurer la dépendance entre deux variables aléatoires, à un contexte symbolique. Si l'on considère qu'un ensemble de règles ou d'axiomes associé à une entité \mathcal{E}_i peut être représenté par un vecteur binaire $\mathbf{x}_i \in \{0,1\}^m$ — où chaque composante $x_i(k) = 1$ indique la présence de l'axiome r_k dans l'ensemble R_i de \mathcal{E}_i — alors on peut définir des variables aléatoires X_i et X_j pour les entités \mathcal{E}_i et \mathcal{E}_j . L'**information mutuelle** entre X_i et X_j se définit par :

$$I(X_i; X_j) = \sum_{\mathbf{x} \in \{0,1\}^m} \sum_{\mathbf{y} \in \{0,1\}^m} p(\mathbf{x}, \mathbf{y}) \log\left(\frac{p(\mathbf{x}, \mathbf{y})}{p(\mathbf{x}) p(\mathbf{y})}\right),$$

où $p(\mathbf{x})$ et $p(\mathbf{y})$ sont les distributions marginales et $p(\mathbf{x}, \mathbf{y})$ la distribution conjointe. Une valeur élevée de $I(X_i; X_j)$ signifie qu'une connaissance du vecteur binaire associé à \mathcal{E}_i réduit significativement l'incertitude concernant celui de \mathcal{E}_j , indiquant ainsi une forte **synergie symbolique**. Afin de normaliser ce score dans l'intervalle $[0,1]$, on peut utiliser la formule :

$$\mathbf{S}_{\text{MI}}(i, j) = \frac{I(X_i; X_j)}{\max\{H(X_i), H(X_j)\}},$$

où $H(X)$ représente l'**entropie** de la variable X . Ce score permet de capturer la part d'information partagée entre deux ensembles d'axiomes, offrant une mesure plus nuancée que le simple recouvrement.

C. Intégration dans la Dynamique du DSL

Dans un **Deep Synergy Learning (DSL)**, la synergie entre deux entités symboliques est utilisée pour adapter la pondération de leur connexion. Ainsi, la mise à jour de la pondération $\omega_{i,j}$ s'effectue généralement selon la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}(i,j) - \tau \omega_{i,j}(t)],$$

où $\mathbf{S}(i,j)$ peut être défini soit comme $\mathbf{S}_{\text{sem}}(i,j)$ dans le cas d'une approche par distance sémantique, soit comme $\mathbf{S}_{\text{MI}}(i,j)$ dans le cadre de l'information mutuelle symbolique, ou une combinaison des deux. Par exemple, on peut poser :

$$\mathbf{S}(i,j) = \alpha \mathbf{S}_{\text{sem}}(i,j) + \beta \mathbf{S}_{\text{MI}}(i,j),$$

où α et β sont des coefficients d'ajustement qui déterminent l'importance relative de chaque méthode dans le calcul global de la synergie. Cette formulation permet au **SCN** de renforcer les liens entre entités dont les sous-graphes ontologiques ou les ensembles d'axiomes présentent une forte similarité sémantique et une information mutuelle élevée, tout en affaiblissant les connexions entre entités peu compatibles.

D. Problèmes de Complexité et Méthodes Approximatives

Il est essentiel de souligner que le calcul exact de la **similarité ontologique** peut s'avérer extrêmement coûteux en termes de complexité algorithmique, surtout lorsqu'il s'agit de déterminer le **maximum common subgraph (MCS)** ou d'évaluer précisément la distance hiérarchique entre concepts dans une ontologie expressive. Pour pallier ce problème, plusieurs stratégies d'approximation peuvent être mises en œuvre :

- **Simplification de l'ontologie** : en réduisant l'expressivité (par exemple en utilisant OWL Lite ou RDF basique), la complexité des inférences logiques peut être significativement diminuée.
- **Utilisation d'index sémantiques** : le recours à des structures de données de type hash ou à des index pré-calculés peut accélérer la détection des correspondances ou des conflits entre axiomes.
- **Heuristiques d'approximation** : des algorithmes d'approximate graph matching permettent d'obtenir des estimations de similarité en temps raisonnable, sans garantir l'exactitude du MCS.

Ces méthodes permettent de rendre l'intégration de la synergie ontologique dans la dynamique du DSL plus **scalable**, même si elles introduisent un compromis entre précision et efficacité.

Conclusion

La **mesure d'overlap conceptuel** entre deux sous-graphes ontologiques s'appuie sur des approches complémentaires telles que la **distance sémantique** et l'**information mutuelle symbolique**. La première exploite la structure hiérarchique des concepts pour évaluer la proximité entre deux entités, tandis que la seconde quantifie la part d'information partagée entre leurs

ensembles d'axiomes. Ces deux approches, qui peuvent être combinées de manière linéaire pour former un score global

$$\mathbf{S}_{\text{ont}}(i, j) = \alpha \frac{|\mathcal{G}_i \cap \mathcal{G}_j|}{|\mathcal{G}_i \cup \mathcal{G}_j|} + \beta \mathbf{S}_{\text{MI}}(i, j),$$

s'intègrent dans la règle de mise à jour du DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}_{\text{ont}}(i, j) - \tau \omega_{i,j}(t)],$$

permettant ainsi d'ajuster dynamiquement la force des connexions en fonction de la similarité sémantique. Malgré les coûts de calcul inhérents à ces méthodes d'inférence, l'usage de techniques d'approximation et d'indexation offre une voie réaliste pour appliquer ces concepts à grande échelle. Ces mesures enrichissent le **SCN** en lui conférant une dimension explicative et robuste, essentielle pour des applications où la cohérence sémantique joue un rôle crucial, notamment dans des domaines tels que la médecine, l'ingénierie ou la recherche scientifique. Les sections **3.4.2** et **3.4.3** aborderont en détail les compromis et les solutions hybrides permettant d'optimiser ces approches dans le cadre du Deep Synergy Learning.

3.4.3. Évolution Symbolique et Gestes des Contradictions

Les systèmes symboliques, bien qu'offrant une **interprétabilité** et un **raisonnement formel** (sections 3.4.1 et 3.4.2), n'échappent pas à la nécessité d'**évoluer** au fil du temps. Au même titre que les embeddings sub-symboliques (chap. 3.3.4), un bloc de connaissances symboliques doit pouvoir accueillir de **nouvelles** règles, en supprimer d'obsoletes ou de contradictoires, et gérer les conflits potentiels qui naissent de ces changements. Cette section (3.4.3) examine comment on peut **insérer** ou **supprimer** des règles (3.4.3.1), comment **détecter** et traiter les **contradictions** (3.4.3.2), et donne un **exemple** d'évolution continue d'un ensemble de postulats (3.4.3.3).

3.4.3.1. Insertion / Suppression de Règles (Chap. 9, Flux Continu)

Dans un contexte de **Deep Synergy Learning (DSL)** à composante symbolique, chaque entité \mathcal{E}_i est caractérisée par un ensemble de règles ou d'axiomes, noté R_i . Ces ensembles R_i incarnent la base de connaissances qui sous-tend la représentation symbolique d'une entité. Lorsque de nouveaux faits sont découverts ou que certaines règles deviennent obsolètes, il est alors nécessaire de modifier ces blocs R_i par l'insertion ou la suppression de postulats. Cette opération de mise à jour dynamique s'inscrit dans la **dynamique** d'un **Synergistic Connection Network (SCN)** et influence directement la **synergie symbolique** entre entités ainsi que la formation des **clusters** qui en résultent.

A. Insertion de Règles : Actualisation et Cohérence

L'insertion de nouvelles règles intervient lorsque l'on souhaite **étendre** ou **affiner** la base de connaissances associée à une entité \mathcal{E}_i . Formellement, si l'on considère qu'à l'instant t l'ensemble de règles associé à \mathcal{E}_i est $R_i(t)$, l'ajout d'un nouveau postulat p se traduit par la mise à jour :

$$R_i(t+1) = R_i(t) \cup \{p\}.$$

Par exemple, dans un contexte médical, l'introduction de la règle « if Toux and OdeurPerdue then CovidSuspect » permet d'actualiser le diagnostic en fonction des nouvelles observations. De même, en ingénierie, l'insertion d'une règle telle que « if PièceX subtend AxisY then isRotational » formalise une relation d'articulation nouvellement identifiée.

Dès l'insertion du postulat p , la **synergie** entre l'entité \mathcal{E}_i et une autre entité \mathcal{E}_j , notée $S(i,j)$, peut être affectée. Si p renforce la compatibilité entre R_i et R_j – par exemple, en augmentant le recouvrement entre les ensembles de règles – alors la pondération $\omega_{i,j}$ se voit augmenter conformément à la dynamique d'auto-organisation donnée par la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Dans ce contexte, $S(i,j)$ pourra être recalculé, par exemple en utilisant une fonction symbolique $S_{\text{sym}}(i,j)$ qui intègre le nouvel axiome. À l'inverse, si l'ajout de p engendre une contradiction avec un axiome existant dans R_j , le score de synergie diminuera, ce qui se traduira par une baisse de $\omega_{i,j}$.

Dans un **flux continu** tel que développé au Chapitre 9, ces opérations d'insertion se produisent fréquemment et doivent être intégrées de façon à ce que la dynamique du SCN puisse s'ajuster en temps réel. Ainsi, chaque modification dans R_i entraîne un réexamen partiel – voire complet – des pondérations $\omega_{i,j}$ associées, afin de refléter l'impact des nouvelles règles sur la **compatibilité** entre entités.

B. Suppression de Règles : Réorganisation Logique

Le retrait de règles constitue l'autre facette de la mise à jour de la base de connaissances dans un DSL symbolique. Lorsque certains axiomes deviennent obsolètes ou sont invalidés par de nouvelles données, on procède à la suppression de ces éléments dans R_i . Cette opération s'exprime par :

$$R_i(t+1) = R_i(t) \setminus \{q\},$$

où q représente le postulat à retirer. Par exemple, dans le domaine médical, la règle « if Toux then Infectieux » pourra être supprimée si elle est jugée trop générale ou contredite par de nouveaux diagnostics. De même, dans un contexte industriel, l'ancienne contrainte « chargeMax = 500\,N » pourrait être écartée au profit d'une nouvelle norme.

La suppression de q affecte également la synergie $S(i,j)$ entre \mathcal{E}_i et d'autres entités \mathcal{E}_j . Si q constituait un point de recouvrement important avec R_j , sa disparition diminuera la valeur de l'intersection $R_i \cap R_j$, et par conséquent, le score de similarité. Inversement, si q était à l'origine source de contradiction avec R_j , son retrait peut améliorer la compatibilité, entraînant une augmentation de $\omega_{i,j}$.

C. Problème de l'Échelle : Révisions en $\mathcal{O}(n)$ et Gestion de la Charge

Dans un SCN comportant un grand nombre d'entités, chaque opération d'insertion ou de suppression sur R_i peut théoriquement nécessiter une réévaluation de la synergie $S(i,j)$ pour chaque entité \mathcal{E}_j du réseau, ce qui représente un coût de l'ordre de $\mathcal{O}(n)$ par modification. Lorsque

de telles mises à jour se produisent simultanément ou de manière fréquente, la charge de recalcul peut devenir prohibitif. Pour atténuer ce problème, deux stratégies d'optimisation peuvent être mises en œuvre :

- **Index sémantique** : En associant à chaque règle ou concept une structure de données indiquant l'ensemble des entités qui l'utilisent, il est possible de restreindre la réévaluation aux seules entités affectées par la modification. Ainsi, l'insertion ou la suppression d'un axiome p dans R_i ne nécessite de recalculer $S(i, j)$ que pour les entités j qui partagent p .
- **Approche paresseuse (lazy evaluation)** : Au lieu de recalculer immédiatement la synergie pour toutes les paires affectées, la mise à jour est différée et réalisée uniquement lorsque la synergie est requise pour une opération spécifique, réduisant ainsi la fréquence des mises à jour globales.

D. Dynamique et Versionnage

Afin de gérer efficacement l'évolution des ensembles de règles, il est crucial de mettre en place un mécanisme de **versionnage**. Ce mécanisme permet de tracer l'historique des modifications et de gérer les mises à jour de manière structurée. On définit ainsi :

$$R_i^{(t+1)} = (R_i^{(t)} \cup R_{\text{add}}(t)) \setminus R_{\text{del}}(t),$$

où $R_{\text{add}}(t)$ et $R_{\text{del}}(t)$ représentent respectivement les ensembles de règles ajoutées et supprimées à l'itération t . Ce versionnage offre plusieurs avantages : il permet de conserver un historique des états $\{R_i^{(0)}, R_i^{(1)}, \dots\}$, de définir des points de contrôle pour revenir à une version antérieure en cas de conflit majeur, et d'analyser l'évolution de la complexité des blocs R_i au fil du temps. Ainsi, le suivi des versions facilite la gestion des contradictions et offre une base pour des mécanismes de rollback si nécessaire.

E. Risques d'Oscillation et de Chaos Symbolique

Un système DSL symbolique, qui évolue en flux continu, peut être sujet à des phénomènes d'**oscillation** ou même de **chaos logique** si les mises à jour des règles se produisent de manière trop rapide ou contradictoire. Par exemple, si une entité \mathcal{E}_i insère successivement un axiome $A \Rightarrow B$ qui contredit les règles d'une entité \mathcal{E}_j et que, peu de temps après, ce même axiome est supprimé pour rétablir la compatibilité, la pondération $\omega_{i,j}$ peut fluctuer de manière excessive. De même, un changement global simultané dans plusieurs blocs R_i peut provoquer un « big bang » dans le SCN, où un grand nombre de synergies doivent être recalculées, ce qui pourrait engendrer des cycles d'instabilité. Pour pallier ces risques, il est souvent nécessaire de regrouper les modifications dans des phases de mise à jour (batch updates) afin de permettre à la dynamique $\omega_{i,j}$ de converger partiellement entre deux vagues de changements.

Conclusion

L'insertion et la suppression de règles dans un **DSL symbolique** constituent des opérations essentielles permettant l'actualisation et l'affinement de la base de connaissances associée à chaque entité \mathcal{E}_i . La mise à jour de R_i se traduit par l'équation

$$R_i(t + 1) = (R_i(t) \cup R_{\text{add}}(t)) \setminus R_{\text{del}}(t),$$

et entraîne une réévaluation des synergies $S(i,j)$ et, par conséquent, des pondérations $\omega_{i,j}$ dans le réseau. Bien que ces opérations permettent d'adapter le SCN aux nouvelles connaissances ou à l'obsolescence de certains axiomes, elles impliquent également un coût algorithmique élevé, particulièrement à grande échelle, ainsi qu'un risque d'oscillation ou de chaos symbolique si les mises à jour ne sont pas régulées. Pour surmonter ces obstacles, il est impératif d'implémenter des stratégies d'optimisation telles que l'indexation sémantique, l'évaluation paresseuse, ainsi qu'un mécanisme de versionnage robuste. Ces stratégies permettent de limiter les révisions globales, d'assurer la cohérence logique du système, et de contrôler le risque d'instabilité lors des mises à jour en flux continu. Dans le Chapitre 9, nous approfondirons ces stratégies et heuristiques pour gérer l'apprentissage continu et la **plasticité** des règles, en combinant la **souplesse** auto-organisée du DSL avec les contraintes strictes d'un système symbolique.

3.4.3.2. Détection de Contradiction, Feedback Top-Down pour Forcer un Réagencement (Chap. 10)

Dans un système de **Deep Synergy Learning (DSL)** à composante symbolique, chaque entité \mathcal{E}_i est décrite par un ensemble de règles ou d'axiomes R_i . La dynamique de mise à jour de ces blocs, comme présentée en section 3.4.3.1, permet d'insérer ou de supprimer des postulats afin d'actualiser la base de connaissances. Toutefois, lorsque ces modifications conduisent à l'introduction de contradictions – c'est-à-dire, lorsque l'union $R_i \cup R_j$ de deux ensembles de règles permet de déduire simultanément un énoncé et son contraire – la **synergie symbolique** $S(i,j)$ entre les entités concernées doit être réévaluée et, souvent, fortement pénalisée. La présente section se propose d'expliquer de manière rigoureuse comment détecter ces contradictions, de quelle manière elles impactent la dynamique du SCN, et comment un mécanisme de **feedback top-down** peut être introduit pour forcer un réagencement global et éviter des oscillations ou un chaos logique.

A. Détection de Contradiction : Fondements et Méthodes

La contradiction entre deux ensembles de règles R_i et R_j se produit lorsqu'ils, pris ensemble, engendrent une incohérence formelle, c'est-à-dire lorsque :

$$R_i \cup R_j \vdash \perp,$$

où le symbole \perp représente l'absurdité ou la fausseté. Pour quantifier cette contradiction, on définit une fonction binaire de contradiction de la manière suivante :

$$\text{Contradiction}(R_i, R_j) = \begin{cases} 1, & \text{si } \exists a \in R_i, b \in R_j \text{ tels que } a \wedge b \vdash \perp, \\ 0, & \text{sinon.} \end{cases}$$

Cette définition peut être raffinée en comptant le nombre de conflits effectifs entre les deux ensembles, en posant :

$$\text{ContradictionCount}(R_i, R_j) = \sum_{(a,b) \in R_i \times R_j} \mathbf{1}[a \wedge b \vdash \perp],$$

où $\mathbf{1}[\cdot]$ désigne la fonction indicatrice qui vaut 1 si l'argument est vrai et 0 sinon. Ce comptage permet d'obtenir une mesure plus graduelle de l'incohérence entre R_i et R_j , laquelle pourra être intégrée dans la fonction de synergie symbolique.

En effet, si l'on considère une fonction de synergie symbolique qui combine à la fois les aspects de recouvrement et de contradiction, on peut écrire :

$$S_{\text{sym}}(i, j) = \alpha \frac{|R_i \cap R_j|}{|R_i \cup R_j|} - \beta \text{ContradictionCount}(R_i, R_j),$$

où $\alpha, \beta > 0$ sont des constantes pondératrices qui ajustent l'importance relative du recouvrement par rapport aux contradictions. Ainsi, une forte contradiction (c'est-à-dire une grande valeur de $\text{ContradictionCount}(R_i, R_j)$) fera tendre $S_{\text{sym}}(i, j)$ vers zéro, ce qui, dans la dynamique du DSL, entraînera une diminution de la pondération $\omega_{i,j}$.

B. Impact sur la Dynamique du SCN et Mise à Jour des Pondérations

La règle de mise à jour du SCN, dans sa version symbolique, est donnée par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{sym}}(i, j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le taux d'apprentissage et $\tau > 0$ le coefficient de décroissance. Dès qu'une contradiction est détectée, c'est-à-dire dès que $\text{Contradiction}(R_i, R_j) = 1$ ou que $\text{ContradictionCount}(R_i, R_j)$ est élevé, le score $S_{\text{sym}}(i, j)$ est pénalisé. Cela se traduit par une réduction immédiate de la pondération $\omega_{i,j}$, isolant ainsi les entités dont les bases de connaissances sont incohérentes.

Si ces contradictions apparaissent de manière isolée, la diminution locale de $\omega_{i,j}$ peut suffire à empêcher la formation de clusters incohérents. Cependant, dans un système à grande échelle, de nombreux conflits peuvent émerger simultanément, engendrant ainsi un risque de **chaos** local ou d'oscillations persistantes dans la dynamique de mise à jour.

C. Mécanisme de Feedback Top-Down pour Forcer le Réagencement

Pour pallier le risque d'instabilité engendré par un excès de contradictions, il est nécessaire d'introduire un **feedback top-down**. Ce mécanisme agit comme un **régulateur global** qui surveille la cohérence du système et intervient lorsque le nombre de contradictions dépasse un certain seuil critique.

On peut définir une métrique globale de contradiction pour l'ensemble du SCN :

$$\text{ContradictCount} = \sum_{i,j} \text{ContradictionCount}(R_i, R_j).$$

Si cette valeur excède un seuil prédéfini T , alors le mécanisme de feedback top-down se déclenche. Ce module supérieur peut alors entreprendre plusieurs actions pour restaurer l'ordre :

- **Revue et Révision des Règles** : Le système identifie les entités E_i présentant le plus grand nombre de conflits (par exemple, celles pour lesquelles $\text{ContradictionCount}(R_i) =$

$\sum_j \text{ContradictionCount}(R_i, R_j)$ est élevé) et demande une révision de leurs ensembles R_i . Cela peut impliquer la suppression ou la modification des axiomes responsables des conflits.

- **Forçage d'un Réagencement** : En imposant une baisse significative de la pondération pour les liens conflictuels, c'est-à-dire en réglant $\omega_{i,j}$ à une valeur faible ou nulle pour les paires en contradiction, le système peut isoler les entités problématiques, forçant ainsi une réorganisation des clusters.
- **Re-Clustering Global** : Dans des cas extrêmes, le méta-module peut déclencher une phase de re-clusterisation, regroupant de manière forcée les entités compatibles et séparant celles qui présentent des incohérences majeures.

Ce mécanisme de feedback top-down permet de sortir d'un régime où des modifications locales répétées – par insertion et suppression successives de règles – conduiraient à des oscillations ou à un chaos logique, et assure ainsi une **stabilité** globale du SCN.

Conclusion

La détection des contradictions logiques dans un DSL symbolique est cruciale pour maintenir la **cohérence** et la **stabilité** des synergies entre entités. Nous avons défini formellement la notion de contradiction à travers la relation

$$R_i \cup R_j \vdash \perp,$$

et avons introduit des mesures telles que $\text{ContradictionCount}(R_i, R_j)$ pour quantifier le nombre de conflits entre les ensembles de règles. En intégrant cette mesure dans la fonction de synergie symbolique

$$S_{\text{sym}}(i, j) = \alpha \frac{|R_i \cap R_j|}{|R_i \cup R_j|} - \beta \text{ContradictionCount}(R_i, R_j),$$

le SCN est amené à ajuster la pondération $\omega_{i,j}$ selon la compatibilité logique entre les entités. Néanmoins, lorsque de nombreuses contradictions apparaissent, le système risque de se désorganiser localement. Pour contrer ce phénomène, un mécanisme de **feedback top-down** (tel que détaillé dans le Chapitre 10) intervient pour réorganiser globalement les règles, soit en imposant la révision des axiomes conflictuels, soit en isolant les entités problématiques, soit en déclenchant une phase de re-clusterisation. Ce processus garantit que, malgré les mises à jour locales potentiellement contradictoires, le SCN demeure globalement cohérent et converge vers une structure de liens stable et explicable.

3.4.3.3. Exemple : un DSL stockant un “Ensemble de Postulats” qui se Modifie au Fil du Temps

Dans un **Deep Synergy Learning (DSL)** intégrant une composante symbolique, chaque entité \mathcal{E}_i est caractérisée par un ensemble de postulats ou d'axiomes, noté R_i . Ces postulats représentent la connaissance explicite associée à l'entité et, en fonction des évolutions du domaine ou de l'arrivée

de nouvelles données, R_i peut évoluer de manière dynamique par l'insertion de nouveaux postulats ou la suppression d'anciens. Cette capacité à modifier le contenu de R_i en flux continu est essentielle pour que le **Synergistic Connection Network (SCN)** puisse adapter ses liaisons, c'est-à-dire ses pondérations $\omega_{i,j}$, en temps réel.

A. Contexte : Entités "Patients" avec Blocs de Connaissances Médicales

Considérons un système médical où chaque entité \mathcal{E}_i correspond à un patient et est associé à un bloc de connaissances R_i qui regroupe à la fois les observations cliniques et les règles diagnostiques pertinentes. Par exemple, pour un patient donné, R_i peut contenir des assertions telles que :

"Patient \mathcal{E}_i présente Toux"

"Patient \mathcal{E}_i a Fièvre(39°C)"

$R_i \ni \{\text{if Toux} \wedge \text{Fièvre then Infectieux}\}$

$R_i \ni \{\text{if Vacciné(Grippe) then RisqueRéduit}\}$

Ces ensembles R_i forment la base symbolique qui sera utilisée pour calculer la **synergie symbolique** entre les patients. En effet, deux patients \mathcal{E}_i et \mathcal{E}_j dont les ensembles de postulats présentent un fort recouvrement (c'est-à-dire, un grand nombre d'axiomes communs ou compatibles) auront une synergie $\mathbf{S}(i,j)$ élevée, traduite par une pondération $\omega_{i,j}$ renforcée dans le SCN. À l'inverse, une faible similarité ou la présence de contradictions entraînera une diminution de $\omega_{i,j}$.

B. Ajout de Règles : Extension du Bloc de Connaissances

Au fil du temps, de nouvelles découvertes ou mises à jour dans le domaine médical peuvent conduire à l'ajout de nouveaux postulats dans le bloc de connaissances d'un patient. Cette opération d'insertion se formalise par :

$$R_i(t+1) = R_i(t) \cup \{p\},$$

où p représente le nouvel axiome, par exemple :

$p:\text{if PerteOdorat} \wedge \text{Fièvre then CovidSuspect}.$

L'insertion de p peut modifier la synergie symbolique entre le patient \mathcal{E}_i et un autre patient \mathcal{E}_j . Si, par exemple, R_j contient déjà une règle similaire ou compatible – telle que “if PerteOdorat \wedge Fièvre then InfectionVirale” – le recouvrement entre R_i et R_j augmente, ce qui se traduit par une augmentation de la synergie, notée $\mathbf{S}_{\text{sym}}(i,j)$. Ainsi, la mise à jour de la pondération dans le SCN s'effectue selon la formule habituelle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[\mathbf{S}_{\text{sym}}(i,j) - \tau \omega_{i,j}(t)].$$

En revanche, si l'axiome p entre en contradiction avec des règles déjà présentes dans R_j , par exemple si R_j contient « if Fièvre \wedge PerteOdorat then NonInfectieux », alors la fonction de synergie se verra pénalisée, et $\omega_{i,j}$ diminuera en conséquence.

C. Suppression ou Révision de Règles : Retrait de Postulats

La suppression de règles intervient lorsque certains axiomes se révèlent obsolètes ou erronés. On définit alors l'opération de suppression par :

$$R_i(t+1) = R_i(t) \setminus \{q\},$$

où q est l'axiome retiré, par exemple une règle trop générale comme :

q:if Toux then Infectieux.

Le retrait de q peut modifier la synergie de deux entités. Si q contribuait à une co-occurrence importante entre R_i et R_j , sa suppression pourrait entraîner une diminution de $\mathbf{S}_{\text{sym}}(i, j)$ et, par la suite, une baisse de la pondération $\omega_{i,j}$. Inversement, si q était à l'origine d'une contradiction avec R_j , son retrait pourrait améliorer la compatibilité et renforcer $\omega_{i,j}$. La dynamique de mise à jour reste donc régie par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[\mathbf{S}_{\text{sym}}(R_i(t+1), R_j(t+1)) - \tau \omega_{i,j}(t) \right].$$

Cette équation illustre que toute modification dans le bloc R_i nécessite une réévaluation immédiate de la synergie avec les autres entités, afin de répercuter l'évolution de la base de connaissances dans la structure du SCN.

D. Gestion Continue et Versionnage

Dans un environnement en flux continu, il est indispensable de suivre l'évolution des blocs de règles afin de pouvoir revenir sur des changements si nécessaire et de garantir la cohérence du système. Pour ce faire, on introduit un mécanisme de **versionnage** :

$$R_i^{(t+1)} = (R_i^{(t)} \cup R_{\text{add}}^{(i,t)}) \setminus R_{\text{del}}^{(i,t)},$$

où $R_{\text{add}}^{(i,t)}$ représente l'ensemble des règles ajoutées à l'itération t et $R_{\text{del}}^{(i,t)}$ celles supprimées. Ce mécanisme permet de conserver un historique des états $\{R_i^{(0)}, R_i^{(1)}, \dots\}$, facilitant ainsi l'analyse rétroactive des décisions prises et, en cas de conflits majeurs, la possibilité d'un rollback. Un tel versionnage s'avère essentiel pour maintenir la cohérence globale et pour minimiser le coût de recalcul de la synergie dans de grands réseaux.

E. Risques d'Oscillation et Feedback Top-Down

Si les modifications des blocs R_i se produisent trop fréquemment ou de manière contradictoire, le SCN risque d'entrer dans une phase d'oscillations ou de chaos symbolique. Par exemple, une entité \mathcal{E}_i pourrait insérer un axiome qui augmente temporairement la synergie avec \mathcal{E}_j , puis le retirer peu après, entraînant ainsi une fluctuation répétée de $\omega_{i,j}$. Pour éviter de telles instabilités, un **feedback top-down** peut intervenir (voir section 3.4.3.2). Ce mécanisme de rétroaction consiste en un module de contrôle global qui surveille l'état du SCN et, si le nombre de contradictions ou le taux de variation des pondérations dépasse un seuil critique, il impose une révision ou une réorganisation forcée des règles. Par exemple, on peut définir une condition de déclenchement :

si $\text{ContradictionCount}(t) > \theta$, alors initier une révision massive.

Une telle intervention permet d'isoler ou de corriger les entités présentant des incohérences, assurant ainsi la convergence de la dynamique et la formation de clusters stables.

Conclusion

L'exemple d'un **DSL** stockant un ensemble de postulats, qui se modifie au fil du temps, illustre de manière concrète comment la dynamique symbolique peut s'inscrire dans un flux continu d'actualisation des connaissances. Chaque entité \mathcal{E}_i est associée à un bloc de règles R_i qui évolue par des opérations d'insertion et de suppression, modifiant ainsi la synergie symbolique $\mathbf{S}_{\text{sym}}(i, j)$ entre les entités. La mise à jour des pondérations se fait selon la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}_{\text{sym}}(i, j, t+1) - \tau \omega_{i,j}(t)],$$

ce qui permet au SCN de s'ajuster en temps réel aux évolutions de la base de connaissances. Toutefois, à grande échelle et en présence de modifications fréquentes, des mécanismes d'optimisation (tels que l'indexation sémantique et les mises à jour paresseuses) ainsi qu'un système de versionnage deviennent indispensables pour assurer la cohérence globale. Par ailleurs, le recours à un feedback top-down est souvent nécessaire pour prévenir les oscillations ou le chaos symbolique, garantissant ainsi la stabilité du système. En définitive, ce scénario démontre comment un DSL peut intégrer et exploiter l'évolution continue des connaissances symboliques pour adapter sa structure d'interactions, tout en maintenant une cohérence logique et une organisation auto-organisée du réseau.

3.5. Représentations Hybrides (Sub-Symbolique + Symbolique)

Les sections précédentes ont montré les **forces** et **faiblesses** respectives des représentations **sub-symboliques** (chap. 3.3) et **symboliques** (chap. 3.4). Dans la pratique, un **DSL** (Deep Synergy Learning) peut grandement tirer profit d'une **fusion** des deux approches — c'est-à-dire que chaque entité \mathcal{E}_i combine à la fois un **embedding** (capable de manipuler le bruit, les données massives, etc.) et un **bloc** de règles ou d'axiomes (garantissant l'interprétabilité, la cohérence logique). C'est ce qu'on appelle une **représentation hybride** ou **neuro-symbolique**, où l'on cherche à marier la **puissance statistique** et la **lisibilité** sémantique.

Dans la section 3.5, nous verrons la **motivation** (3.5.1) qui pousse à mélanger sub-symbolique et symbolique, puis nous discuterons (3.5.2) des **stratégies** concrètes pour concevoir un tel assemblage dans un DSL (comment définir $r(i)$ comme un couple ou deux nœuds, comment modular la synergie mixte, etc.). Enfin, nous conclurons (3.5.3) sur les **avantages** (explicabilité, adaptabilité) et les **défis** (complexité, gestion des contradictions), en prenant un exemple d'**agent conversationnel** où la composante logique et l'embedding linguistique cohabitent.

3.5.1. Motivation de la Fusion

L'idée d'**hybrider** le sub-symbolique (vecteurs, embeddings neuronaux) et le symbolique (règles, ontologies) découle du constat que ni l'un ni l'autre ne suffisent, à eux seuls, à couvrir tous les besoins du DSL (Deep Synergy Learning). Les embeddings gèrent bien le **bruit** et la **variabilité**, mais souffrent d'un manque d'**interprétabilité** et ne fournissent pas de raisonnement logique direct ; les représentations symboliques, elles, sont **rigides** et intransigeantes face à l'ambiguïté ou l'imprécision, mais elles offrent un cadre formel pour **expliquer** et **inférer**.

3.5.1.1 : BÉNÉFICIER DE LA PUISSANCE STATISTIQUE DES EMBEDDINGS ET DE LA CLARTÉ DES RÈGLES LOGIQUES

Dans le cadre d'un **Deep Synergy Learning (DSL)**, l'intégration d'**embeddings sub-symboliques** et de blocs de **règles logiques** offre un paradigme hybride capable de conjuguer la robustesse statistique des approches d'apprentissage profond et la transparence inhérente aux systèmes symboliques. Ce modèle hybride tire profit de la **puissance** des techniques de projection vectorielle tout en fournissant un **raisonnement explicite** qui facilite l'interprétation des résultats dans des domaines critiques tels que la médecine ou la surveillance industrielle.

A. Dimension Sub-symbolique et Extraction d'Embeddings

Chaque entité \mathcal{E}_i est associée à un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$ qui encode de manière dense ses attributs discriminants. Ces représentations issues d'architectures telles que les **CNN**, **Transformers** ou **autoencodeurs** permettent d'exploiter la **puissance statistique** des modèles d'apprentissage profond pour capturer les structures complexes dans des données massives et bruitées. Par exemple, la proximité entre deux entités peut être mesurée à l'aide de la **distance euclidienne** :

$$d_{\text{eucl}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|,$$

ou à l'aide d'un **noyau Gaussien** défini par

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2),$$

où $\gamma > 0$ ajuste la sensibilité de la mesure. Ces formules illustrent la capacité des **embeddings** à tolérer un **bruit** modéré et à gérer un flux massif d'instances, en extrayant automatiquement des caractéristiques discriminantes pertinentes.

B. Dimension Symbolique et Règles Logiques

En complément de la représentation sub-symbolique, le DSL dote chaque entité \mathcal{E}_i d'un **bloc** de règles ou d'axiomes, noté $\{R_i\}$. Cette dimension symbolique permet d'initier un **raisonnement logique** explicite sur les entités. Les règles logiques offrent plusieurs avantages essentiels :

- Elles permettent d'identifier des **contradictions** ou des incohérences dans l'ensemble des connaissances.
- Elles facilitent l'**inférence explicite**, par exemple, en déduisant qu'« si A et B sont vraies, alors C l'est également ».
- Elles confèrent une **träçabilité** qui rend les décisions du système plus compréhensibles pour des experts dans des domaines où la **justification** des recommandations est indispensable.

Ainsi, dans des domaines tels que le **médical**, le bloc de règles peut encapsuler des protocoles diagnostiques (par exemple, « si **Toux**, **Fièvre** et saturation en oxygène $O_2 < 92\%$, alors **Admission Urgente** »), offrant ainsi un **langage intelligible** aux experts.

C. Fusion Hybride : La Synergie Globale

Pour exploiter au mieux ces deux dimensions complémentaires, le DSL hybride définit la **synergie globale** entre deux entités \mathcal{E}_i et \mathcal{E}_j comme un **mélange** pondéré des similarités sub-symboliques et symboliques. Cette synergie est formalisée par l'équation suivante :

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où $\alpha \in [0,1]$ est un paramètre de **pondération** qui permet de calibrer l'influence relative de chaque composante. La fonction S_{sub} évalue la **similarité vectorielle** (par exemple, en utilisant la formule $\exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2)$) ou le **cosinus normalisé**, tandis que S_{sym} mesure la **compatibilité symbolique** entre les ensembles de règles $\{R_i\}$ et $\{R_j\}$ en tenant compte de critères tels que la **co-occurrence** des axiomes, l'absence de contradiction ou encore la **distance ontologique** entre les concepts.

Cette approche hybride permet d'obtenir une mesure de **synergie** qui bénéficie simultanément de la **robustesse statistique** et de la **clarté logique**, assurant ainsi que les entités se rapprochent dans le réseau non seulement lorsqu'elles sont proches sur le plan des **embeddings** (c.-à-d. lorsque $\mathbf{x}_i \approx \mathbf{x}_j$) mais également lorsqu'elles partagent des **règles logiques compatibles**.

D. Mise à Jour des Pondérations et Auto-Organisation

L'architecture du DSL utilise la synergie hybride dans la mise à jour dynamique des pondérations entre entités. La règle de mise à jour s'exprime par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hybrid}}(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ représente le **taux d'apprentissage** et $\tau > 0$ est un **coefficent de décroissance** qui assure la stabilité des poids. Ainsi, le système procède à une **auto-organisation** dans laquelle les liaisons se renforcent si la **synergie hybride** est élevée et s'affaiblissent dans le cas contraire, aboutissant à des **clusters** d'entités cohérents tant du point de vue statistique que logique.

E. Applications dans des Domaines Critiques

L'intérêt d'une telle approche hybride se manifeste particulièrement dans des domaines où la **précision** et la **transparence** des décisions sont primordiales. Par exemple :

- Dans le **médical**, la partie sub-symbolique du DSL peut encoder des données brutes telles que des images médicales, des relevés cliniques ou des textes issus de l'analyse NLP, tandis que la composante symbolique intègre des règles diagnostiques éprouvées. Cette dualité permet de détecter des corrélations complexes tout en offrant des justifications claires pour chaque diagnostic.
- Dans la **surveillance industrielle**, les vecteurs décrivant des signaux (températures, vibrations) sont couplés à des règles de sécurité et de contrôle, garantissant ainsi une détection fine des pannes et la certification que les limites opérationnelles ne sont pas dépassées.

F. Conclusion

L'approche hybride proposée par le DSL permet de surmonter les limites d'un système purement sub-symbolique, souvent critiqué pour son **opacité**, et d'un système exclusivement symbolique, qui peut manquer de **souplesse** face aux signaux bruités et à la variabilité des données massives. En combinant la **puissance statistique** des **embeddings** et la **clarté** des **règles logiques**, le DSL hybride offre une solution robuste et **explicable**. Ce modèle permet non seulement une auto-organisation efficace par renforcement des liens entre entités similaires, mais fournit également une **träçabilité** indispensable dans des contextes où le **contrôle** et la **transparence** sont essentiels pour la prise de décision.

Ainsi, en conjuguant ces deux dimensions complémentaires, le DSL hybride confère au réseau la **capacité** de s'auto-organiser de manière **robuste** et **explicable**, répondant aux exigences des domaines critiques et offrant un avantage concurrentiel tant sur le plan de la performance que de l'interprétabilité.

3.5.1.2 : APPROCHE “NEURO-SYMBOLIQUE” : DÉJÀ MENTIONNÉE DANS LA LITTÉRATURE IA

L'approche **neuro-symbolique** se présente comme une stratégie intégrative visant à fusionner la **puissance statistique** des techniques d'**embeddings neuronaux** avec la **rigueur logique** des systèmes symboliques, c'est-à-dire l'ensemble de règles ou d'axiomes formels. Cette démarche, qui a été explorée dès les années 1980-1990 (voir notamment les travaux de Smolensky), a connu un regain d'intérêt avec la montée du **deep learning** dans les années 2010. Dans le contexte d'un **Deep Synergy Learning (DSL)**, cette approche permet d'associer, dans un même système, un

apprentissage efficace sur de larges volumes de données bruitées à un raisonnement explicite assurant **traçabilité** et **cohérence** globale.

A. Dimension Sub-symbolique : Puissance des Embeddings Neuronaux

Dans le sous-système sub-symbolique, chaque entité \mathcal{E}_i est représentée par un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$ issu d'un processus d'extraction automatique de caractéristiques via des réseaux de neurones profonds tels que les **CNN**, **Transformers** ou **autoencodeurs**. Ces **embeddings** permettent d'obtenir des représentations continues qui capturent des propriétés discriminantes des données. La proximité entre deux entités peut être quantifiée par des mesures telles que la **distance euclidienne** :

$$d_{\text{eucl}}(\mathbf{x}_i, \mathbf{x}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|,$$

ou par un **noyau Radial Basis Function (RBF)** défini par

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2),$$

où $\gamma > 0$ est un paramètre d'échelle. Ces mesures sont particulièrement adaptées pour gérer des données massives et bruyantes, puisque les **embeddings** permettent de projeter des instances dans un espace où les structures sous-jacentes sont révélées par des similitudes quantitatives. La tolérance au bruit et la scalabilité de ces méthodes confèrent au système une **robustesse** statistique essentielle dans le traitement de flux de données complexes.

B. Dimension Symbolique : Rigueur Logique et Raisonnement Formatif

Parallèlement au traitement sub-symbolique, le DSL intègre une dimension symbolique qui se matérialise par un **bloc** de règles ou d'**axiomes** associé à chaque entité, noté R_i . Ce sous-système symbolique permet d'effectuer un **raisonnement explicite** et de garantir la **cohérence** des inférences. Par exemple, un ensemble de règles peut être formulé pour détecter des contradictions ou pour valider la cohérence d'un ensemble de postulats. Dans des domaines exigeant une **explicabilité** accrue, comme le domaine médical ou industriel, la capacité à justifier une décision à l'aide d'un langage formel (par exemple, « si A et B sont vraies, alors C doit l'être ») est un atout majeur.

La **dimension symbolique** permet également d'opérer des inférences en se basant sur des formalismes tels que les logiques classiques, la logique floue ou des systèmes experts, assurant ainsi que le raisonnement ne repose pas uniquement sur des corrélations statistiques, mais sur des critères explicites et traçables.

C. Fusion Hybride : Définition de la Synergie Globale

Pour conjuguer harmonieusement ces deux dimensions complémentaires, un **DSL hybride** définit la **synergie globale** entre deux entités \mathcal{E}_i et \mathcal{E}_j par une combinaison linéaire pondérée. Cette synergie est formalisée par l'équation

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où $\alpha \in [0,1]$ module l'influence relative de la composante sub-symbolique par rapport à la composante symbolique. La fonction S_{sub} peut être, par exemple, une mesure de **similarité vectorielle** telle que

$$S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|^2)$$

ou une mesure basée sur le **cosinus normalisé**. Parallèlement, S_{sym} évalue la **compatibilité symbolique** entre les blocs de règles R_i et R_j en tenant compte d'éléments tels que la co-occurrence des règles, l'absence de contradiction ou la proximité ontologique entre les concepts impliqués. Ce modèle hybride permet ainsi aux entités de se regrouper dans le réseau non seulement en fonction de leur **proximité** dans l'espace vectoriel, mais aussi en fonction de leur **compatibilité logique**.

D. Auto-organisation du SCN Basée sur la Synergie Hybride

Le **Synergistic Connection Network (SCN)** intègre la synergie hybride dans la mise à jour dynamique de ses **pondérations** $\omega_{i,j}$. La règle de mise à jour adoptée est généralement de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S_{\text{hybrid}}(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ représente le **taux d'apprentissage** et $\tau > 0$ un **coefficent de décroissance** permettant de réguler l'évolution des poids. Cette équation assure que les connexions entre entités se renforcent lorsque la synergie hybride est élevée – c'est-à-dire lorsque les entités sont à la fois proches du point de vue des **embeddings** et compatibles du point de vue des règles – et s'affaiblissent dans le cas contraire. Ainsi, le SCN s'auto-organise pour former des **clusters** d'entités cohérents, caractérisés par des liens forts qui résultent d'un double critère d'affinité.

E. Références Historiques et Applications Marquantes

L'**approche neuro-symbolique** s'inscrit dans un courant de recherche de longue date dans le domaine de l'**IA**. Dès les années 1980-1990, des chercheurs tels que Smolensky ont proposé des idées visant à relier les **représentations discrètes** aux modèles neuronaux, soulignant l'intérêt de combiner les forces des approches **connexionnistes** et **symboliques**. Avec l'essor du **deep learning**, de nombreux frameworks tels que **DeepProbLog** ou des méthodes de **Differentiable Reasoners** ont émergé, proposant des solutions pour intégrer la logique au sein de réseaux de neurones différentiables. De plus, le couplage d'ontologies (par exemple via OWL) avec des **embeddings** a permis d'enrichir les modèles de raisonnement en introduisant des connaissances formelles.

Les applications de l'approche neuro-symbolique sont nombreuses :

- Dans le domaine **médical**, l'intégration d'un module d'**embedding** pour l'analyse d'images ou de textes cliniques avec des règles diagnostiques permet d'expliquer les décisions d'un système et d'assurer une **explicabilité** indispensable.
- Dans un contexte **industriel**, la gestion de flux sensoriels massifs combinée à des règles de sécurité garantit une **tolérance** au bruit tout en assurant le respect des normes.
- En **linguistique**, le couplage d'un **transformer** avec des règles syntaxiques ou sémantiques permet de concilier l'**apprentissage** des représentations linguistiques et la rigueur de la grammaire explicite.

F. Difficultés et Interfaces Entre Systèmes Neuronaux et Symboliques

L'interface entre le sous-système **différentiable** (neuronale) et le sous-système **non différentiable** (symbolique) représente un défi majeur. Tandis que la partie neuronale se met à jour par rétropropagation et bénéficie de gradients continus, la partie symbolique relève souvent d'un raisonnement binaire. Des approches telles que la logique floue, la sémantique continue des règles ou l'intégration de solveurs SAT dans des architectures neuronales ont été proposées pour atténuer cette discontinuité et permettre une meilleure **intégration** des deux mondes. Par ailleurs, la gestion simultanée de nombreux **embeddings** et de multiples blocs de règles pose des problèmes de **complexité**, nécessitant des stratégies d'optimisation pour garantir l'évolutivité du système.

. Conclusion

L'**approche neuro-symbolique** se présente comme une voie prometteuse pour allier les avantages des méthodes **connexionsnistes** – telles que la **tolérance** au bruit, la **scalabilité** et la capacité **d'apprentissage** sur de larges ensembles de données – aux atouts des systèmes **symboliques**, notamment la **traçabilité**, la **cohérence** et la capacité d'**inférence** explicable. Dans le cadre d'un **DSL**, cette intégration se matérialise par la formulation de la synergie hybride

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

et par l'adaptation des pondérations du SCN via

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hybrid}}(i, j) - \tau \omega_{i,j}(t)].$$

Cette intégration donne naissance à des systèmes d'**IA robustes** capables de manipuler des données complexes tout en garantissant une **explicabilité** et une **cohérence** conceptuelle, confirmant ainsi l'intérêt soutenu de la recherche neuro-symbolique pour la construction d'algorithmes IA complets et explicables.

3.5.2.1. Entité = Couple (\mathbf{x}_i, R_i) ? Ou Bien Deux Nœuds Connectés ?

L'intégration simultanée de la **partie sub-symbolique** (les embeddings) et de la **partie symbolique** (les règles logiques) dans un **DSL** (*Deep Synergy Learning*) peut être réalisée suivant deux stratégies architecturales distinctes dans la construction du **Synergistic Connection Network** (**SCN**). Ces deux approches se distinguent par la manière dont elles structurent les entités du réseau et par la granularité à laquelle la double information – à la fois sub-symbolique et symbolique – est traitée.

A. Entité = Couple (\mathbf{x}_i, R_i)

Dans cette première configuration, chaque entité \mathcal{E}_i est modélisée comme un **couple** unique constitué d'un **embedding** $\mathbf{x}_i \in \mathbb{R}^d$ et d'un ensemble de règles ou axiomes symboliques R_i . Le vecteur \mathbf{x}_i peut être obtenu par divers mécanismes d'extraction automatique (par exemple, un réseau de neurones convolutif pour l'analyse d'images, un Transformer pour le traitement de texte ou encore un autoencodeur pour d'autres types de signaux), tandis que R_i recense des connaissances explicites telles que des règles logiques, des axiomes ou des déclarations formelles qui décrivent l'entité.

Dans ce schéma, le **SCN** comporte exactement n nœuds pour n entités, chaque nœud encapsulant simultanément la dimension sub-symbolique et la dimension symbolique. La **synergie** $S(i,j)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j est alors évaluée à partir d'une combinaison des similarités issues de leurs embeddings et de la compatibilité de leurs blocs de règles. Ainsi, le calcul de $S(i,j)$ intègre par exemple des mesures telles que

$$S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j)$$

pour quantifier la proximité dans l'espace vectoriel, et

$$S_{\text{sym}}(R_i, R_j)$$

pour juger de l'absence de contradiction ou de la co-occurrence des axiomes associés.

Un avantage majeur de cette approche réside dans sa **simplicité** architecturale, puisque l'on ne duplique pas le nombre de nœuds dans le réseau. Chaque entité gère en interne la double information, ce qui facilite une mise à jour unifiée des pondérations $\omega_{i,j}$ selon une règle d'auto-organisation unique. Toutefois, cette fusion présente également des inconvénients : la gestion conjointe du vecteur \mathbf{x}_i et du bloc R_i peut complexifier le processus de réapprentissage, notamment si l'on souhaite réentraîner la composante sub-symbolique indépendamment des règles logiques. La modularité interne peut ainsi s'avérer moins flexible, car la mise à jour globale repose sur un seul nœud regroupant des modalités hétérogènes.

B. Deux Nœuds Connectés : L'Entité Sub-Symbolique et l'Entité Symbolique

La seconde stratégie consiste à dissocier les deux composantes d'une entité \mathcal{E}_i en deux **nœuds** distincts, chacun spécialisé dans l'une des deux modalités. Ainsi, on définit :

$$\begin{aligned} \mathcal{E}_i^{(\text{sub})} &: \text{embedding neuronal } \mathbf{x}_i \in \mathbb{R}^d, \\ \mathcal{E}_i^{(\text{sym})} &: \text{bloc de règles ou axiomes } R_i. \end{aligned}$$

Ces deux nœuds, qui représentent respectivement la partie sub-symbolique et la partie symbolique de l'entité, sont interconnectés par un **lien** explicite ou un identifiant commun qui permet d'indiquer qu'ils décrivent ensemble la même entité \mathcal{E}_i . Dans le SCN ainsi constitué, il est alors possible de distinguer deux sous-réseaux : un sous-réseau sub-symbolique où les pondérations $\omega_{(i^{(\text{sub})}, j^{(\text{sub})})}$ reflètent la proximité vectorielle (par exemple, via la norme $\|\mathbf{x}_i - \mathbf{x}_j\|$) et un sous-réseau symbolique où les pondérations $\omega_{(i^{(\text{sym})}, j^{(\text{sym})})}$ quantifient la compatibilité entre les ensembles de règles R_i et R_j . Par la suite, un mécanisme de **fusion** combine ces deux scores pour restituer la synergie globale $S(i,j)$.

L'approche à deux nœuds offre une meilleure **modularité** puisqu'elle permet de traiter séparément et d'optimiser indépendamment les deux sous-systèmes. Par exemple, il devient aisément de réentraîner la partie d'extraction d'embeddings sans impacter le module symbolique, ou inversement. Néanmoins, cette dissociation a pour conséquence de doubler le nombre de nœuds dans le réseau (passant de n à $2n$ pour n entités), ce qui peut accroître le coût de calcul lors de la mise à jour des pondérations. De plus, il faut élaborer une stratégie explicite pour fusionner les scores sub-symbolique et symbolique, c'est-à-dire déterminer la fonction de combinaison qui produira le score final de synergie entre deux entités.

C. Choix Pratique et Impact Architectural

Le choix entre la modélisation d'une entité en tant que **couple unique** (\mathbf{x}_i, R_i) et la dissociation en deux **nœuds distincts** dépend de plusieurs considérations pratiques :

- **Échelle du Système** : Dans un DSL comportant un grand nombre d'entités, la multiplication du nombre de nœuds (passage de n à $2n$) peut alourdir la structure du SCN et impacter la complexité computationnelle de la mise à jour des pondérations.
- **Modularité et Flexibilité** : Si l'on souhaite réentraîner ou modifier indépendamment la composante sub-symbolique (les embeddings) et la composante symbolique (les règles), la séparation en deux noeuds offre une meilleure modularité et facilite la gestion de la mise à jour de chacun des sous-systèmes.
- **Simplicité de l'Implémentation** : Un nœud mixte, qui intègre directement les deux composantes dans un unique vecteur complexe, présente l'avantage d'une architecture plus simple et d'une auto-organisation unifiée. Cependant, cette simplicité peut être contrebalancée par une complexification dans la gestion interne des informations hétérogènes.

Dans les deux cas, la mise à jour des pondérations suit une règle de la forme

$$\omega_{a,b}(t+1) = \omega_{a,b}(t) + \eta [\text{synergie}(a, b) - \tau \omega_{a,b}(t)],$$

où le terme $\text{synergie}(a, b)$ est défini différemment selon que a et b représentent des nœuds mixtes (cas A) ou des nœuds séparés (cas B). La fonction de fusion, qui combine la similarité sub-symbolique et la compatibilité symbolique, doit être soigneusement conçue pour garantir que le SCN converge vers une configuration stable et interprétable.

Conclusion

Les deux stratégies pour intégrer simultanément la partie sub-symbolique et la partie symbolique dans un DSL se distinguent par leur approche de la représentation des entités. La première option, qui définit chaque entité \mathcal{E}_i comme un **couple** (\mathbf{x}_i, R_i), offre une solution unifiée et simple en termes de nombre de nœuds, mais peut limiter la modularité en imposant une gestion conjointe des deux types d'informations. La seconde option, qui dissocie l'entité en deux nœuds distincts – l'un pour l'**embedding neuronal** et l'autre pour le **bloc symbolique** – apporte une meilleure séparation des préoccupations et une flexibilité accrue dans la mise à jour de chaque sous-système, au prix d'une duplication du nombre de nœuds et d'une complexité de fusion supplémentaire.

Le choix entre ces deux approches dépend donc des contraintes architecturales, de la taille du réseau et des exigences en matière de modularité et de réapprentissage. Le reste de la section (3.5.2.2) détaillera la **formule** de synergie mixte qui permet de fusionner les informations sub-symboliques et symboliques, tandis que la section (3.5.2.3) illustrera l'application de ce schéma dans des cas pratiques exploitant la force statistique des embeddings et la clarté du raisonnement symbolique.

3.5.2.2. Fonctions de Synergie Mixtes : $\mathbf{S}_{\text{sub}, \text{sym}}$

Dans le contexte d'un **Deep Synergy Learning (DSL)**, la nécessité de mesurer la proximité ou la compatibilité entre deux entités \mathcal{E}_i et \mathcal{E}_j se complexifie lorsqu'on intègre simultanément une composante sub-symbolique – représentée par des **embeddings** \mathbf{x}_i – et une composante symbolique – exprimée par un ensemble de règles ou axiomes R_i . Afin d'obtenir un **score global** qui reflète à la fois la similarité dans l'espace vectoriel et la compatibilité logique, il convient de définir une fonction de synergie mixte, notée $\mathbf{S}_{\text{hybrid}}(i, j)$. Cette section présente diverses approches d'agrégation qui permettent de fusionner ces deux dimensions, chacune offrant des compromis entre **robustesse statistique** et **explicabilité logique**.

A. Formule Linéaire Simple

La méthode la plus intuitive consiste à combiner les deux scores de manière **linéaire**. Ainsi, la synergie globale peut être définie par :

$$\mathbf{S}_{\text{hybrid}}(i, j) = \alpha \mathbf{S}_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) \mathbf{S}_{\text{sym}}(R_i, R_j),$$

où $\alpha \in [0, 1]$ est un paramètre de pondération ajustable. La fonction \mathbf{S}_{sub} évalue la **similarité vectorielle** – par exemple, via une mesure de similarité cosinus ou une fonction de noyau Gaussien telle que

$$\mathbf{S}_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2),$$

avec $\gamma > 0$ –, tandis que \mathbf{S}_{sym} quantifie la **compatibilité logique** entre les blocs de règles R_i et R_j (en évaluant, par exemple, la co-occurrence des axiomes ou en appliquant des tests de contradiction). Une telle combinaison linéaire présente l'avantage d'être aisément interprétable : en modulant α , il est possible de privilégier l'une ou l'autre des composantes en fonction du contexte d'application. Par ailleurs, une variation affine peut être introduite afin d'ajuster les échelles des scores issus des deux sous-systèmes :

$$\mathbf{S}_{\text{hybrid}}(i, j) = \alpha g(\mathbf{S}_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j)) + (1 - \alpha) h(\mathbf{S}_{\text{sym}}(R_i, R_j)),$$

où les fonctions g et h assurent un recalibrage permettant d'éviter que l'une des composantes n'écrase l'autre en cas de plages de valeurs très différentes.

B. Combinaison Multiplicative ou Min–Max

Une autre approche consiste à opter pour une agrégation **multiplicative**, dans laquelle le score global est obtenu en multipliant les scores individuels :

$$\mathbf{S}_{\times}(i, j) = \mathbf{S}_{\text{sub}}(i, j) \times \mathbf{S}_{\text{sym}}(i, j).$$

Cette formulation exige que chacune des deux composantes prenne une valeur élevée pour que le produit soit important. Ainsi, si l'une des deux valeurs est proche de zéro, le score global s'en trouve fortement diminué, ce qui renforce l'idée d'une nécessité d'accord simultané entre la partie sub-symbolique et la partie symbolique. Une alternative au produit consiste à utiliser l'opérateur min, définissant le score global comme le minimum des deux scores :

$$\mathbf{S}_{\min}(i, j) = \min\{\mathbf{S}_{\text{sub}}(i, j), \mathbf{S}_{\text{sym}}(i, j)\}.$$

Cette formulation, tout en partageant l'esprit de l'approche multiplicative, présente un comportement légèrement différent en ce qu'elle retient la composante la plus faible. Par conséquent, un faible résultat dans l'une des dimensions impose directement une limitation sur la synergie globale.

C. Stratification ou Pondération Conditionnelle

Une troisième piste consiste à introduire une **pondération conditionnelle** ou une stratification, c'est-à-dire à faire intervenir l'une des composantes uniquement lorsque l'autre satisfait un critère minimal de fiabilité. Par exemple, on peut définir une fonction de synergie hybride conditionnelle comme suit :

$$\mathbf{S}_{\text{hybrid}}(i, j) = \begin{cases} \mathbf{S}_{\text{sub}}(i, j), & \text{si } \mathbf{S}_{\text{sub}}(i, j) < \theta, \\ \alpha \mathbf{S}_{\text{sub}}(i, j) + (1 - \alpha) \mathbf{S}_{\text{sym}}(i, j), & \text{sinon,} \end{cases}$$

où θ est un seuil prédéfini. Dans cette configuration, la composante symbolique n'intervient que lorsque la similarité sub-symbolique atteint un certain niveau, assurant ainsi que le score global reflète d'abord une exclusion basée sur le critère vectoriel avant d'intégrer une pondération combinée. De plus, il est envisageable d'introduire une fonction de pondération dynamique, $\alpha(i, j)$, qui évolue en fonction de la fiabilité ou de la cohérence de l'embedding \mathbf{x}_i ou du bloc R_i , permettant ainsi un ajustement local et adaptatif de la fusion des scores.

D. Impact sur la Mise à Jour des Pondérations dans le DSL

Quel que soit le schéma d'agrégation choisi – qu'il s'agisse de la somme linéaire, du produit, du minimum ou d'une combinaison conditionnelle – la règle de mise à jour des pondérations au sein du SCN demeure inchangée :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}_{\text{hybrid}}(i, j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le taux d'apprentissage et $\tau > 0$ le coefficient de décroissance. Dans ce cadre, la fonction $\mathbf{S}_{\text{hybrid}}(i, j)$ remplace une mesure unidimensionnelle de similarité en offrant une évaluation composite qui prend en compte à la fois la **proximité** dans l'espace des embeddings et la **compatibilité** des ensembles de règles. Ainsi, une paire d'entités renforce sa connexion dans le réseau uniquement si les deux critères – la similarité sub-symbolique et la cohérence symbolique – sont satisfaits, renforçant ainsi l'idée d'un accord simultané nécessaire pour obtenir un score élevé. Le choix du schéma d'agrégation détermine alors la sensibilité du système aux valeurs extrêmes et la manière dont les imperfections dans l'une des composantes affectent globalement la synergie.

Conclusion

Les différentes fonctions de synergie mixtes présentées ici constituent des outils essentiels pour fusionner les informations provenant des domaines sub-symbolique et symbolique dans un DSL. La formulation linéaire simple offre une solution claire et modulable, tandis que les approches multiplicatives ou basées sur le minimum imposent une exigence d'accord simultané plus stricte. Les stratégies de pondération conditionnelle, quant à elles, permettent une intégration adaptative qui tient compte de la fiabilité relative des deux sources d'information. Quel que soit le choix effectué, l'agrégation se traduit dans la mise à jour des pondérations du SCN, garantissant que le

réseau s’auto-organise de manière à renforcer les liens entre des entités à la fois proches dans l'espace vectoriel et compatibles sur le plan logique. Cette intégration fine des deux modalités confère au DSL une robustesse et une explicabilité particulièrement appréciables dans des applications critiques, où la convergence d'une **puissance statistique** et d'un **raisonnement formel** constitue un avantage majeur.

3.5.2.3. Applications : Bases de Connaissance Enrichies, Multi-sensoriels et Concepts

Dans le cadre d'un **Deep Synergy Learning (DSL)** hybride, l'intégration simultanée des informations sub-symboliques – représentées par des **embeddings** extraits de données brutes – et des informations symboliques – traduites par des **règles** ou des axiomes formels – permet de créer des systèmes capables de traiter des environnements complexes tout en garantissant la **cohérence** et l'**explicabilité** de leurs inférences. Cette section examine plusieurs domaines d'application de ce paradigme hybride, en illustrant comment la fusion de ces deux types de représentations enrichit les capacités d'un système et améliore la qualité de ses clusters ou associations.

A. Bases de Connaissance Enrichies

Une application classique de cette approche hybride apparaît dans la construction de **bases de connaissance (Knowledge Bases, KB)**. Dans ces systèmes, chaque entité \mathcal{E}_i se voit attribuer à la fois un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$ issu d'un traitement sub-symbolique – par exemple, via des méthodes d'apprentissage profond appliquées à des textes ou des images – et un **bloc** de règles R_i qui formalise des assertions ou des relations symboliques (telles que des triplets RDF ou des graphes ontologiques). L'objectif consiste à conjuguer la **fiabilité formelle** (mesurée par la cohérence et la non-contradiction des règles) avec la capacité de détection de motifs latents offerte par les embeddings.

Ainsi, lorsque le DSL évalue la synergie globale

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

une forte similarité vectorielle (i.e. $\mathbf{x}_i \approx \mathbf{x}_j$) peut conduire à un rapprochement même en l'absence d'un lien explicite dans le domaine symbolique. Inversement, une forte **compatibilité** ontologique, où $R_i \cup R_j$ est hautement cohérent, peut compenser un manque relatif d'information dans l'extraction d'embeddings. Des applications telles que **Wikidata** illustrent ce phénomène, chaque entité (lieu, personne, concept) étant ainsi dotée d'une double description permettant au DSL de révéler des clusters enrichis et dynamiques, facilitant la navigation et l'exploitation d'un vaste ensemble de connaissances hétérogènes.

B. Applications Multi-Sensoriels et Concepts

Dans des environnements **multi-sensoriels**, où les données proviennent de sources variées (images, vidéos, audio, signaux de capteurs physiques), la combinaison d'un **embedding** sub-symbolique et d'un bloc symbolique offre un avantage décisif pour la détection d'anomalies ou la catégorisation de contextes. La partie sub-symbolique, obtenue par des modèles tels que les CNN pour l'imagerie ou des transformateurs pour l'audio et le texte, capture la **structure** complexe des signaux bruts. Parallèlement, le bloc symbolique R_i encode des concepts ou des règles – par exemple, des spécifications indiquant que « la température dépasse 50°C, donc il existe un risque de surchauffe

» ou « cette séquence vidéo se situe dans une ZoneInterdite » – qui permet d’interpréter ces signaux de manière explicite.

Dans un tel contexte, un DSL industriel de **surveillance** ou en **robotique** peut ainsi renforcer les liens entre entités uniquement lorsque la **cohérence** est constatée à la fois sur le plan sensoriel (embeddings similaires) et sur le plan conceptuel (règles non contradictoires). Cette double validation conduit à la formation de clusters robustes, capables de détecter avec précision des événements anormaux ou des contextes critiques. Par exemple, dans un système de surveillance, les signaux analogues détectés par différents capteurs pourront être rapprochés s’ils respectent en parallèle des conditions explicites définies dans les règles de sécurité.

C. Cas d’Étude et Bénéfices

Plusieurs cas d’étude illustrent l’intérêt d’une approche hybride :

172. **En Médecine :**

Un dossier patient peut être représenté par un **embedding** \mathbf{x}_i issu d’images médicales, de relevés cliniques ou de textes, complété par un bloc R_i contenant des règles diagnostiques (par exemple, « si Toux, Fièvre et saturation en oxygène $O_2 < 92\%$, alors admission en urgence »). La synergie mixte permet ainsi de regrouper des patients présentant des caractéristiques similaires tout en validant ces regroupements par la cohérence des antécédents ou des protocoles médicaux.

173. **En Droit ou en Compliance :**

Dans la gestion de contrats ou de documents juridiques, le texte brut d’un document peut être transformé en un vecteur \mathbf{x}_i et associé à des règles précises clarifiant des clauses ou des obligations. Le DSL rapproche des documents non seulement par la similitude de leur contenu, mais aussi par le respect des critères légaux ou contractuels, facilitant ainsi la vérification et la conformité.

174. **En Surveillance Industrielle :**

Dans un environnement industriel, les signaux provenant de capteurs (température, vibrations, etc.) sont traités par des modèles neuronaux pour extraire des embeddings, tandis que des règles de sécurité précisent les seuils critiques ou les conditions d’alerte. Le DSL hybride permet alors de détecter des anomalies en associant la similarité des signaux à la validation des règles opérationnelles.

L’avantage clé d’un DSL hybride réside dans la **complémentarité** des deux approches : la partie sub-symbolique offre une grande capacité d'**apprentissage** et de découverte de nouveaux motifs en présence de données bruitées et hétérogènes, tandis que la composante symbolique assure une **cohérence** et une **explicabilité** indispensables dans des applications critiques. Le résultat est un système capable de constituer des clusters ou des associations d’entités plus fiables et interprétables.

Conclusion

Les applications d'un **DSL hybride** se déploient de manière efficace dans des domaines aussi variés que les bases de connaissances enrichies, la surveillance multi-sensorielle ou des environnements nécessitant un raisonnement formel (médical, juridique, industriel). En combinant pour chaque entité une représentation sub-symbolique \mathbf{x}_i et un bloc de règles R_i , le système bénéficie simultanément de la **puissance** des techniques d'apprentissage profond et de la **rigueur** des approches symboliques. Les fonctions de synergie mixtes, telles que celles décrites dans la section 3.5.2.2, permettent d'auto-organiser un réseau – le **SCN** – où les liens $\omega_{i,j}$ se renforcent uniquement lorsque les deux dimensions convergent vers une **similitude** forte et une **compatibilité** logique élevée. Ce paradigme assure ainsi une meilleure gestion des environnements complexes, en offrant à la fois robustesse, flexibilité et explicabilité dans l'analyse et le regroupement des entités.

3.5.3. Avantages et Défis

Après avoir exploré (3.5.1) la **motivation** pour fusionner sub-symbolique et symbolique, et (3.5.2) les **stratégies** possibles (comment construire une entité hybride et définir la synergie mixte), nous abordons maintenant les **avantages** (3.5.3.1) et les **défis** (3.5.3.2) qu'implique une telle fusion dans le cadre d'un **DSL** (Deep Synergy Learning). Nous conclurons (3.5.3.3) par un **cas d'usage** illustratif (agent conversationnel logique + embedding).

3.5.3.1. Meilleure Explicabilité et Adaptabilité Statistique

La représentation hybride, qui associe un **embedding** neuronal \mathbf{x}_i à un **bloc logique** R_i , confère à un **Deep Synergy Learning (DSL)** une capacité accrue à rendre ses décisions et ses regroupements à la fois **transparents** et **adaptatifs**. Dans un système purement sub-symbolique, la similarité entre deux entités \mathcal{E}_i et \mathcal{E}_j se mesure essentiellement par des critères quantitatifs, tels que la distance euclidienne

$$\| \mathbf{x}_i - \mathbf{x}_j \|,$$

ou la similarité cosinus, qui, bien que efficaces pour capturer des corrélations statistiques, restent difficiles à interpréter de manière sémantique. L'intégration d'un bloc symbolique, consistant en un ensemble de règles, axiomes ou ontologies, apporte un éclairage supplémentaire permettant de comprendre la **raison** d'un rapprochement entre deux entités. Ainsi, si deux dossiers patients présentent des embeddings similaires, c'est la présence de règles diagnostiques communes ou la cohérence de leurs antécédents qui justifiera leur regroupement. Cette double dimension – **adaptation statistique** et **explicabilité logique** – offre un justificatif explicite et vérifiable aux experts.

L'**explicabilité** se matérialise par la capacité du système à lister ou à mettre en évidence les règles partagées, à démontrer l'absence de contradictions et à exposer le recouvrement ontologique entre deux entités. Dans un contexte médical, par exemple, la partie sub-symbolique peut établir une similarité basée sur des images scannées ou des relevés vitaux, tandis que le bloc symbolique clarifie que les dossiers patients se rapprochent parce qu'ils satisfont les mêmes critères

diagnostiques, tels que définis par des règles explicites. Cette transparence renforce la confiance dans le système et facilite le contrôle des décisions prises par le réseau.

Sur le plan de l'**adaptabilité statistique**, l'association d'un embedding neuronal à un cadre logico-symbolique permet de bénéficier d'un apprentissage continu et évolutif. La partie sub-symbolique, grâce à des mécanismes tels que le fine-tuning ou le ré-entraînement partiel, est capable de s'ajuster en fonction des nouveaux flux de données, absorbant ainsi la variabilité et le **bruit** inhérents à un grand volume d'informations. Par exemple, lorsque de nouvelles données apparaissent, les vecteurs \mathbf{x}_i se réajustent pour intégrer de nouveaux motifs ou corrélations, tout en restant encadrés par la composante symbolique qui assure la **stabilité conceptuelle**. En effet, la couche symbolique agit comme une contrainte qui interdit des dérives excessives en imposant des règles immuables ou des axiomes fondamentaux, ce qui évite une divergence inappropriée de l'espace des embeddings.

L'**interaction** entre ces deux composantes crée un couplage dynamique dans lequel les découvertes statistiques peuvent mener à la mise à jour ou à l'enrichissement des règles existantes. À l'inverse, des ajustements dans le bloc symbolique, par exemple la création d'un nouvel axiome fondé sur des observations répétées, peuvent orienter la réorganisation des embeddings, en forçant des rapprochements ou des éloignements dans l'espace \mathbb{R}^d . Ce mécanisme de « double contrainte » renforce la capacité du DSL à créer des liens solides et explicables tout en restant ouvert à de grandes variations dans les données, combinant ainsi **robustesse** et **transparence**.

En résumé, la **représentation hybride** utilisée dans le DSL permet de concilier deux objectifs essentiels dans des domaines sensibles tels que le médical, l'industriel ou le juridique. La composante sub-symbolique offre une **adaptation continue** aux nouveaux influx de données grâce à la puissance des techniques d'apprentissage profond, tandis que la composante symbolique garantit une **explicabilité** et une **cohérence** qui rendent le système compréhensible et contrôlable par des experts. Ce couplage renforce ainsi la capacité globale du réseau à s'auto-organiser de manière efficace et transparente, en assurant que les regroupements d'entités soient à la fois statistiquement pertinents et logiquement justifiés.

3.5.3.2. Complexité et Gestion des Contradictions Symboliques vs. Incertitude Sub-Symbolique

Dans un **Deep Synergy Learning (DSL)** hybride, l'intégration simultanée des informations sub-symboliques – représentées par des **embeddings neuronaux** – et des informations symboliques – issues de **règles** ou d'**ontologies** – permet de combiner la **robustesse statistique** à la capacité d'**inférence formelle**. Toutefois, cette fusion engendre également des défis majeurs, tant sur le plan **algorithmique** que conceptuel, du fait de la coexistence d'une **logique stricte** et d'une **incertitude progressive** inhérente aux représentations apprises. Nous proposons ici une analyse en deux axes complémentaires : d'une part, la **complexité algorithmique** induite par la comparaison conjointe des composantes sub-symboliques et symboliques, et d'autre part, la gestion des tensions entre les **contradictions symboliques** et l'**incertitude sub-symbolique**.

A. Complexité Algorithmique Accrue

Dans un système purement sub-symbolique, le calcul de la similarité entre deux entités \mathcal{E}_i et \mathcal{E}_j repose sur des opérations vectorielles ou sur des fonctions noyau, par exemple

$$\mathbf{S}_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma \|\mathbf{x}_i - \mathbf{x}_j\|^2),$$

où $\gamma > 0$. La complexité de telles opérations est généralement de l'ordre de $O(d)$ par comparaison, ou $O(n^2 \times d)$ pour l'évaluation de toutes les paires dans un ensemble de n entités. Lorsque l'on ajoute la composante symbolique, qui implique la comparaison de deux blocs de règles R_i et R_j , la charge de calcul s'accroît significativement. En effet, le **matching** de règles ou d'axiomes, et les tests de cohérence – qui peuvent nécessiter de parcourir un espace exponentiel en fonction de la complexité du langage logique (par exemple dans des formalismes tels que OWL DL) – viennent s'ajouter au coût de calcul sub-symbolique. Ainsi, la mise à jour des pondérations dans le DSL, donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hybrid}}(i,j) - \tau \omega_{i,j}(t)],$$

devient une opération dont la complexité globale est la somme de l'évaluation sub-symbolique (en $O(n^2 \times d)$) et d'une composante symbolique potentiellement NP-complète.

Pour pallier cette surcharge, plusieurs **stratégies** heuristiques peuvent être déployées. Par exemple, l'utilisation d'algorithmes de type **k-NN** ou d'**approximate nearest neighbor** permet de limiter les comparaisons symboliques à un voisinage restreint des entités les plus proches sur le plan sub-symbolique. De même, l'imposition d'un **seuil de similarité** permet d'effectuer des comparaisons logiques uniquement lorsque la proximité vectorielle atteint un certain niveau. Enfin, des techniques d'**indexation symbolique** peuvent organiser les axiomes ou concepts de manière hiérarchique, afin de réduire la nécessité de tests complets de contradiction pour chaque paire.

B. Contradictions Symboliques vs. Incertitude Sub-Symbolique

L'intégration de deux modalités hétérogènes dans le DSL peut générer des **signaux contradictoires** entre la partie symbolique et la partie sub-symbolique. D'un côté, la logique formelle, caractérisée par une **précision stricte**, peut conduire à rejeter la compatibilité entre deux entités si leurs blocs de règles R_i et R_j présentent des **contradictions** (par exemple, lorsque des axiomes incompatibles sont détectés dans $R_i \cup R_j$). Dans un système purement symbolique, une telle contradiction aurait pour effet d'annuler toute similarité entre les entités.

D'un autre côté, la composante sub-symbolique, fondée sur des embeddings calculés à partir de données bruyantes et complexes, introduit une **incertitude** inhérente. Il est fréquent que deux entités présentent des **embeddings** très similaires, malgré une ambiguïté dans l'interprétation logique de leurs caractéristiques. Ainsi, il peut exister un désaccord : la partie logique indique une incompatibilité alors que la partie neuronale suggère une forte similarité.

Pour gérer cette ambiguïté, plusieurs mécanismes d'**ajustement** sont envisageables. Dans l'agrégation hybride

$$\mathbf{S}_{\text{hybrid}}(i,j) = \alpha \mathbf{S}_{\text{sub}}(i,j) + (1 - \alpha) \mathbf{S}_{\text{sym}}(i,j),$$

le paramètre α peut être modulé afin de refléter la confiance relative dans chaque composante. Par ailleurs, l'implémentation d'un **feedback top-down** peut permettre d'ajuster dynamiquement les règles logiques si des incompatibilités persistantes sont détectées, ou inversement, de modifier les embeddings si la composante symbolique démontre une cohérence forte dans la formation des clusters. Une autre solution consiste à appliquer un mécanisme de **saturation** qui limite l'influence

négative d'une contradiction symbolique, de manière à ce que celle-ci réduise la synergie sans l'annuler complètement. Cette approche permet ainsi d'obtenir un compromis où une contradiction symbolique forte diminue significativement le score, mais ne paralyse pas la formation d'un lien si les embeddings fournissent un signal robuste.

C. Synthèse et Implications Pratiques

La cohabitation dans un DSL d'une **dimension sub-symbolique** et d'une **dimension symbolique** conduit à une complexité accrue due à la nécessité de calculer et de fusionner deux types d'évaluations : la proximité dans l'espace vectoriel, qui est déjà coûteuse pour de grands ensembles de données, et la vérification logique, souvent associée à des algorithmes dont la complexité peut croître de manière exponentielle. En parallèle, la gestion des éventuelles contradictions symboliques et de l'incertitude sub-symbolique impose un arbitrage délicat, qui se traduit par le choix du paramètre de pondération α et par l'adoption d'opérateurs combinatoires adaptés (tels que le produit ou le minimum).

Dans une implémentation pratique, la stabilité d'un DSL hybride repose sur trois axes principaux :

- La **gestion algorithmique** par le biais d'heuristiques (k-NN, seuils, indexation) permettant de limiter le coût de calcul des comparaisons logiques.
- La capacité à arbitrer les conflits entre une logique stricte et une incertitude statistique en adaptant dynamiquement la contribution de chaque composante.
- L'établissement d'un **métaniveau de contrôle** (feedback top-down) qui permet, en cas de divergence trop marquée, d'ajuster soit les règles, soit les embeddings afin d'assurer une convergence cohérente du système.

Cette approche, en somme, renforce l'intérêt des systèmes neuro-symboliques en montrant que, bien que l'intégration de deux modalités de représentation introduise une complexité supplémentaire et des tensions potentielles, il est possible d'élaborer des stratégies pour en atténuer les effets négatifs tout en tirant parti de la complémentarité offerte par chaque sous-système. Le DSL hybride, s'il est correctement paramétré et optimisé, parvient ainsi à allier la **robustesse statistique** d'un apprentissage profond avec la **r rigueur** et la **transparence** d'un raisonnement formel, garantissant ainsi des clusters d'entités qui sont à la fois pertinents et explicables.

En définitive, la gestion de la **complexité algorithmique** et des **contradictions symboliques** dans un DSL hybride nécessite un pilotage attentif des interactions entre les composants sub-symboliques et symboliques. La mise en œuvre de techniques d'approximation et de stratégies d'arbitrage permet d'obtenir un système capable de s'adapter à des environnements complexes, tout en maintenant une cohérence conceptuelle et une capacité d'explication indispensable dans des domaines sensibles.

3.5.3.3. Cas d'Usage : Un Agent Conversationnel Fusionnant Logique et Embedding

Dans le cadre d'un **DSL** (*Deep Synergy Learning*) hybride, l'intégration d'une composante **sub-symbolique** (embeddings neuronaux) et d'une composante **symbolique** (règles logiques, ontologies) trouve une illustration particulièrement parlante dans le domaine des **agents conversationnels**. L'idée fondamentale consiste à doter un chatbot d'une **représentation vectorielle** issue de modèles transformeurs (par exemple, BERT ou GPT) et d'un **bloc logique** structuré, capable d'assurer la cohérence et l'explicabilité du dialogue. Cette architecture permet ainsi de répondre aux requêtes en conciliant la capacité d'apprentissage à grande échelle et la transparence des raisonnements formels.

A. Architecture Hybride : Sub-Symbolique et Symbolique

Dans une configuration hybride, chaque message utilisateur, noté $\mathcal{E}_{\text{user}}$, est modélisé de manière double. La composante sub-symbolique se traduit par un **vecteur** $\mathbf{x}_{\text{user}} \in \mathbb{R}^d$, obtenu via un réseau neuronal entraîné sur un large corpus de dialogues. Parallèlement, la composante symbolique est représentée par un **bloc** de règles $\{R_{\text{user}}\}$ qui capture l'intention ou le sens explicite du message, par exemple en identifiant des concepts tels que "vol", "réservation", ou "date". De manière similaire, chaque module interne ou scénario de dialogue – tel que « achat de billets », « météo » ou « small-talk » – est également caractérisé par un embedding \mathbf{x}_j et un bloc de règles $\{R_j\}$. La **synergie** entre le message utilisateur et un module spécifique est alors évaluée par la fonction hybride

$$S_{\text{hybrid}}(\mathcal{E}_{\text{user}}, \mathcal{E}_j) = \alpha S_{\text{sub}}(\mathbf{x}_{\text{user}}, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_{\text{user}}, R_j),$$

où le paramètre $\alpha \in [0,1]$ module l'importance relative de la **proximité vectorielle** et de la **compatibilité logique**.

B. Exemple de Sélection de Module

Pour illustrer cette approche, considérons un scénario où un utilisateur formule la requête suivante : « Peux-tu me réserver un vol pour Barcelone la semaine prochaine ? ». Le système d'extraction sub-symbolique, reposant sur un transformeur, génère un embedding \mathbf{x}_{user} qui sera naturellement proche dans l'espace vectoriel du vecteur typique associé au module « FlightBooking » (réservation de vols). Parallèlement, le bloc logique $\{R_{\text{user}}\}$ extrait des indices explicites comme "vol", "destination" et "date", tandis que le module « FlightBooking » dispose d'un embedding \mathbf{x}_j spécifique et d'un ensemble de règles R_j définissant, par exemple, que la présence simultanée d'une destination et d'une date déclenche un scénario de réservation. La synergie sub-symbolique, calculée par une mesure telle que le **cosinus de similarité** ou un noyau gaussien, sera élevée si la sémantique de la requête correspond bien à celle du module, tandis que la synergie symbolique validera la cohérence entre les indices extraits. La combinaison des deux permet ainsi au DSL de sélectionner le module « FlightBooking » et de justifier cette décision par l'argumentation suivante :

« La similarité sub-symbolique indique que vous parlez de billets et de dates, et la logique détecte la présence de la destination 'Barcelone' ainsi que d'une date 'la semaine prochaine', ce qui correspond aux règles du scénario de réservation. »

C. Avantage en Explicabilité et Apprentissage Continu

L'intégration de ces deux composantes confère à l'agent conversationnel une **explicabilité** accrue. En effet, la partie sub-symbolique, tirée d'un modèle neuronal, est capable de capter des variations subtiles et de s'adapter continuellement grâce à des techniques de fine-tuning sur de nouveaux corpus. En parallèle, le **bloc logique** offre une justification transparente en exposant les règles partagées ou les axiomes utilisés pour interpréter la requête, ce qui est particulièrement précieux dans des domaines sensibles. Lorsqu'un nouveau scénario de dialogue est introduit (par exemple, la gestion d'abonnements), il est possible d'ajouter un nouvel embedding \mathbf{x}_{new} et un bloc de règles R_{new} associé. Cette mise à jour se répercute sur la fonction de synergie hybride, permettant au système de s'adapter en temps réel tout en maintenant une architecture explicable. De plus, si une contradiction symbolique apparaît ou si la partie sub-symbolique présente une confusion, le système peut activer des mécanismes de rétroaction (feedback) pour réviser les règles ou affiner les embeddings, garantissant ainsi une **adaptation continue** sans perte de lisibilité.

D. Conclusion

L'exemple d'un agent conversationnel illustrant l'approche hybride d'un DSL montre comment la fusion de la **partie neuronale** (embeddings) et de la **partie logique** (règles et ontologies) permet de concevoir un chatbot capable non seulement de détecter l'intention de l'utilisateur via la **proximité vectorielle**, mais également de justifier sa sélection de scénario par une **cohérence explicite**. Ce mécanisme de fusion, incarné par la fonction de synergie

$$S_{\text{hybrid}}(\mathcal{E}_{\text{user}}, \mathcal{E}_j) = \alpha S_{\text{sub}}(\mathbf{x}_{\text{user}}, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_{\text{user}}, R_j),$$

confère à l'agent une capacité d'**apprentissage continu** et d'**explicabilité**. Ainsi, le DSL hybride, en orchestrant ces deux dimensions, aboutit à un système conversationnel plus **intelligent** et plus **transparent**, capable de traiter des requêtes hétérogènes et de fournir des justifications explicites quant aux décisions prises, tout en évoluant pour s'adapter aux nouveaux flux de données et scénarios.

3.6. Aspects Avancés et Études de Cas

Les sections précédentes ont détaillé les **fondements** de la représentation des entités (chap. 3.2, 3.3, 3.4, 3.5), qu’elles soient sub-symboliques, symboliques ou hybrides. Dans cette partie (3.6), nous abordons des **perspectives** plus poussées et des **exemples concrets** illustrant l’étendue du **DSL** (Deep Synergy Learning) lorsqu’il s’attaque à des configurations complexes :

175. Les **hypergraphes** et la **synergie n-aire** (3.6.1) : on dépasse le lien binaire pour relier un **ensemble** d’entités, ouvrant la voie à des interactions collectives (image–son–texte, par exemple).
176. Les **structures fractales** (3.6.2) : la représentation elle-même peut devenir **multi-échelle** ou **self-similar**, en lien avec la fractalité (chap. 6) ; une manière d’organiser des données en hiérarchie d’emboîtements.
177. Des **études de cas** (3.6.3) : applications pratiques (analyse audio-visuelle, agent conversationnel, robotique sensorielle) démontrant comment combiner embeddings et logiques.
178. Une **comparaison expérimentale** (3.6.4) : mesurer l’influence du type de représentation (sub, sym, hybride) sur la qualité de la synergie, la vitesse de convergence et la robustesse.

3.6.1. Hypergraphes et Synergie n-aire

Jusqu’ici, la plupart des exemples de **synergie** ont porté sur des **paires** (i, j) . Mais un **DSL** peut aller plus loin en considérant des **interactions collectives** entre $\{i_1, i_2, \dots, i_k\}$. Au lieu de ne mesurer que $S(i, j)$, on définit une **hyper-synergie** $S(\{i_1, \dots, i_k\})$ qui évalue la compatibilité (sub-symbolique, symbolique, ou mixte) d’un groupe d’entités dans son ensemble.

3.6.1.1. Au-delà du Lien Binaire, Possibilité de Relier $\{i_1, i_2, \dots, i_k\}$ par une “Hyper-Synergie”

Dans la plupart des *Synergistic Connection Networks* (SCN), l’architecture se fonde sur des liaisons binaires $\omega_{i,j}$ entre deux entités \mathcal{E}_i et \mathcal{E}_j . Cette structure est suffisante lorsque l’on se limite à des interactions par paires, mais on se heurte à des limites dès lors que certains phénomènes ou combinaisons essentielles ne prennent leur sens qu’en considérant plusieurs entités simultanément. C’est le cas, par exemple, dans des données multimodales (texte + audio + image), où la cohérence globale ne s’exprime pas par la somme ou la moyenne de trois liaisons binaires, mais réellement par la compatibilité collective des trois modalités.

Pour saisir de telles interactions d’ordre supérieur, on peut s’éloigner du simple concept d’arêtes binaires et se tourner vers le formalisme des *hypergraphes*, où une *hyper-arête* relie un sous-ensemble de noeuds $\{i_1, \dots, i_k\}$. Le SCN se dote alors d’**hyper-liens** dotés d’un *score* ou d’un “hyper-synergie” :

$$S(\{i_1, \dots, i_k\}) \in \mathbb{R}^+.$$

Ce score rend compte d'une cohérence globale non réductible aux simples paires. Il peut s'agir, par exemple, d'une fonction

$$S_k(\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}),$$

si l'on se situe dans un registre sub-symbolique (des embeddings multiples), ou encore d'une compatibilité logique n -aire, si l'on manipule des blocs de règles. Dans un SCN où l'on mettrait à jour les pondérations, on n'aurait plus seulement $\omega_{i,j}$ à itérer, mais potentiellement un poids $\omega_{(i_1, \dots, i_k)}$ pour chaque hyper-arête, reflétant la force de l'interaction d'ordre k .

Un premier exemple surgit dans la *fusion multimodale* : un *triplet* {texte, image, audio} n'a de sens que collectivement (une légende d'image, une bande-son, un contexte verbal), et la validité ou la cohérence de l'objet multimédia se repère à l'échelle du triplet. Un SCN classique reliant texte–image d'un côté, image–audio de l'autre, etc., ne traduirait pas la véritable interdépendance à trois entités.

Un autre exemple, dans le domaine de la *maintenance prédictive*, concerne des *capteurs multiples* : la détection d'une alarme requiert que trois capteurs (température, pression et vibration) dépassent certains seuils simultanément. Une hyper-synergie $\omega_{(i_1, i_2, i_3)}$ se voit alors renforcée dès que l'on observe ce triple alignement.

Par ailleurs, certaines *communautés* ou *clusters* n'émergent que via des liens d'ordre supérieur. Dans la biologie, on peut étudier un système où l'interaction n'est pas captée par la somme de couples, mais par la triple ou quadruple co-occurrence de gènes ou de protéines.

Sur le plan algorithmique, étudier toutes les combinaisons de taille k dans un ensemble de n entités revient à considérer $\binom{n}{k}$ sous-ensembles, ce qui en $O(n^k)$ peut devenir prohibitif dès que $k > 2$ ou que n est grand. Les mises à jour itératives du SCN, ou l'évaluation de la synergie sur toutes ces hyper-arêtes, risquent de se révéler inenvisageables à grande échelle.

Pour conserver la *puissance* de l'hyper-synergie tout en évitant une explosion combinatoire, on adopte des heuristiques ou des restrictions :

- 179. **Limiter la taille de l'hyper-synergie** à 3 ou 4 entités maximum, si l'on sait qu'au-delà, l'utilité n'augmente plus.
- 180. **Filtrer** via un score pair à pair avant de constituer des triplets ou quadruplets : on ne forme un hyper-lien $\{i, j, k\}$ que si $\omega_{i,j}, \omega_{j,k}, \omega_{i,k}$ dépassent un certain seuil.
- 181. **Appliquer** des techniques de *sampling* ou de *randomisation* pour échantillonner parmi les combinaisons multiples, évitant de tester chaque ensemble $\{i_1, \dots, i_k\}$.

Si l'on introduit un poids $\omega_{(i_1, \dots, i_k)}$, la mise à jour prend une forme :

$$\omega_{(i_1, \dots, i_k)}(t+1) = \omega_{(i_1, \dots, i_k)}(t) + \eta [S(\{i_1, \dots, i_k\}) - \tau \omega_{(i_1, \dots, i_k)}(t)].$$

Chacune de ces hyper-arêtes se renforce ou s'affaiblit selon la “cohérence globale” du sous-groupe $\{i_1, \dots, i_k\}$. Les entités concernées peuvent se retrouver **fortement** reliées en tant que *tuple*, alors même que la somme des liaisons binaires prises deux à deux ne le suggérait pas forcément. On obtient donc une *structure hypergraphique* potentiellement plus riche et plus adaptée à certaines

relations collectives, au détriment d'une gestion plus complexe (tant dans la mise en œuvre que dans la lecture du résultat).

Conclusion

Les liens binaires $\omega_{i,j}$ constituent l'épine dorsale des SCN habituels, mais ne suffisent pas à décrire certains *patterns* où plusieurs entités s'imbriquent dans une interaction **n-aire**. L'introduction d'**hyper-liens** ou d'**hyper-synergie** $\omega_{(i_1, \dots, i_k)}$ étend les possibilités, que ce soit pour la fusion multimodale (texte–image–audio) ou pour des détecteurs multiples exigeant un “ET” logique sur plusieurs capteurs. Cette extension à des *hyperarêtes* requiert des aménagements en complexité ($\binom{n}{k}$ combinaisons) et, souvent, des heuristiques restrictives pour que la démarche reste applicable. Cependant, elle dote le **DSL** d'une **puissance** nouvelle : modéliser les **relations collectives** que ne capture pas la somme ou la moyenne de synergies binaires.

3.6.1.2. Représentation Plus Complexe : Capturer un “Ensemble” d'Entités Conjointes (par exemple, un Triplet Image–Texte–Audio)

Lorsqu'un *Synergistic Connection Network* (SCN) doit traiter une interaction qui ne se limite plus à de simples paires (i, j) , il peut être nécessaire de considérer un *ensemble* plus vaste, comme $\{i_1, i_2, \dots, i_k\}$. Cette exigence apparaît notamment dans des scénarios multimodaux, où plusieurs entités doivent être observées conjointement (par exemple, un triplet image–texte–audio). Dans le cas d'un triplet, l'auto-organisation ne peut pas se contenter de lier séparément image–texte, texte–audio et audio–image ; une **synergie** réellement *n-aire* est requise pour évaluer la *cohérence globale* de l'ensemble.

Considérons la situation d'un article ou d'un post combinant une **image**, un **texte** descriptif et un **segment audio**. Si l'on se borne à analyser les paires (image–texte, texte–audio, audio–image), on peut rater des aspects qui n'apparaissent qu'en considérant simultanément les trois modalités. Une contradiction, par exemple, peut se glisser dans l'interaction globale (image et audio se rapprochent, texte est proche de l'audio, mais la combinaison image + texte ne correspond pas). Une vision strictement binaire ne détectera pas forcément cette incohérence, tandis qu'une **hyper-synergie** {image, texte, audio} peut en faire ressortir la contradiction ou, au contraire, la cohérence collective.

Dans une représentation hypergraphique, on remplace les arêtes binaires par des *hyperarêtes* connectant plusieurs noeuds $\{i_1, i_2, \dots, i_k\}$, chacune dotée d'un *score* ou *poids* $\omega_{\{i_1, \dots, i_k\}}$. On peut alors définir une **hyper-synergie**

$$S(\{i_1, i_2, \dots, i_k\}) \in \mathbb{R}^+,$$

qui exprime la cohérence *n*-aire. Dans le cas d'un triplet {img, txt, aud}, on peut, par exemple, construire un vecteur de *fusion* $\mathbf{z}_{\text{joint}} = \text{Fusion}(\mathbf{x}_{\text{img}}, \mathbf{x}_{\text{txt}}, \mathbf{x}_{\text{aud}})$ pour la composante sub-symbolique. Et si chacune des entités possède des **règles** ou attributs symboliques $\{R_{\text{img}}, R_{\text{txt}}, R_{\text{aud}}\}$, on agrège ces blocs en un ensemble *conjoint* $R_{\text{joint}} = \bigcup_{\ell=1}^3 R_{i_\ell}$. Il est alors possible d'évaluer :

$$S(\{\text{img}, \text{txt}, \text{aud}\}) \\ = \alpha S_{\text{sub}} \left(\text{Fusion}(\mathbf{x}_{\text{img}}, \mathbf{x}_{\text{txt}}, \mathbf{x}_{\text{aud}}) \right) + (1 - \alpha) S_{\text{sym}} \left(\text{Union}(R_{\text{img}}, R_{\text{txt}}, R_{\text{aud}}) \right).$$

Si cette *hyper-synergie* est élevée, le SCN conclut à un ensemble multimodal cohérent ; dans le cas contraire, il identifie une *incohérence*.

Le passage à des hyper-synergies n -aires confronte le SCN à une explosion combinatoire : pour des ensembles de n entités, tester toutes les combinaisons de taille k se chiffre en $\binom{n}{k}$. Même pour un triplet, cela peut représenter $O(n^3)$ d'hyperarêtes. Des heuristiques de *filtrage* s'avèrent alors essentielles : on ne retient que les ensembles $\{i_1, \dots, i_k\}$ pour lesquels les paires (i_p, i_q) dépassent un certain seuil de synergie binaire, ou l'on restreint la taille k à 3 ou 4 au maximum pour contenir la complexité.

Lorsque l'on dispose d'une pondération $\omega_{\{i_1, \dots, i_k\}}$, la règle d'itération dans un SCN hybride devient :

$$\omega_{\{i_1, \dots, i_k\}}(t+1) = \omega_{\{i_1, \dots, i_k\}}(t) + \eta [S(\{i_1, \dots, i_k\}) - \tau \omega_{\{i_1, \dots, i_k\}}(t)].$$

Cette équation se passe désormais au niveau de l'hyperarête (ou “nœud conjugué”). Dans des applications multimodales, si les trois éléments image, texte et audio forment vraiment un ensemble cohérent, le *poids* $\omega_{\text{img}, \text{txt}, \text{aud}}$ s'accroît, reflétant l'identification d'un “micro-cluster” n -aire.

Conclusion

Représenter un *ensemble conjoint* $\{i_1, \dots, i_k\}$ plutôt qu'une simple paire (i, j) apporte au DSL la possibilité de gérer **d'emblée** des interactions complexes, comme la fusion multimodale (image–texte–audio) ou la co-occurrence de plus de deux capteurs. Cette approche hypergraphique — ou plus généralement n -aire — alourdit la charge en $O(n^k)$, mais permet de **déetecter** et de **stabiliser** des configurations collectives non réductibles à la somme des liaisons binaires. Dans le contexte multimédia ou sensoriel avancé, un tel mécanisme d'hyper-synergie devient souvent **indispensable** pour saisir la véritable cohérence globale.

3.6.1.3. Implications : Algorithmes Plus Lourds, mais Patterns Plus Riches

La transition d'une synergie strictement binaire ($S(i, j)$) à une **hyper-synergie** ($S(\{i_1, \dots, i_k\})$) engendre un bouleversement considérable dans l'architecture et la dynamique d'un *Synergistic Connection Network* (SCN). Si les liens binaires suffisent à décrire la plupart des interactions entre deux entités, il existe des configurations où la cohérence d'un ensemble complet $\{i_1, \dots, i_k\}$ n'est pas réductible à la somme de paires, notamment dans des contextes multimodaux (image–texte–audio) ou lorsqu'un événement résulte d'un groupe de capteurs se déclenchant simultanément. Cet enrichissement se paie toutefois par une **charge algorithmique** et une gestion plus complexes, qu'il est essentiel d'analyser pour un DSL (Deep Synergy Learning) de grande dimension.

Le principal défi provient de la **combinatoire**. Dans un réseau de n entités, considérer tous les sous-ensembles de taille k impose d'en examiner $\binom{n}{k}$, ce qui peut se chiffrer en $O(n^k)$. Le simple fait de passer de $k = 2$ (paires) à $k = 3$ (triplets) fait déjà croître le nombre d'entités à évaluer de

façon significative, et la situation s’aggrave pour des ordres supérieurs. Cela rend l'**exploration exhaustive** pratiquement impossible pour de grands n . Sur le plan du DSL, qui met à jour de façon itérative un score (pondération) lié aux synergies, on ne peut plus se contenter de vérifier $\omega_{i,j}$ pour toutes paires (i,j) . Il faut dorénavant gérer $\omega_{\{i_1, \dots, i_k\}}$ pour les hyper-arêtes, et la recalculer chaque fois qu’une entité \mathcal{E}_{i_ℓ} modifie ses attributs (embedding, règles). L’effort de réévaluation peut vite s’avérer prohibitif. Des approches **heuristiques** émergent alors, consistant à filtrer ou restreindre le champ d’examen aux groupes considérés comme prometteurs.

Un hyper-lien $\omega_{(i_1, \dots, i_k)}$ s’actualise selon un schéma dérivé de la dynamique DSL habituelle :

$$\omega_{(i_1, \dots, i_k)}(t+1) = \omega_{(i_1, \dots, i_k)}(t) + \eta [S(\{i_1, \dots, i_k\}) - \tau \omega_{(i_1, \dots, i_k)}(t)].$$

La fonction de synergie $S(\{i_1, \dots, i_k\})$ peut être défini dans un cadre sub-symbolique (fusion des embeddings $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k}$ et évaluation par un réseau) ou sémantique (compatibilité globale des blocs logiques R_{i_1}, \dots, R_{i_k}), ou encore un mélange des deux. Sitôt qu’une seule entité \mathcal{E}_{i_ℓ} change son embedding ou ses axiomes, l’hyper-synergie se retrouve possiblement obsolète, imposant un recalculation si l’on ne met pas en place un dispositif pour restreindre son impact.

Ce surcroît de complexité, s’il représente une entrave opérationnelle, fait aussi la force d’une synergie n-aire. Il est désormais possible de détecter des *patterns* ou des phénomènes collectifs absents du spectre binaire. Dans un triplet {image, texte, audio}, la cohérence globale n’est plus strictement la somme ou la moyenne des relations deux à deux, mais véritablement un effet conjoint. De même, un ensemble de capteurs {capteur₁, capteur₂, …} peut signaler un événement critique seulement si tous se déclenchent en même temps. La synergie n-aire permet alors une modélisation plus fidèle et plus “haut niveau” de ces scénarios.

Au plan de l’implémentation, on recourt souvent à des **stratégies de filtrage** visant à n’explorer que les sous-ensembles d’entités ayant déjà prouvé une certaine proximité binaire (chacun des couples (i_ℓ, i_m) doit dépasser un seuil de similarité). Cette méthode de “candidats par paires” réduit la quantité d’hyperarêtes à évaluer en détail. Il est également envisageable d’employer des procédures d'**exploration incrémentale** (où l’on agrège peu à peu des entités sur la base d’un gain de synergie) ou des techniques de **recuit simulé** pour éviter l’énumération systématique de $\binom{n}{k}$.

La possibilité de mettre à jour un poids $\omega_{(i_1, \dots, i_k)}$ a, dans un *Synergistic Connection Network*, d’importantes répercussions : certains *hyper-lots* d’entités (groupements n-aires) vont se renforcer et former des “hyper-clusters”. On peut par exemple voir émerger un “triple cluster” {img₁, txt₅, aud₃} fermement relié par $\omega_{1,5,3}$ si ces trois modalités s’accordent fortement. Cette structure n’est pas réductible à un graphe usuel, puisqu’il faut appréhender la cohésion collective. Un aspect intéressant survient en logique symbolique, où la vérification d’une cohérence globale $\{R_{i_1}, \dots, R_{i_k}\}$ peut nécessiter des tests complexes (cohérence n-aire, potentiellement NP-complets selon la expressivité logique). Tout l’enjeu consiste à contrôler la croissance de la complexité via un design parcimonieux (filtres, heuristiques, indices) et à tirer avantage de l’expressivité supplémentaire qu’offre une synergie n-aire.

Conclusion

L’extension du DSL vers une *hyper-synergie* ouvre la possibilité de **capturer** des *patterns* ou *groupements* qui se manifestent seulement par la convergence simultanée de plusieurs entités. Les

coûts de calcul croissent toutefois à un rythme combinatoire, et la mise à jour de chaque hyper-lien $\omega_{(i_1, \dots, i_k)}$ implique de nouveaux processus (fusion d'embeddings, compatibilité symbolique multiple). Les *algorithmes* deviennent plus lourds, mais le réseau s'enrichit de la capacité à *déetecter et stabiliser* des configurations complexes, cruciales pour de nombreuses applications multimodales, collectives ou multisensorielles. C'est ce **compromis** entre l'effort algorithmique et la découverte de *patterns plus riches* qui motive l'exploration de la synergie n-aire et des hyperarêtes dans un *Synergistic Connection Network*.

3.6.2. Structures Fractales de Représentation

Au-delà des hypergraphes et de la synergie n-aire (section 3.6.1), une autre voie **avancée** pour organiser la représentation dans un **DSL** (Deep Synergy Learning) consiste à exploiter des **structures fractales** ou auto-similaires. L'idée est d'employer un **multi-niveau** ou un **multi-échelle** (en lien avec le chapitre 6) où la représentation de chaque entité (ou de chaque cluster) peut se répliquer ou s'itérer sous une forme **fractalement** organisée. Dans certains types de données, on observe en effet des **patterns** se répétant à différentes échelles — qu'il s'agisse d'images (motifs fractals), de taxonomies hiérarchiques (ontologies en forme d'arbre auto-similaire), voire d'embeddings qui se réorganisent récursivement.

3.6.2.1. Lien avec le chap. 6 (multi-échelle + fractalité) : la représentation elle-même peut s'avérer fractale (ex. embeddings récursifs, hiérarchie de concepts)

La perspective multi-échelle exposée au chapitre 6 souligne que, dans un **DSL** (*Deep Synergy Learning*), un *Synergistic Connection Network* peut se structurer en plusieurs *niveaux* ou *couches* successifs. Chaque niveau agrège ou récapitule les motifs détectés au niveau inférieur, créant ainsi un effet d'**auto-organisation hiérarchique**. Cet agencement à étages peut aller plus loin lorsque la représentation des entités exhibe un caractère **auto-similaire**, voire **fractal** : à chaque nouvelle échelle, on retrouve un schéma semblable, reproduisant le même mode d'organisation à des granularités différentes. Il en résulte la possibilité d'embeddings « récursifs » ou de hiérarchies conceptuelles répliquant la même forme à divers niveaux.

Sur le plan **sub-symbolique**, on peut concevoir que l'embedding $\mathbf{x}_i \in \mathbb{R}^d$ représentant une entité \mathcal{E}_i se scinde en *mini-embeddings* itératifs, ce qui reflète une *structure fractale*. Dans certains domaines, comme la modélisation de textures ou de formes naturelles (ex. arbres, littoraux), les mêmes motifs géométriques se répètent à diverses résolutions. On formalise alors des schémas d'autoencodeurs profonds ou de CNN « en cascade », où l'opération $\mathbf{x}_i^{(\ell+1)} = F(\mathbf{x}_i^{(\ell)})$ reproduit un *pattern* identique à chaque étape ℓ . Cette auto-similarité sous-tend une représentation fractale, que le **DSL** peut exploiter pour repérer ou stabiliser des clusters multi-échelle : une sous-structure locale se retrouve dans une macro-structure globale.

Sur le plan **symbolique**, les ontologies ou ensembles de règles peuvent, elles aussi, refléter une organisation fractale. Dans une ontologie de grande taille, il arrive qu'une arborescence de concepts se *duplique* partiellement dans diverses branches : le même *motif* notionnel réapparaît, générant des *sous-ontologies* répétant la même logique de liens (subClassOf, partOf, etc.). Un **DSL** hiérarchique peut alors reconnaître, au niveau micro, un groupement conceptuel, puis observer le

même schéma à une échelle plus macro, renforçant la cohérence globale et la capacité de généralisation.

La dimension **multi-échelle** intervient lorsque l'on empile plusieurs *couches* dans le réseau, chacune gérant une échelle de représentation. On obtient un flux *ascendant* (bottom-up), où les motifs découverts localement se consolident, et un flux *descendant* (top-down), où les structures macro peuvent imposer une cohérence aux entités de rang inférieur. Dans un *cadre fractal*, cette interaction est d'autant plus fluide que la même forme **p** ou la même logique **r** se reproduit à chaque palier, ce qui évite de réinventer un nouveau schéma à chaque fois.

Mathématiquement, on peut décrire la fractalité dans le sous-espace \mathbb{R}^d par une opération F appliquée de façon itérative à un point ou à une région, créant un ensemble invariant par F . Dans un **DSL**, si l'on modélise l'évolution d'un embedding $\mathbf{x}_i^{(\ell)}$ à mesure qu'on change de niveau ℓ , on peut écrire $\mathbf{x}_i^{(\ell+1)} = F(\mathbf{x}_i^{(\ell)})$ pour une transformation donnée. Lorsque F est contractante, on obtient un *attracteur fractal*, dont la structure se reflète dans l'organisation multi-niveau des entités.

Le chapitre 6, centré sur l'apprentissage multi-échelle, se prête donc à accueillir ce genre de *représentation fractale* : à chaque palier, l'**auto-organisation** exploite la même règle de mise à jour, la même forme d'ancrage (sub-symbolique ou symbolique) mais scannée à un degré de finesse différent. Ce principe peut renforcer l'efficacité du **DSL** dans des domaines où la notion de *self-similarité* est prépondérante (biologie, géologie, architecture conceptuelle, etc.).

En **conclusion**, l'idée que la représentation (embedding vectoriel ou bloc logique) elle-même s'avère fractale met en évidence une *self-similitude* reliant les niveaux micro et macro. Cela s'inscrit dans l'esprit du *multi-échelle* : le **DSL** ne se contente pas de gérer $\mathcal{E}_1, \dots, \mathcal{E}_n$ à un unique niveau, mais décline la même dynamique auto-organisée sur plusieurs strates, avec un effet de *miroir* ou de *répétition* du même motif. La complexité ainsi introduite (gérer une fractalité d'échelles) se voit compensée par la puissance descriptive obtenue : les *patterns* détectés localement se généralisent, et la hiérarchie globale se nourrit de la *ressemblance* entre niveaux, au bénéfice d'une meilleure cohérence d'ensemble.

3.6.2.2. Exemple : embeddings fractals pour des données “self-similar”, ou ontologies fractales pour la taxonomie

La **fractalité** joue un rôle fondamental dès que l'on observe des motifs identiques ou très similaires qui se reproduisent à différentes **échelles**. Dans le cadre d'un **Deep Synergy Learning (DSL)**, où l'on cherche à organiser des données (vectorielles ou symboliques) de façon auto-similaire, l'exploitation de cette propriété fractale permet une **intégration** plus économique et plus robuste. Cette section illustre deux cas d'application : l'un concerne la construction d'**embeddings fractals** pour des données intrinsèquement **self-similar** (images, signaux, textures), l'autre traite de **taxonomies** ou **ontologies** présentant elles-mêmes une structure auto-similaire récurrente.

Afin de souligner l'importance de ces approches, il convient de rappeler que la **dimension fractale** d'un objet F peut être définie, par exemple, via la formule usuelle de **self-similarité** : si la figure est couverte par N copies d'elle-même à l'échelle $1/r$, alors

$$\dim_{\text{fractale}}(F) = \frac{\ln(N)}{\ln(r)}.$$

La réitération d'une même structure à différentes résolutions se révèle un facteur clé tant dans le cas sub-symbolique (données géométriques ou multi-échelle) que dans le cas symbolique (ontologies arborescentes). Cette auto-similarité offre un levier pour construire des **représentations** et des **mesures de similarité** adaptées au principe de fractalité.

A. Embeddings fractals pour des données self-similaires

Dans de nombreux champs scientifiques, les données présentent un **comportement auto-similaire** : un motif se reproduit à plusieurs échelles, révélant une structure fractale sous-jacente. On rencontre cette propriété dans des *textures naturelles* (extraction de *patches* à différents niveaux de zoom), dans des *signaux biologiques* pouvant dévoiler la même forme d'onde à des fréquences multiples, ou dans certains *objets géométriques* (côtes maritimes, contours fractals). L'idée est alors de coder chaque entité \mathcal{E}_i à l'aide d'un **embedding fractal**, c'est-à-dire d'un vecteur qui capture la redondance d'échelle.

Si l'on note $\mathbf{x}_i^{(l)}$ la description de \mathcal{E}_i à l'échelle l (par exemple, la sortie d'un *patch-encoder* appliqué à un *patch* de résolution l), on peut concaténer ou fusionner ces descriptions sur plusieurs niveaux, produisant au final un **embedding** $\mathbf{x}_i \in \mathbb{R}^D$ qui agrège cette information multi-résolution. La **fractalité** de l'entité, exprimée par la répétition statistique du même motif à plusieurs granularités, se retrouve alors dans la structure même de \mathbf{x}_i .

Une formalisation plus mathématique introduit une fonction de type

$$\mathbf{x}_i = [f(\mathbf{x}_i^{(0)}), f(\mathbf{x}_i^{(1)}), \dots, f(\mathbf{x}_i^{(L)})],$$

où f est un encodeur non linéaire (par exemple un autoencodeur profond) appliqué à différentes résolutions. La **synergie** entre deux entités \mathcal{E}_i et \mathcal{E}_j peut alors se définir via une **similarité** (cosinus, noyau gaussien, etc.) entre leurs embeddings fractals respectifs :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \exp(-\alpha \parallel \mathbf{x}_i - \mathbf{x}_j \parallel),$$

assurant que deux **configurations fractales** très proches (i.e. répétant le même motif à chaque niveau) se verront attribuer une **synergie** élevée. Dans un **SCN**, ces entités fortes en **self-similarité** locale formeront un **cluster** auto-organisé révélant leur communauté de forme.

B. Ontologies fractales pour la taxonomie

Si l'on se situe dans un registre plus symbolique ou conceptuel, la **fractalité** peut apparaître sous forme d'une *hiérarchie* ou *arborescence* qui se répète, quasi à l'identique, dans chaque sous-branche. De nombreuses *taxonomies* en biologie, chimie, ou dans des classifications industrielles, reproduisent la même structure de relations et de propriétés à différents étages. On parle alors de **fractalité sémantique**, puisque la même architecture se manifeste à plusieurs niveaux de la hiérarchie, de sorte que l'on peut utiliser un *motif ontologique* unique pour chacune des sous-branches.

On peut modéliser une telle ontologie comme un graphe \mathcal{G} muni d'une application récursive, par exemple

$$\phi: \mathcal{G} \rightarrow \mathcal{G} \times \{0, \dots, k - 1\}$$

qui identifie des *sous-graphes* isomorphes répliquant un même ensemble de classes ou de relations. Cette propriété est analogue à la fonction d'itération d'un fractal géométrique, mais appliquée aux liens sémantiques (*subClassOf*, *hasPart*, *connectedTo*, etc.).

Dans le cadre d'un **DSL**, une telle **auto-similarité** conceptuelle renforce la **cohérence multi-niveau**. La **mise à jour** des poids de synergie, notée $\omega_{ij}(t + 1) = \dots$, ou l'intégration d'une règle symbolique au niveau supérieur, peuvent automatiquement se propager aux étages fractals sous-jacents, réduisant l'effort de maintenance. Par ailleurs, la similarité sémantique entre deux *sous-domaines* peut se calculer en comparant leurs **motifs ontologiques**. Deux branches véhiculant des structures quasi identiques, à quelques variations de propriétés près, obtiendront une **distance** sémantique faible et, en conséquence, une **synergie** élevée dans le **SCN** symbolique.

C. Gains potentiels

L'approche **fractal-based** procure un **gain** en compression : un même schéma ou un même encodeur multi-résolution peut être réutilisé à diverses échelles. Elle amplifie aussi la **robustesse** : lorsqu'une révision touche la structure de base, elle s'applique à toutes les répliques, évitant les *incohérences locales*. Sur le plan purement mathématique, l'uniformisation de la **dimension fractale** ou la réplication d'un **pattern** identique signifient que la **synergie** ne fluctue pas brutalement d'un étage à l'autre, mais demeure gouvernée par un **principe** d'auto-similarité. Les mises à jour locales des poids ω_{ij} (ou l'apparition de nouvelles entités analogues) s'intègrent plus harmonieusement dans l'ensemble.

On peut inclure une *analyse dimensionnelle* pour vérifier que l'object fractal formé par la hiérarchie ou l'ensemble des embeddings est susceptible de présenter un degré de redondance réduisant la *complexité effective*. Ainsi, dans une ontologie auto-similaire, la **profondeur** L de la structure peut conditionner la **dimension** du schéma de liens, souvent inférieure à ce qu'on obtiendrait si chaque étage était dessiné de manière indépendante. Cette propriété apparaît clairement dans les constructions fractales pures, où la **taille** du système croît moins vite que l'**échelle** (puisque l'on recycle le même motif à chaque niveau).

Conclusion

En tirant parti de la **fractalité**, le DSL adopte une vision **multi-niveau** où la récurrence du même motif, qu'il soit sub-symbolique (embedding fractal) ou symbolique (ontologie arborescente), facilite la **mise à jour** simultanée à toutes les échelles. Dans le cas sub-symbolique, un **embedding** multi-résolution reflète l'auto-similarité de la donnée, améliorant la précision de la synergie pour des entités effectivement "fractal-like". Dans le cas symbolique, une **taxonomie** ou **ontologie** fractale évite la duplication inutile de motifs dans chaque sous-branche et garantit que toute évolution locale (règle, propriété, relation) se répercute partout. Le **Synergistic Connection Network** profite alors d'une **cohésion** globale où les mêmes principes se déclinent à chaque étage, tout en maintenant un haut degré de **stabilité** et de **compacité**. Cette exploitation de la fractalité, englobant l'aspect géométrique (sub-symbolique) et l'aspect conceptuel (symbolique), s'avère

cruciale pour de nombreux domaines, de la géologie à la classification industrielle, et constitue un atout essentiel dans la démarche d'auto-organisation du **Deep Synergy Learning**.

3.6.2.3. Gains potentiels en compression ou en cohérence multi-niveaux

L'exploitation d'une **structure fractale** dans le cadre d'un **Deep Synergy Learning (DSL)** permet de renforcer à la fois la **compression** des descriptions et la **cohérence** entre différents niveaux d'analyse. Cette intégration repose sur l'idée que lorsqu'un *motif* se répète à diverses échelles, on peut le stocker ou le calculer une seule fois et le réutiliser de façon récurrente.

A. Compression et réduction de la redondance

Lorsqu'une représentation fractale ou une ontologie auto-similaire est mise en place, la **redondance** entre les différentes entités se matérialise par la récurrence du même schéma à plusieurs échelles. Supposons qu'un motif **m** se retrouve répliqué dans plusieurs branches d'un arbre ou à différents niveaux de résolution. Au lieu de décrire ce motif **m** chaque fois qu'il apparaît, on peut le factoriser dans une structure commune. Si le fractal possède un facteur d'échelle $r > 1$ et qu'on retrouve le même motif dans N zones à chaque itération, la quantité totale d'information mémorisée peut se réduire à un *coût de base* plus faible que la somme naïve des contributions de chaque zone.

Une modélisation plus formelle peut invoquer une fonction de self-similarité. Si la structure globale \mathcal{F} se décrit par un ensemble de copies de dimension α , on a $\dim_{\text{fractale}}(\mathcal{F}) = \alpha$, alors que la description complète (sans prise en compte de l'auto-similarité) conduirait à une dimension apparente plus grande. Dans un **SCN** (Synergistic Connection Network), un grand nombre de nœuds peuvent partager la même sous-structure interne ; la référence à cette sous-structure suffit alors, ce qui donne un **gain** en $O(\log n)$ ou $O(n^\delta)$ (selon la fractalité) plutôt qu'en $O(n)$. D'un point de vue algorithmique, le recalcul des **synergies** $S(i, j)$ s'en trouve lui aussi allégé, car chaque motif commun **m** ne s'encode qu'une fois avant d'être propagé dans l'ensemble des entités concernées.

B. Cohérence multi-niveaux et synergie fractale

Les modèles fractals sont particulièrement adaptés à la logique **multi-niveau** qui sous-tend le DSL. Lorsque la même forme se répète à différentes échelles, la **cohérence** entre niveaux local et global se voit considérablement renforcée. Une entité \mathcal{E}_i peut se trouver au niveau micro avec des liens $\omega_{i,j}(t)$ qui reproduisent, en "miniature", la structure générale observable au niveau macro.

Dès que l'on considère la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

on remarque que le schéma fractal induit un renforcement cohérent à chaque échelle. Si les entités \mathcal{E}_i et \mathcal{E}_j appartiennent à un motif reproduit dans plusieurs parties du réseau, la synergie $S(i,j)$ demeure élevée dans chacun de ces contextes, ce qui stabilise $\omega_{i,j}$ de manière récursive. Les mêmes **clusters** ou agrégations se retrouvent ainsi à divers niveaux de granularité, ce qui unifie l'auto-organisation.

Sur le plan conceptuel, cette **auto-similarité** multi-niveau peut être rapprochée d'un zoom fractal : on peut se déplacer entre une échelle fine et une échelle globale, tout en constatant la même topologie répliquée. Le **DSL** bénéficie alors d'une stabilité accrue, car la redondance des liaisons synergiques évite les contradictions entre un niveau et l'autre.

C. Stabilité et scalabilité dans les applications fractales

La dimension fractale sert également de garant de **stabilité** et de **scalabilité** pour le DSL. Dans des domaines comme la **géologie**, où les images présentent des textures fractales, on peut adopter un embedding fractal afin de capturer la redondance d'échelle et réduire la dimension effective des données. Dans l'**ontologie** d'un système complexe, on peut dupliquer un même module conceptuel pour chaque subdivision, ce qui préserve la cohérence de la base de connaissances. On formalise parfois cette récurrence par une application Φ qui envoie chaque branche vers un motif identique mais étiqueté différemment,

$$\Phi: \mathcal{G} \rightarrow \bigsqcup_{\ell=1}^k \mathcal{G}_\ell,$$

là où \mathcal{G}_ℓ est une copie partiellement isomorphe de \mathcal{G} .

Ce principe accroît la **scalabilité** du réseau, car l'augmentation de la taille (nombre de nœuds ou de classes) n'exige pas de réinventer la structure de liens, mais simplement d'appliquer la transformation Φ ou d'exploiter les mêmes blocs encodés. Les **clusters** ou groupes de nœuds se forment ainsi de façon reproductible, même lorsque l'on déploie une ontologie à plusieurs milliers de branches. Le rapport entre la complexité globale et la profondeur des niveaux demeure raisonnable, en raison de la répétition du *motif fractal* à chaque sous-niveau.

L'effet bénéfique sur la **stabilité** tient au fait qu'un motif validé localement engendre les mêmes propriétés au niveau global. Les paramétrages $\omega_{i,j}$ déjà convergés à un étage se transposent aux autres étages, conférant au **DSL** une robustesse face aux modifications et aux extensions. Un schéma local ne nécessite plus d'être redéfini dans toute la hiérarchie, ce qui évite les incohérences et limite les oscillations.

Conclusion

Les **structures fractales** mises en place dans un DSL procurent des **gains** appréciables en matière de **compression** et de **cohérence multi-niveaux**. En réduisant la redondance, on obtient des économies de calcul et de mémoire ; en garantissant une forme identique entre plusieurs échelles, on assure une meilleure stabilité et une propagation plus simple des mises à jour. Qu'il s'agisse d'**embeddings fractals** pour des données self-similaires ou d'une **ontologie** répliquant le même module conceptuel à chaque sous-branche, la synergie fractale améliore la **scalabilité** globale et renforce le principe d'auto-organisation. Les **pondérations** s'accordent ainsi entre local et global, et la dynamique de renforcement s'applique uniformément, permettant au **SCN** de révéler naturellement ses patterns récurrents et de maintenir une cohérence solide à toutes les échelles d'observation.

3.6.3. Études de Cas Illustratives

Après avoir exploré différentes **extensions** et **approches** avancées (hypergraphes n-aires, structures fractales, etc.), il est utile de **concrétiser** ces idées à travers quelques **cas d'usage** représentatifs. Dans cette section (3.6.3), nous décrirons trois scénarios où un DSL (Deep Synergy Learning) peut trouver une application pratique et mettre en œuvre les principes de représentation (sub-symbolique, symbolique, hybride) examinés tout au long du Chapitre 3 :

182. (3.6.3.1) Un **système d'analyse audio-visuelle**, combinant des **vecteurs CNN** pour l'image, des **embeddings** pour l'audio et des **règles logiques** propres à un domaine spécifique (ex. surveillance, broadcast, etc.).
183. (3.6.3.2) Un **agent conversationnel**, déjà évoqué dans des sections antérieures, qui fusionne des "règles" (cohérence contextuelle) et des embeddings textuels (ex. BERT).
184. (3.6.3.3) Un **scénario de robotique sensorielle**, où la représentation sub-symbolique (capteurs multiples) cohabite avec une représentation symbolique décrivant les actions, les objets et la logique de manipulation.

3.6.3.1. Système d'analyse audio-visuelle : vecteurs CNN + embeddings audio + logiques domain-specific

Dans les applications de **surveillance**, **d'indexation multimédia** ou de **diagnostic** dans un environnement audiovisuel complexe, il est souvent crucial de fusionner une **information visuelle** (généralement encodée par un **réseau de neurones convolutifs**, CNN) et une **information sonore** (par exemple encodée via des **embeddings audio**), tout en les contrignant par des **règles logiques** spécifiques au domaine. Le **Deep Synergy Learning (DSL)**, et plus particulièrement son réseau SCN (Synergistic Connection Network), offre un cadre systématique pour opérer cette fusion et détecter des **événements** ou **patterns** pertinents.

Il est d'usage de représenter chaque segment temporel issu de la **vidéo** par un vecteur $\mathbf{x}_{\text{vid}} \in \mathbb{R}^{d_1}$, obtenu à l'aide d'un **CNN** pré-entraîné ou d'un autre modèle **sub-symbolique**. Parallèlement, chaque segment temporel **audio** se voit attribuer un vecteur $\mathbf{x}_{\text{aud}} \in \mathbb{R}^{d_2}$, issu d'un traitement du signal tel qu'un **spectrogramme** encodé ou un modèle **transformer audio**. Une procédure de **fusion** des deux composantes, notée

$$\mathbf{x}_{\text{joint}} = F_{\text{fusion}}(\mathbf{x}_{\text{vid}}, \mathbf{x}_{\text{aud}}),$$

peut se contenter d'une **concaténation** ou employer des techniques plus avancées (attention croisée, somme pondérée). La dimension symbolique intervient à travers un ensemble de **règles** ou **d'axiomes** propres au domaine, par exemple : "Si une zone interdite est franchie et qu'un son d'alarme est détecté, alors alerte.". On peut modéliser ces règles par des **propriétés logiques** regroupées dans un bloc \mathbf{R}_{vid} pour la partie vidéo, et \mathbf{R}_{aud} pour la partie audio, ou bien un bloc commun si des règles globales couvrent l'ensemble du signal.

La **fonction de synergie** doit alors refléter à la fois la proximité sub-symbolique (cohérence **audio-visuelle**) et la compatibilité des **règles logiques** (compatibilité sémantique). Afin d'agréger ces deux critères, on définit

$$S_{\text{hybrid}}(\mathcal{E}_{\text{vid}}, \mathcal{E}_{\text{aud}}) = \alpha S_{\text{sub}}(\mathbf{x}_{\text{vid}}, \mathbf{x}_{\text{aud}}) + (1 - \alpha) S_{\text{sym}}(\mathbf{R}_{\text{vid}}, \mathbf{R}_{\text{aud}}).$$

La composante sub-symbolique S_{sub} peut être une **similarité** exponentielle sur la norme de la différence,

$$S_{\text{sub}}(\mathbf{x}, \mathbf{y}) = \exp(-\beta \|\mathbf{x} - \mathbf{y}\|),$$

ou tout autre indicateur mesurant à quel point la **signature visuelle** et la **signature audio** coïncident (corrélation temporelle, alignement spectral, etc.). La composante symbolique S_{sym} se fonde sur la logique du domaine : si les axiomes audio et vidéo ne se **contredisent** pas, et mieux encore s'ils **confirment** l'un l'autre, alors $S_{\text{sym}}(\mathbf{R}_{\text{vid}}, \mathbf{R}_{\text{aud}}) \approx 1$. En revanche, une contradiction explicite (par exemple, "alarmeActivée=TRUE" côté audio mais "alarmeInopérante=TRUE" côté vidéo) fait chuter drastiquement le score symbolique.

Le **SCN** peut ensuite **auto-organiser** les segments audio-visuels au moyen d'une **règle de mise à jour** itérative :

$$\omega_{(\text{vid}, \text{aud})}(t+1) = \omega_{(\text{vid}, \text{aud})}(t) + \eta [S_{\text{hybrid}}(\mathcal{E}_{\text{vid}}, \mathcal{E}_{\text{aud}}) - \tau \omega_{(\text{vid}, \text{aud})}(t)].$$

Un lien $\omega_{(\text{vid}, \text{aud})}$ se voit **renforcé** (croît progressivement) si la synergie audio-vidéo reste élevée ; dans le cas contraire, il décroît. Après plusieurs itérations, ce mécanisme local de **plasticité** fait apparaître des **clusters** de segments fortement reliés, correspondant à des événements ou situations reconnues.

Pour décider d'une **alerte** ou d'une **étiquette** associée à un intervalle temporel, on peut fixer un **seuil** ω_{\min} : dès que $\omega_{(\text{vid}, \text{aud})}$ dépasse ω_{\min} , le **réseau** considère la relation (vid, aud) comme un événement fusionné. La logique additionnelle (par exemple, "intrusion interdite + alarme => situation critique") permet alors de produire un **verdict** plus précis, tant en termes de classification (type d'incident) que **d'explicabilité** (la logique indique pourquoi il y a match entre la vidéo et l'audio).

Ce **système multimodal** s'avère particulièrement robuste. Si la **vidéo** se trouve partiellement brouillée (mauvaise luminosité, occlusions), la **composante sonore** peut prendre le relai et maintenir un niveau élevé de synergie. Symétriquement, une détérioration du signal audio peut être compensée par l'analyse visuelle. Dans des applications réelles comme la **surveillance** en zones sensibles ou la **détection d'incidents** (pannes, alarmes, cris, collisions), l'association d'un bloc de **règles symboliques** vient enrichir la détection : on ne se contente pas d'une simple ressemblance sub-symbolique, mais on vérifie la **compatibilité logique**. La structure DSL autorise aussi une **adaptation continue** : l'**embedding** vidéo ou audio peut être réentraîné sur de nouvelles données, tandis que les **règles** du domaine se mettent à jour (nouvelles situations réglementaires, nouveaux types d'alarmes), le tout sans remettre en cause l'ensemble des acquis du SCN.

Conclusion

Ce prototype d'**analyse audio-visuelle** illustre toute la **puissance** d'une intégration multimodale dans le **Deep Synergy Learning**. Les vecteurs **CNN** et les **embeddings audio** assurent la modélisation sub-symbolique, tandis que la **logique domain-specific** clarifie la cohérence ou la contradiction entre segments. La **synergie** hybride (sub-symbolique + symbolique) régit la mise à jour des pondérations, aboutissant à un **réseau** qui se stabilise sur les événements les plus

signifiants. Cette approche valorise la **complémentarité** des signaux audio et vidéo, tout en restant explicable au moyen de **règles** aisément interprétables, ce qui s'avère déterminant dans des contextes exigeant de la transparence (sécurité, documentation légale, fiabilité industrielle).

3.6.3.2. Agent conversationnel : combine “règles” pour la cohérence contextuelle + embeddings BERT

Dans le cadre d'un **Deep Synergy Learning (DSL)**, la construction d'un **agent conversationnel** (chatbot) repose sur l'intégration d'une dimension **sub-symbolique** et d'une dimension **symbolique**. L'objectif est de combiner, d'une part, la puissance d'un modèle d'**embeddings** capable de saisir la richesse et la variabilité du langage naturel – en l'occurrence, des représentations contextuelles produites par un modèle comme **BERT** (*Bidirectional Encoder Representations from Transformers*) – et, d'autre part, un ensemble de **règles logiques** qui garantissent la cohérence contextuelle et la maîtrise du dialogue. Cette approche hybride permet ainsi à l'agent d'analyser de multiples formulations textuelles tout en respectant une séquence de *slots* ou de conditions qui encadrent la progression du dialogue.

A. Composante sub-symbolique : embeddings BERT

Lorsqu'un utilisateur énonce une phrase ou un segment textuel, le DSL fait appel à **BERT** pour générer un **embedding** qui représente la sémantique de l'énoncé. On définit ainsi, pour une entité textuelle \mathcal{E}_i ,

$$\mathbf{x}_i = \text{BERT}(\mathcal{E}_i) \in \mathbb{R}^d,$$

où d représente la dimension de l'espace des embeddings. Ce vecteur \mathbf{x}_i encapsule les informations contextuelles et syntaxiques de la phrase. Par exemple, deux phrases exprimant une intention similaire (même signification même si formulées différemment, malgré des fautes ou l'usage de synonymes) conduiront à des vecteurs dont la **similarité cosinus** sera élevée, exprimée mathématiquement par

$$S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

La robustesse de BERT, particulièrement lorsqu'il est **fine-tuned** sur des corpus spécifiques à un domaine (assurance, banque, médical, etc.), garantit que même des variations linguistiques (fautes, abréviations) n'altèrent pas significativement la capture de l'intention. Ce niveau de détail permet au DSL de comparer de manière fine les intentions des utilisateurs et de sélectionner le module de dialogue le mieux adapté, en se fondant sur le **rapprochement sémantique** des embeddings.

B. Composante symbolique : règles de cohérence contextuelle

Outre la représentation sub-symbolique, un agent conversationnel doit maintenir une **cohérence logique** dans le déroulement du dialogue. Pour ce faire, le DSL intègre un ensemble de **règles logiques** regroupé dans un bloc R_i associé à chaque entité ou état de dialogue. Ces règles, élaborées en fonction des contraintes du domaine, définissent des conditions d'activation ou de transition. Par exemple, dans un scénario de réservation d'hôtel, il est possible de spécifier que l'utilisateur doit d'abord fournir la date et le lieu avant de passer à la sélection du mode de paiement.

Formellement, la **synergie symbolique** peut être représentée par une fonction qui attribue un score en fonction de la conformité des règles, telle que

$$S_{\text{sym}}(R_i, R_j) = \begin{cases} 1, & \text{si les règles se valident mutuellement,} \\ 0, & \text{en cas de contradiction totale,} \\ s_{ij}, & \text{pour une satisfaction partielle,} \end{cases}$$

où $s_{ij} \in (0,1)$ représente un score intermédiaire mesurant le degré de compatibilité entre les blocs de règles associés aux entités \mathcal{E}_i et \mathcal{E}_j . Cette approche assure que l'agent conversationnel ne franchit jamais des étapes incohérentes, préservant ainsi la logique du dialogue et la non-répétition de questions redondantes.

C. Calcul de la Synergie Mixte et Mise à Jour des Pondérations

La force du DSL réside dans l'intégration des deux composantes précédentes afin de définir une **synergie hybride** qui guide la mise à jour des **pondérations** dans le SCN. La synergie hybride, qui évalue simultanément la **proximité sémantique** et la **compatibilité logique**, est formulée comme suit :

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où le paramètre $\alpha \in [0,1]$ ajuste l'importance relative accordée à la composante sub-symbolique par rapport à la composante symbolique. La **mise à jour** des pondérations du SCN se fait alors selon la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hybrid}}(i, j) - \tau \omega_{i,j}(t)],$$

où η représente le **taux d'apprentissage** et τ le **coefficent de décroissance**. Cette équation traduit la **plasticité locale** du système : lorsque la synergie hybride entre deux entités est élevée – indiquant que leurs intentions sont à la fois **sémantiquement proches** et **logiquement cohérentes** – la pondération $\omega_{i,j}$ est renforcée, ce qui favorise la formation d'enchaînements de dialogue pertinents. Inversement, des valeurs faibles de $S_{\text{hybrid}}(i, j)$ conduisent à un affaiblissement des liaisons, évitant ainsi des transitions incohérentes.

D. Bénéfices et Retours d'Expérience

L'approche hybride présente plusieurs **avantages** clés. La **robustesse linguistique** est assurée grâce à BERT, qui, par son fine-tuning, permet de gérer efficacement les variations dans les formulations textuelles, qu'il s'agisse de fautes d'orthographe, d'abréviations ou de synonymes. La **cohérence** du dialogue est maintenue par l'intégration d'un ensemble de **règles logiques** qui structurent la progression du dialogue et garantissent que les transitions respectent des contraintes préétablies. Par ailleurs, la combinaison de ces deux dimensions permet d'atteindre une **explicabilité accrue**, car chaque décision d'enchaînement peut être justifiée à la fois par une correspondance sémantique (le score sub-symbolique élevé) et par la validation d'un ensemble de règles (le score symbolique). Enfin, l'**évolutivité** du système est favorisée puisque le SCN, par le biais de la mise à jour continue des pondérations, s'adapte progressivement aux nouvelles données et aux révisions des règles, assurant une continuité du dialogue tout en permettant une mise à jour fluide des modules. En pratique, ces avantages se traduisent par un agent conversationnel capable

de gérer des requêtes complexes et variées tout en maintenant un fil de dialogue cohérent et explicable, ce qui est particulièrement pertinent dans des domaines spécialisés.

Conclusion

L'exemple d'un agent conversationnel qui combine **embeddings BERT** et **règles logiques** illustre parfaitement l'approche hybride du DSL. Dans cette architecture, la **composante sub-symbolique** assure une analyse fine et robuste de l'entrée utilisateur à travers des représentations vectorielles extraites par BERT, tandis que la **composante symbolique** garantit la cohérence contextuelle et le contrôle du dialogue en s'appuyant sur des règles logiques bien définies. La synergie hybride, exprimée par

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

permet de concilier la diversité des formulations textuelles et la nécessité d'un raisonnement explicite, conduisant à une mise à jour efficace des pondérations selon

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hybrid}}(i, j) - \tau \omega_{i,j}(t)].$$

Ce mécanisme d'**auto-organisation** permet à l'agent de sélectionner le module de dialogue le plus pertinent, d'assurer une transition cohérente dans le fil de la conversation et de fournir des justifications explicites quant à ses décisions. L'ensemble de ces éléments confère au chatbot un comportement plus **fiable, intelligent et adaptable** à de nouveaux contextes, en garantissant à la fois la flexibilité d'un modèle sub-symbolique et la rigueur d'un système symbolique.

3.6.3.3. Robotique sensorielle : capteurs sub-symboliques + représentation symbolique des actions/objets

L'association de **capteurs multiples** (caméra, LiDAR, capteur de force, etc.) avec un **ensemble de règles** décrivant les actions et les objets manipulés constitue un scénario riche pour le **Deep Synergy Learning (DSL)**. Le robot, plongé dans un environnement parfois incertain (lumière changeante, bruit sur les mesures), doit à la fois **analyser** les informations sub-symboliques fournies par ses capteurs et **respecter** la logique imposée par ses règles (scripts, protocoles, axiomes de sécurité). Cette double contrainte façonne un réseau **SCN** où chaque état ou action intègre à la fois un **embedding sub-symbolique** et un **bloc** de conditions symboliques, et où la **synergie** commande la mise à jour adaptative des liens.

A. Contexte : un robot multisensoriel

Imaginez un **robot logistique** évoluant dans un entrepôt, équipé de multiples capteurs :

- 185. **Caméra** fournissant un flux d'images. Un **CNN** (ou réseau de vision spécialisé) produit un vecteur $\mathbf{x}_{\text{cam}} \in \mathbb{R}^{d_1}$ décrivant l'information visuelle (objets détectés, caractéristiques de la scène).
- 186. **LiDAR** fournissant un nuage de points 3D. Un encodeur (type *PointNet* ou *autoencodeur 3D*) calcule $\mathbf{x}_{\text{LiDAR}} \in \mathbb{R}^{d_2}$.

187. **Capteur de force** détectant la pression exercée par un bras manipulateur, ou la force-torque globale, résumé dans un vecteur $\mathbf{x}_{\text{force}} \in \mathbb{R}^{d_3}$.

Par ailleurs, le robot se réfère à un **bloc symbolique** explicitant des règles de fonctionnement : “si la zone scannée est interdite, le robot doit contourner”, “si l’objet est fragile, la force de préhension doit rester sous un certain seuil”, “si la distance est < 0.5 m et que l’objet est inconnu, refuser l’avancée”. Ces règles logiques forment un corpus **R**, scindé éventuellement entre celles attachées à l’état (quelles conditions valent pour la position, la posture ?), et celles décrivant les actions (pré-requis pour saisir un objet, interdictions particulières, etc.).

Dans l’esprit d’un **DSL** hybride, chaque entité \mathcal{E}_i peut être un **état** sensoriel, un **objet**, ou une **action** potentielle, muni à la fois :

- d’un **embedding** sub-symbolique $\mathbf{x}_i \in \mathbb{R}^d$ (obtenu, par exemple, en concaténant $\mathbf{x}_{\text{cam}}, \mathbf{x}_{\text{LiDAR}}, \mathbf{x}_{\text{force}}$ ou un encodeur fusionné),
- d’un **bloc** de règles symboliques \mathbf{R}_i , exprimant, par exemple, la classe d’objet (fragile, volumineux), des restrictions de mouvement, ou un protocole d’action (*pick, place, push*).

B. Synergie entre capteurs et règles d’action

La fusion sub-symbolique permet d’évaluer la **proximité** ou la **compatibilité** entre deux états sensoriels, en calculant une distance ou une similarité sur leurs embeddings $\mathbf{x}_{\text{state1}}, \mathbf{x}_{\text{state2}}$. Parallèlement, si l’on compare un état \mathcal{E}_i et une action \mathcal{A} , la **synergie** inclut la vérification logique : l’action \mathcal{A} est-elle autorisée dans l’état \mathcal{E}_i ? Quelles conditions $R_{\mathcal{A}}$ imposent de s’appliquer (p. ex. “si l’objet est fragile, vérifier que la force < 10 N”)? On écrit :

$$S_{\text{hybrid}}(\mathcal{E}_i, \mathcal{A}) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_{\mathcal{A}}) + (1 - \alpha) S_{\text{sym}}(R_i, R_{\mathcal{A}}).$$

Ici, \mathbf{x}_i représente l’**embedding** agrégé de l’état sensoriel du robot (caméra, LiDAR, force), tandis que $\mathbf{x}_{\mathcal{A}}$ peut (ou non) exister selon la nature de l’action (certaines actions se dotent d’un vecteur sub-symbolique décrivant leurs contraintes). Les règles \mathbf{R}_i et $R_{\mathcal{A}}$ reflètent la logique symbolique, comme “ne pas déplacer un objet fragile au-delà de 2 m de hauteur” ou “nécessite zone autorisée = TRUE”.

Ce score de **synergie** sub-symbolique + symbolique conditionne l’**auto-organisation** dans le **SCN** : si \mathcal{A} est fréquemment validée au vu des capteurs (S_{sub} élevé) et respecte toutes les règles ($S_{\text{sym}} \approx 1$), alors le lien $\omega_{(i,\mathcal{A})}$ se **renforce**. Dans la mise à jour hebbienne :

$$\omega_{(i,\mathcal{A})}(t+1) = \omega_{(i,\mathcal{A})}(t) + \eta [S_{\text{hybrid}}(\mathcal{E}_i, \mathcal{A}) - \tau \omega_{(i,\mathcal{A})}(t)].$$

Si, en revanche, l’action apparaît incohérente (la force nécessaire dépasse un seuil, la zone est interdite, etc.), la pondération décroît, évitant que le robot ne réitère cette transition. Avec le temps, le **SCN** se stabilise en **clusters** ou en **chaînes** d’états-actions validées, traduisant un *plan* de manœuvre flexible qui reflète la fois la perception neuronale et les axiomes symboliques.

C. Exécution auto-organisée dans un DSL robotique

Dans un **SCN** robotique, chaque **état** ou **situation** \mathcal{E}_i encapsule un **embedding** (issu des capteurs) et un **bloc** de règles (posture autorisée, zone permise, classe d’objet détecté, etc.). Les **actions** (prendre, poser, avancer, reculer) apparaissent comme des noeuds à part ou comme des transitions

rattachées à un état. La force des liaisons $\omega_{(i,\mathcal{A})}$ se met à jour au fil des expériences, chaque essai ou simulation renforçant ou affaiblissant le couple (état, action) selon la synergie $\alpha S_{\text{sub}} + (1 - \alpha) S_{\text{sym}}$.

- **Sub-symboliquement**, si le robot “voit” un objet de forme X (via l’encodage de vision) et ce même objet a été identifié comme Y dans la base, on obtient un haut degré de **similarité**.
- **Symboliquement**, si une règle R impose de ne pas saisir cet objet sans vérifier la force, alors la conformité de la force de préhension agit sur S_{sym} .

Ainsi, la commande “saisir l’objet” depuis un état \mathcal{E}_i (caractérisé par un ensemble de mesures) ne sera validée que si le rapprochement sub-symbolique indique qu’il s’agit bien de l’objet visé (pas de confusion visuelle), et si la règle ne détecte pas d’incohérence (non-fragile, autorisé, zone libre, etc.). Le robot suit alors une **dynamique auto-organisée** — les **liaisons** ω se stabilisent sur des transitions conformes, et s’évanouissent pour celles jugées peu cohérentes.

Exemple

Un **robot picking** manipule divers objets sur un tapis roulant : boîtes en carton fragile, pièces métalliques lourdes, etc. Les règles imposent que si l’objet est fragile, la force de pincement doit demeurer < 10 N. Le sous-système sub-symbolique (caméra + force sensor) encode la forme et la masse apparente. Quand le robot tente l’action “pincer fort” pour un carton fragile, la synergie symbolique sera quasi nulle (contradiction avec la règle), même si sub-symboliquement la forme du carton correspond à un objet saisissable. Le poids $\omega_{(i,\text{pincerFort})}$ se réduit, tandis que “pincerDoucement” reçoit un renforcement positif. Progressivement, le **SCN** converge vers des gestes adaptés à la nature de chaque objet, tout en respectant la cohérence logicielle.

D. Bénéfices

188. **Robustesse**

Les capteurs sub-symboliques (vision, LiDAR, force) gèrent la **variabilité** de l’environnement. Si la luminosité change, le CNN peut toujours encoder un vecteur cohérent, et le LiDAR un nuage de points ; l'**auto-organisation** DSL poursuit son travail.

189. **Contrôle explicable**

Les **règles** rendent l’architecture transparente : on sait expliquer pourquoi une action est refusée (contrainte de sécurité) ou validée (capteurs confirmant l’alignement, règles satisfaites).

190. **Adaptation continue**

Si un nouveau capteur apparaît (caméra infrarouge, RFID), on l’ajoute à l’embedding sub-symbolique. Si une nouvelle règle de sécurité surgit, on l’insère dans la base symbolique. Le DSL régule la mise à jour de ω sans rupture totale du système.

191. **Scalabilité**

Dans de larges environnements industriels, un robot peut gérer un grand nombre d'états, tout en s'appuyant sur les mêmes principes de synergie : la matrice ω indique quels enchaînements (état → action → nouvel état) sont les plus performants.

Conclusion

En **robotique sensorielle**, la combinaison de **capteurs sub-symboliques** (extraits CNN, LiDAR, force-torque, etc.) et de **règles** (logique symbolique sur les actions, objets et protocoles) trouve un aboutissement naturel dans un **DSL multimodal**. Chaque **état** ou **action** hérite d'une représentation mixte (embedding vectoriel + bloc de règles), et la **synergie** décide, via la mise à jour des pondérations, quels chemins (état → action) forment un comportement valide. Cette architecture se prête tant à la **manipulation d'objets** (picking), qu'au **déplacement** (navigation), qu'à la **coordination** dans un entrepôt ou un atelier. Le résultat est un **réseau** capable de gérer en continu l'apprentissage sensoriel et la cohérence logico-fonctionnelle, assurant au robot une **adaptation** souple aux variations de l'environnement et un **respect** des règles imposées par la tâche.

3.6.4. Comparaison Expérimentale

Tout au long de ce chapitre (3), nous avons présenté divers **types de représentation** (sub-symbolique, symbolique, hybride) et plusieurs **approches** avancées (hypergraphes, fractales). Une question cruciale demeure : **comment** comparer expérimentalement les performances d'un **DSL** (Deep Synergy Learning) en fonction du **choix** de la représentation ? Dans cette section (3.6.4), nous abordons les éléments clés d'une **étude comparative** :

192. (3.6.4.1) L'**impact** du type de représentation sur la **qualité** de la synergie,
193. (3.6.4.2) Les **mesures** de clustering, le temps de convergence et la robustesse face à l'insertion de nouvelles entités,
194. (3.6.4.3) L'**approche HPC** pour gérer un grand nombre de dimensions ou un grand volume de données.

3.6.4.1. Impact du type de représentation (sub, sym, hybride) sur la qualité de la synergie

Dans le cadre d'un **Deep Synergy Learning (DSL)**, le choix de la représentation des entités constitue un paramètre déterminant pour la qualité de la synergie entre celles-ci, c'est-à-dire pour l'efficacité de l'auto-organisation du réseau de connexion. Selon qu'on opte pour une approche **sub-symbolique** (basée sur des **embeddings neuronaux**), une approche **symbolique** (fondée sur des **règles**, des **axiomes** ou des **ontologies**), ou une approche **hybride** qui combine ces deux modalités, la structure des liaisons $\{\omega_{i,j}\}$ ainsi que la formation des clusters peuvent varier de manière significative. En effet, ces différences se manifestent principalement en termes de **robustesse** face au bruit, de **lisibilité** ou **explicabilité**, de **capacité de raisonnement** et de **scalabilité**.

A. Comparaison entre les représentations sub-symbolique, symbolique et hybride

Dans une approche **sub-symbolique**, chaque entité \mathcal{E}_i est représentée par un **embedding** $\mathbf{x}_i \in \mathbb{R}^d$ issu d'un modèle d'apprentissage profond, lequel convertit des données brutes (images, extraits audio, phrases textuelles, etc.) en vecteurs qui condensent les caractéristiques les plus saillantes. La similarité entre deux entités est évaluée par des mesures classiques telles que la **similarité cosinus** ou la **distance euclidienne**. Par exemple, la similarité cosinus est donnée par

$$S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

ce qui reflète le degré de rapprochement angulaire entre \mathbf{x}_i et \mathbf{x}_j . Cette approche se caractérise par une **souplesse** et une **robustesse** intrinsèque, en particulier pour gérer un grand volume de données potentiellement bruitées ; toutefois, elle présente l'inconvénient d'être peu **explicable**, car il est difficile de déterminer explicitement quelles caractéristiques contribuent à la proximité entre deux embeddings, hormis par la valeur numérique obtenue.

En revanche, dans une approche **symbolique**, chaque entité \mathcal{E}_i est décrite par un **bloc** de règles, d'axiomes ou d'attributs logiques, noté R_i . La similarité entre deux entités se mesure alors par un score $S_{\text{sym}}(R_i, R_j)$ qui quantifie la **compatibilité** entre les ensembles de règles ou les structures logiques associées. Par exemple, une fonction de compatibilité peut être formulée comme

$$S_{\text{sym}}(R_i, R_j) = \begin{cases} 1, & \text{si les règles se valident mutuellement,} \\ 0, & \text{en cas de contradiction totale,} \\ S_{ij}, & \text{pour une compatibilité partielle,} \end{cases}$$

où S_{ij} est un score intermédiaire compris entre 0 et 1. Cette approche apporte une **explicabilité** élevée, puisque la logique sous-jacente offre la possibilité de retracer précisément pourquoi deux entités sont considérées comme compatibles, bien que le traitement symbolique soit souvent plus rigide et moins tolérant aux variations ou au bruit inhérent aux données.

Le **modèle hybride** vise à combiner le meilleur des deux approches précédentes. Dans ce cas, chaque entité est dotée à la fois d'un **embedding** \mathbf{x}_i et d'un bloc de règles R_i . La synergie globale entre deux entités \mathcal{E}_i et \mathcal{E}_j est alors définie par la formule

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où $\alpha \in [0,1]$ est un paramètre qui ajuste l'importance relative de la **dimension sub-symbolique** par rapport à la **dimension symbolique**. Ce schéma de fusion permet d'exploiter la **robustesse** des embeddings pour absorber le bruit et les variations lexicales, tout en bénéficiant du **raisonnement formel** et de la **trägabilité** qu'apporte la couche symbolique. Néanmoins, il nécessite un double stockage des informations (vecteur et règles) et des calculs supplémentaires, ce qui augmente la **complexité** de l'implémentation. La sélection judicieuse de α est cruciale pour obtenir le bon compromis entre **souplesse** et **précision**.

B. Métriques pour évaluer la qualité de la synergie

Pour évaluer la **qualité** des liaisons $\{\omega_{i,j}\}$ établies dans un DSL, il est important d'utiliser des **métriques** qui permettent de quantifier à la fois la **compacité** des clusters (mesurée par des indices

tels que l'indice de Silhouette ou le Davies–Bouldin Index) et la **séparation** entre ces clusters. Lorsque l'approche sub-symbolique est employée, la justification d'un regroupement se base essentiellement sur la proximité numérique entre les embeddings. En revanche, une approche hybride offre la possibilité de lier la **proximité sémantique** à des critères explicites, c'est-à-dire de fournir une explication formelle à la raison pour laquelle deux entités se trouvent dans le même cluster. Cette capacité à fournir une **explication** riche renforce la lisibilité des résultats, en permettant par exemple d'associer à un regroupement une liste de règles communes ou un indice de cohérence formelle qui soutient la formation des clusters.

C. Observations empiriques et compromis

Dans la pratique, l'approche **purement sub-symbolique** se distingue par sa **robustesse** et sa capacité à absorber de grandes quantités de données bruitées, tout en permettant un calcul rapide des similarités à l'aide de fonctions vectorielles telles que le cosinus ou la distance euclidienne. Toutefois, cette méthode reste limitée en termes d'**explicabilité**, car elle ne fournit qu'un score numérique sans indication sur les motifs formels qui sous-tendent la similarité. En revanche, une approche **purement symbolique** offre une **cohérence** et une **transparence** importantes, puisque chaque regroupement peut être expliqué par des règles logiques précises, mais elle est souvent rigide et sensible aux incohérences, rendant sa mise en œuvre coûteuse en calcul lorsque la base de règles est très large. Le modèle **hybride** se présente alors comme un compromis idéal, car il combine la **flexibilité** et la **robustesse** sub-symbolique avec la **rigueur** et la **clarté** symboliques. Cependant, cette approche hybride impose une augmentation du coût de stockage et de calcul, et requiert un réglage fin du paramètre α pour assurer le bon équilibre entre les deux composantes. En général, les expérimentations montrent qu'un DSL hybride parvient à générer des clusters d'entités plus **cohérents** et plus **explicables** que les approches purement sub-symboliques, tout en offrant une meilleure tolérance au bruit que les systèmes strictement symboliques.

Conclusion

L'impact du **type de représentation** sur la qualité de la synergie dans un DSL est considérable. L'approche **sub-symbolique** offre une grande **robustesse** et une facilité de calcul, mais demeure limitée en termes d'**explicabilité** et de capacité de raisonnement formel. À l'inverse, l'approche **symbolique** assure une **cohérence** et une **transparence** formelle, au prix d'une rigidité et d'un coût de calcul élevé lorsqu'il s'agit de traiter de grandes bases de règles. La solution la plus équilibrée consiste souvent à adopter un modèle **hybride** qui combine les forces de ces deux approches, en utilisant des **embeddings** pour capter la richesse sémantique et des **règles** pour assurer la cohérence contextuelle. La synergie hybride, définie par

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

et utilisée dans la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hybrid}}(i, j) - \tau \omega_{i,j}(t)],$$

permet de former des clusters d'entités qui sont à la fois **robustes** face aux perturbations et **explicables** par des critères formels. Le choix du type de représentation, ainsi que le réglage du paramètre α , doit être soigneusement adapté à la **nature** des données et aux exigences en matière d'**explicabilité** et de **raisonnement** dans le domaine d'application, de façon à optimiser la qualité des liaisons et la structure globale du DSL.

3.6.4.2. Mesures de clusters, temps de convergence, robustesse à l'insertion de nouvelles entités

Dans le cadre d'un **Deep Synergy Learning (DSL)**, l'évaluation du comportement du système ne se limite pas à l'analyse statique de la structure auto-organisée, mais englobe également des aspects dynamiques essentiels, notamment la **qualité des clusters**, le **temps de convergence** du réseau ainsi que la **robustesse** face à l'insertion de nouvelles entités. Ces trois dimensions permettent de mesurer de façon approfondie la stabilité et la scalabilité du **Synergistic Connection Network (SCN)**. L'analyse de ces critères s'appuie sur des **indices mathématiques** et des **métriques** qui quantifient la compacité des regroupements, le temps nécessaire à la stabilisation des pondérations, ainsi que la capacité du système à intégrer de nouvelles informations sans perturber la structure existante.

A. Mesures de qualité des clusters

Pour évaluer la qualité des clusters obtenus dans le SCN, on se réfère à des **indices de partition** qui mesurent la compacité intra-cluster et la séparation inter-clusters. Par exemple, l'**indice de Silhouette** est défini pour une entité \mathcal{E}_i par

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}},$$

où $a(i)$ représente la distance moyenne entre \mathcal{E}_i et les autres entités du même cluster, et $b(i)$ la distance moyenne entre \mathcal{E}_i et les entités du cluster le plus proche. Un score global s proche de 1 indique que les clusters sont bien définis et distincts. De même, l'**indice de Davies–Bouldin** s'exprime par

$$\text{DB} = \frac{1}{k} \sum_{i=1}^k \max_{j \neq i} \left\{ \frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right\},$$

où k est le nombre de clusters, σ_i la dispersion moyenne des entités dans le cluster i , et $d(c_i, c_j)$ la distance entre les centres c_i et c_j de clusters différents. Un indice DB plus faible signifie des clusters compacts et bien séparés. Par ailleurs, l'**indice Calinski–Harabasz** se définit comme

$$\text{CH} = \frac{\text{Tr}(B_k)}{\text{Tr}(W_k)} \times \frac{n-k}{k-1},$$

où $\text{Tr}(B_k)$ et $\text{Tr}(W_k)$ représentent respectivement la trace de la matrice de dispersion inter-clusters et intra-clusters, n étant le nombre total d'entités. Ces indices, appliqués aux représentations obtenues par le DSL – qu'elles soient sub-symboliques, symboliques ou hybrides – permettent d'évaluer de manière quantitative la **qualité des regroupements**. Dans un système hybride, par exemple, la combinaison d'un score basé sur la **proximité vectorielle** et d'un score issu des **règles logiques** offre une mesure plus riche et explicative, où l'on peut retracer l'origine d'un regroupement à la fois par la similarité des embeddings et par la cohérence des règles associées.

B. Temps de convergence

Le **temps de convergence** du SCN est une mesure dynamique qui reflète la rapidité avec laquelle les pondérations $\omega_{i,j}(t)$ se stabilisent. La mise à jour des poids dans le DSL s'exprime par l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ représente la synergie entre les entités, η le taux d'apprentissage et τ le coefficient de décroissance. La convergence se produit lorsque $\omega_{i,j}(t+1) \approx \omega_{i,j}(t)$ pour toutes les paires (i,j) . On peut définir un critère de convergence, par exemple, lorsque

$$\max_{i,j} |\omega_{i,j}(t+1) - \omega_{i,j}(t)| < \varepsilon,$$

où ε est un seuil de tolérance fixé. Dans les systèmes **sub-symboliques**, le calcul des similarités vectorielles est généralement rapide, ce qui permet une convergence relativement rapide, à condition que la dimension d ne soit pas excessive. En revanche, dans les systèmes **symboliques** et **hybrides**, les opérations logiques et la fusion des scores augmentent la complexité, rallongeant ainsi le temps de convergence. L'évaluation empirique du temps de convergence permet de mesurer la **vitesse d'auto-organisation** et d'ajuster les hyperparamètres η et τ de façon à optimiser la stabilisation des clusters.

C. Robustesse à l'insertion de nouvelles entités

Un critère fondamental pour l'évolutivité d'un DSL est la capacité du SCN à intégrer de nouvelles entités sans perturber la structure auto-organisée préexistante. Si une nouvelle entité \mathcal{E}_{new} dotée d'un embedding \mathbf{x}_{new} (et, dans le cas d'un modèle hybride, d'un bloc de règles R_{new}) est insérée, il est souhaitable que les nouveaux liens $\omega_{\text{new},i}$ soient établis de manière **locale** et que la structure globale ne soit pas remise en cause. En pratique, le DSL compare \mathbf{x}_{new} aux embeddings existants et, éventuellement, vérifie la compatibilité des règles associées pour calculer la synergie

$$S(\mathcal{E}_{\text{new}}, \mathcal{E}_i),$$

afin d'initier les pondérations $\omega_{\text{new},i}$. Cette approche permet de garantir une **insertion locale** sans nécessiter une re-synchronisation globale du SCN. La robustesse du système peut être évaluée par des métriques telles que le **changement moyen** des pondérations suite à l'ajout de nouvelles entités ou par des tests de stabilité de la structure de clusters, par exemple en mesurant l'évolution de l'indice de Silhouette ou du Davies–Bouldin Index avant et après l'insertion. Un DSL robuste sera capable de s'adapter de manière incrémentale, de sorte que l'ajout d'une entité provoque des ajustements locaux limités plutôt qu'une réorganisation massive du réseau.

Conclusion

L'impact du type de représentation sur la **qualité** de la synergie dans un DSL se mesure non seulement par la pertinence statique des clusters formés, mais également par la **vitesse de convergence** du SCN et sa **robustesse** face à l'insertion de nouvelles entités. Les **indices de partition** tels que l'indice de Silhouette, l'indice de Davies–Bouldin et l'indice Calinski–Harabasz permettent d'évaluer quantitativement la compacité et la séparation des clusters, tandis que des critères basés sur la variation maximale des pondérations servent à mesurer le temps de

convergence. De plus, la capacité d'un DSL à intégrer de nouvelles entités sans perturber la structure existante reflète sa scalabilité et son adaptabilité. Dans un système purement sub-symbolique, la rapidité du calcul vectoriel favorise une convergence rapide, mais l'explicabilité reste limitée. En revanche, un système symbolique offre une transparence totale mais est souvent plus rigide et coûteux en calcul. Le modèle hybride, quant à lui, tente de combiner les atouts des deux approches, obtenant généralement des clusters plus **cohérents** et **explicables** au prix d'un temps de convergence légèrement plus élevé et d'une gestion plus complexe lors de l'insertion de nouvelles entités. Ce compromis doit être évalué en fonction des exigences de l'application, afin de trouver le juste équilibre entre robustesse, rapidité et capacité à s'adapter aux évolutions du système.

3.6.4.3. Approche HPC si le DSL manipule un grand nombre de dimensions

Lorsqu'un **Deep Synergy Learning (DSL)** se déploie à large échelle, que ce soit en raison de la **dimension** élevée des vecteurs de représentation (embeddings sub-symboliques de plusieurs centaines ou milliers de composantes) ou du **nombre** massif d'entités manipulées, la charge de calcul pour maintenir la **synergie** $\{S(i,j)\}$ et la mise à jour itérative des $\{\omega_{i,j}\}$ peut rapidement croître au point de devenir prohibitive. Dans de telles configurations, une **approche HPC (High Performance Computing)** devient indispensable pour gérer de manière parallèle ou distribuée l'important volume d'opérations. L'idée consiste à recourir à des ressources matérielles avancées (GPU, clusters multiprocesseurs) et à des algorithmes spécifiques permettant de limiter l'impact de la croissance en $O(n^2d)$, où n est le nombre total d'entités et d la taille des embeddings.

A. Échelle et complexité dans un DSL de grande dimension

Si l'on considère un **DSL** manipulant $\mathbf{x}_i \in \mathbb{R}^d$ pour chaque entité \mathcal{E}_i , et qu'on souhaite évaluer une synergie sub-symbolique via un **produit scalaire** ou une **distance** de la forme

$$\|\mathbf{x}_i - \mathbf{x}_j\|^2 = \sum_{k=1}^d (x_{i,k} - x_{j,k})^2,$$

le coût se chiffre rapidement en $O(n^2d)$ si on compare toutes les paires (i,j) . Pour un simple examen exhaustif de quelques centaines de milliers d'entités (ou davantage) en embeddings de dimension 768, le nombre d'opérations requises s'avère colossal. La **composante symbolique**, lorsqu'elle est présente, peut ajouter une vérification de règles dont la **complexité** peut être encore plus élevée, parfois de nature exponentielle selon la logique mise en œuvre. Il devient alors essentiel d'exploiter des **moyens de calcul parallèle** et de **techniques d'approximation** pour préserver la faisabilité du DSL.

La mise à jour répétée des pondérations, exprimée par exemple sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

exige de réévaluer partiellement ou globalement la **synergie** $\{S(i,j)\}$ à chaque itération, d'où la nécessité d'un **High Performance Computing**. Sans ce levier, il est inenvisageable de mettre à jour une matrice de taille $O(n^2)$ dès que n atteint plusieurs centaines de milliers d'entités, surtout

si la logique impose un raisonnement ponctuel pour vérifier la compatibilité de certaines règles $\mathbf{R}_i, \mathbf{R}_j$.

B. Parallélisation sub-symbolique et distribution symbolique

La partie **sub-symbolique** (calcul de similarité entre embeddings) peut être largement parallélisée sur GPU ou CPU multicœurs. Le produit scalaire

$$\mathbf{x}_i \cdot \mathbf{x}_j = \sum_{k=1}^d (x_{i,k} x_{j,k})$$

peut se répartir par blocs matriciels ou par batchs, chaque nœud HPC traitant un sous-groupe de vecteurs. Dans un DSL géant, il est fréquent d'employer des algorithmes d'**approximate nearest neighbors** (Faiss, Annoy, HNSW) pour éviter un balayage exhaustif en $O(n^2)$. On réduit ainsi la construction du **SCN** aux seuls couples (i, j) dont la distance vectorielle est jugée potentiellement faible, ce qui autorise la parcimonie.

La partie **symbolique**, impliquant des règles ou des axiomes logiques \mathbf{R}_i , bénéficie parfois d'un déploiement sur des *reasoners* distribués ou des bases de connaissances échelonnées en grappes de serveurs. La compatibilité logique $\mathbf{R}_i, \mathbf{R}_j$ s'examine alors de manière partiellement parallèle, avec synchronisation des conclusions (contradiction, recouvrement) uniquement lorsqu'un sous-groupe d'entités sub-symboliquement proches exige une vérification fine. Cette stratégie "two-layers" consiste à filtrer sur la proximité neuronale et ne procéder au test symbolique complet que pour quelques paires pertinentes.

C. Organisation du SCN à grande échelle

Le **SCN** (Synergistic Connection Network) se construit de manière incrémentale et partiellement distribuée. Pour un lot de données, un ensemble de machines HPC calcule les voisainages k-NN dans l'espace \mathbb{R}^d . Chaque entité se voit attribuer une liste de candidats $\mathcal{N}_i \subset \{1, \dots, n\}$ correspondant à ses proches en embedding. Pour chaque couple $(i, j) \in \mathcal{N}_i$, on évalue la **composante symbolique** éventuelle $\mathbf{R}_i, \mathbf{R}_j$. On obtient ainsi

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

que l'on exploite pour initialiser ou ajuster la pondération $\omega_{i,j}$. La matrice $\{\omega_{i,j}\}$ reste parcimonieuse, car on n'active que les liens dépassant un certain seuil, ou répondant à une condition d'intérêt sub-symbolique ou symbolique. À l'issue de chaque vague d'itérations, un protocole de synchronisation (barrière, agrégation) peut consolider les résultats.

Dans un tel schéma HPC, la **synchronisation** demeure un facteur critique : si les nœuds traitent des partitions disjointes, la cohérence globale du DSL requiert un échange périodique des informations sur les couples inter-partitions. Les **paramètres** η (taux d'apprentissage), τ (décroissance) et α (poids sub-symbolique vs. symbolique) nécessitent des ajustements pour que l'on évite les oscillations dues à la latence ou à l'asynchronie.

Cette organisation rend cependant possible le traitement d'entités en nombre très élevé (jusqu'à des dizaines ou centaines de millions si l'on dispose d'assez de ressources), car on exploite des **librairies** HPC spécialisées (MPI, CUDA, OpenMP) et des *frameworks* (Spark, Hadoop) adaptés aux graphes distribués. Les opérations en $O(n^2d)$ sont ainsi contournées par des mécanismes de filtrage (approximate k-NN) et de parallélisation.

D. Gains, contraintes et perspectives

L'approche HPC ouvre la voie à un **DSL** capable de gérer simultanément de larges volumes de données sub-symboliques et un grand nombre de règles. On préserve la **richesse** de l'auto-organisation (renforcement local $\Delta\omega_{i,j}$) sans voir les temps d'exécution exploser, grâce à une exploitation massive du calcul parallèle. Le principal **gain** est la capacité à mettre à jour la synergie $\{\omega_{i,j}\}$ pour plusieurs millions d'entités en un temps acceptable, tout en intégrant les compatibilités logiques.

Les **limites** résident dans la **complexité de déploiement** : il faut configurer des clusters GPU ou multicœurs, mettre en place des reasoners distribués, synchroniser des partitions du SCN. Les **coûts** (financiers et en ingénierie) peuvent être considérables, réservant cette mise en œuvre à des instituts de recherche ou à des organisations gérant déjà des infrastructures HPC. De plus, l'approche reste conditionnée à des **heuristiques** de réduction (approximate k-NN, restrictions sur la portée des règles) pour rendre la dynamique $\omega_{i,j}(t)$ réellement scalable.

Néanmoins, ces méthodes **HPC** s'avèrent cruciales pour faire évoluer un **Deep Synergy Learning** vers des scénarios *big data*, où le DSL incorpore des embeddings dimensionnels imposants (issus de modèles neuronaux à grande échelle) et une **composante symbolique** non triviale. Il devient alors possible de concilier la **puissance** statistique (sub-symbolique) et la **cohérence** ou l'**explicabilité** (symbolique), malgré la croissance spectaculaire de la dimension d et du nombre n de nœuds, le tout orchestré par l'infrastructure HPC.

3.7. Conclusion

Au terme de ce **Chapitre 3**, nous avons examiné en profondeur la **représentation** des entités dans un cadre **DSL** (Deep Synergy Learning). Du choix d'une **approche sub-symbolique** (vecteurs, embeddings) à celui d'une **approche symbolique** (règles, ontologies), en passant par les **solutions hybrides** mêlant les deux, nous avons souligné à quel point la **qualité** de cette représentation influe directement sur la **synergie** calculée (et, partant, sur la structure du Synergistic Connection Network). Nous avons aussi abordé des **extensions avancées** (synergie n-aire, modèles fractals, multimodalité, etc.) et illustré ces principes à travers plusieurs **études de cas** (analyse audio-visuelle, agent conversationnel, robotique sensorielle).

3.7.1. Synthèse des Contributions du Chapitre

La question de la **représentation** d'une entité \mathcal{E}_i dans un **Deep Synergy Learning (DSL)** a émergé comme un élément fondamental, dont dépend l'intégralité de la **fonction** de synergie $S(i, j)$ et, par voie de conséquence, la dynamique de mise à jour $\omega_{i,j}(t)$. Les analyses conduites dans ce chapitre ont démontré l'extrême diversité des **approches** possibles, depuis des **embeddings** neuronaux (sub-symboliques) jusqu'aux **règles** logiques (symboliques), en passant par des schémas **hybrides** plus complexes.

A. Diversité des approches de représentation

L'exploration a porté sur trois **familles** principales. D'abord, la **représentation sub-symbolique** s'appuie sur un **embedding** dans \mathbb{R}^d , souvent calculé via des réseaux neuronaux (par exemple, BERT, GPT, ViT). Cette forme de modélisation met l'accent sur la **similarité vectorielle**, qu'il s'agisse d'une distance euclidienne ou d'un produit scalaire normalisé (cosinus). Une telle approche assure une **tolérance** appréciable au bruit et aux variations sémantiques, ce qui s'avère précieux pour de grandes bases de données textuelles ou visuelles. En revanche, l'absence de structure explicite rend l'interprétation plus délicate et limite les possibilités de **raisonnement** formel.

Une seconde voie consiste en des représentations **symboliques**, articulées autour de **règles**, **d'axiomes** ou **d'ontologies**. Cette méthode repose sur une **cohérence déclarative** : la synergie $S_{\text{sym}}(i, j)$ naît de la compatibilité ou de l'absence de contradiction entre les blocs logiques attachés à \mathcal{E}_i et \mathcal{E}_j . Pareille approche garantit une **transparence** conceptuelle, un **raisonnement** potentiellement exact, et la capacité de vérifier la non-contradiction de façon formelle. On lui reproche cependant une certaine rigidité face à un monde bruyant ou incertain, et une complexité algorithmique parfois élevée si la logique est riche (OWL, règles complexes).

Enfin, la troisième famille est **hybride**, associant le meilleur des deux précédentes : la robustesse statistique des embeddings sub-symboliques et la force explicative ou normative de la logique. Le calcul de synergie devient alors un **mélange**, par exemple

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

de sorte que l'on bénéficie d'une tolérance au bruit couplée à une explicabilité formelle. Cette combinatoire exige néanmoins un paramétrage plus lourd et un certain surcoût algorithmique.

Cette **diversité** d'approches reflète la pluralité des **domaines** auxquels se confronte un DSL. Dans certains cas (images, flux sensoriels massifs), l'embedding sub-symbolique domine ; dans d'autres (règlementaire, médical), les règles symboliques fournissent le cadre essentiel ; les scénarios mixtes justifient l'usage d'une représentation hybride.

B. Importance de la qualité de la représentation

L'acte même de calculer la synergie $S(i, j)$ dépend étroitement de la **qualité** de la représentation employée pour \mathcal{E}_i et \mathcal{E}_j . Si celle-ci est incomplète, trop approximative ou mal adaptée, la fonction de similarité/distance devient peu fiable, et les pondérations $\omega_{i,j}$ s'en ressentent : la **stabilité** de la mise à jour, la cohésion des clusters et l'émergence d'une structure interprétable dans le SCN se voient compromises. À l'inverse, une **représentation** minutieusement pensée – qu'il s'agisse d'un embedding « calibré » ou d'une base de règles sans redondances inutiles – renforce la **pertinence** des liens et la **lisibilité** du réseau.

Le chapitre a insisté sur le fait que le choix initial (sub-symbolique pur, logique pure, ou mixte) influence la manière dont évolue la distribution des $\omega_{i,j}$. Une représentation essentiellement vectorielle engendre une organisation liée aux distances dans \mathbb{R}^d (clustering géométrique), tandis qu'une représentation purement symbolique façonne des **groupes** souvent déterminés par la logique d'inclusion, de contradiction ou de compatibilité d'axiomes. Les systèmes hybrides donnent des clusters plus sélectifs : on exige à la fois une proximité numérique et une cohérence de règles, ce qui affine la formation d'ensembles stables.

C. Aspects avancés : synergie n-aire, fractalité, embeddings transformers

Au-delà de la traditionnelle comparaison binaire, le chapitre a montré que la **synergie** peut se généraliser à des **structures n-aires** (hypergraphes) où l'on compare simultanément $\{\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_k}\}$. Cette extension s'avère primordiale dès que l'on veut saisir la configuration conjointe de plusieurs modalités (ex. audio–vidéo–texte). Elle impose une représentation capable de décrire des entités multiples au sein d'un même **bloc** de calcul.

La **fractalité**, ou **auto-similarité**, constitue une autre dimension abordée : certaines représentations fractales (embeddings multirésolution, ontologies fractales) offrent une **compression** et une **récurrence** conceptuelle à plusieurs niveaux. Les bénéfices se révèlent dans des scénarios multi-échelle (Chap. 6) où l'on exploite la réitération systématique d'un même schéma à différentes granularités.

La question des **embeddings avancés** issus de **Transformers** (BERT, GPT, ViT) a souligné l'opportunité d'accéder à des **vecteurs** plus riches et plus contextuels, tout en augmentant la complexité de calcul et le besoin potentiel d'**approches HPC** pour gérer d'énormes ensembles de données.

Synthèse

L'axe majeur de ce **Chapitre 3** tient donc à la mise en évidence du rôle central de la **représentation** dans un DSL : toutes les formules $\omega_{i,j}(t + 1)$ et tous les patterns $\{\omega_{i,j}\}$ qu'on observe à la fin de la convergence découlent in fine de **comment** on encode chaque entité \mathcal{E}_i . L'**embedding** sub-symbolique, la base de **règles** logiques ou l'approche **hybride** imposent un langage qui influence :

- La robustesse au bruit (approximation, flexibilité neuronale),
- La cohérence explicable (raisonnement symbolique, absence de contradiction),
- La complexité (c'est-à-dire le coût temporel et matériel de calcul),
- La forme même des “clusters” ou “hyper-clusters” émergents.

Les **prochaines** parties (Chapitres 4, 5, 6) s'appuieront précisément sur cette **base** pour décrire la manière dont un **SCN** organise ses entités, étudie la dynamique de mise à jour (convergence, attracteurs) et peut s'étendre à des architectures distribuées ou multi-échelle, confirmant que les **choix** opérés dans le présent chapitre quant à la **représentation** constituent l'ossature de toute la suite du **Deep Synergy Learning**.

3.7.2. Limites et Pistes Futures

Malgré la diversité d'approches abordées au **Chapitre 3** – qu'il s'agisse de représentations **sub-symboliques** par embeddings neuronaux, de descriptions **symboliques** formées de règles ou d'axiomes, ou encore de modèles **hybrides** fusionnant les deux dimensions –, il demeure manifeste qu'aucune formulation n'est universellement optimale. Chaque méthode se plie à des **besoins** précis, des **données** particulières et des **ressources** de calcul données, si bien que le choix d'une représentation dépend étroitement du contexte. Il subsiste aussi de nombreux axes de **recherche**, qu'il s'agisse d'unification, de standardisation ou de perfectionnement des mécanismes de synergie au sein d'un **DSL** (*Deep Synergy Learning*).

A. Pas de représentation “unique” optimale

La première observation tient au fait qu'il n'existe pas de **format** apte à couvrir tous les cas d'usage avec la même efficacité. Les **embeddings** sub-symboliques (calculés dans \mathbb{R}^d) s'imposent lorsque l'on traite un grand volume de données bruitées, telles que des images, du son ou des segments textuels en langage naturel. Ils offrent une **robustesse** notable aux variations aléatoires et aux erreurs mineures, mais n'apportent pas la même explicabilité formelle que des règles logiques. À l'inverse, une approche strictement **symbolique** excelle pour des tâches exigeant un **raisonnement** strict, un contrôle de la **cohérence** ou une traçabilité explicite des décisions (domaine médical, légal, etc.). Cependant, elle se révèle rigide dès lors qu'apparaissent des informations ambiguës ou bruitées que la logique pure gère difficilement. L'option **hybride** tente de réconcilier ces deux mondes en calculant la **synergie** de façon mixte, ce qui peut apporter une meilleure **interprétabilité** tout en conservant une certaine tolérance aux imprécisions, au prix d'une infrastructure plus complexe et d'un temps de calcul plus important.

Un second aspect a trait aux **ressources** de calcul : des représentations sub-symboliques de grande dimension, ou des bases de règles logiques massives, peuvent exiger des **moyens** HPC (High Performance Computing) pour atteindre la performance requise. Dans certaines situations où l'on dispose de peu de capacité machine, il est parfois plus judicieux de réduire la complexité des modèles, soit en limitant la taille des embeddings, soit en simplifiant l'ontologie ou les règles symboliques, afin de maintenir le **DSL** dans des délais acceptables. Enfin, la **capacité d'évolution** du DSL joue un rôle : lors d'un apprentissage continu, de nouvelles entités et de nouveaux attributs apparaissent. Une représentation sub-symbolique est souvent plus facile à “raffiner” (par un fine-

tuning local) qu'une base de règles, dont la moindre modification peut exiger de recomposer la cohérence globale.

B. Recherche : standardisation des formats, outils pour la synergie multi-modal ou multi-domaine

Les différentes méthodes rencontrées dans le Chapitre 3 soulignent la nécessité de **standards** : chaque domaine (vision, son, ontologies, textes, robotique) recourt à un système distinct (RDF,OWL pour la partie symbolique, ONNX ou TF SavedModel pour la partie neuronale, etc.). L'élaboration d'un format **commun** – permettant de décrire à la fois un bloc symbolique \mathbf{R}_i et un embedding \mathbf{x}_i de façon unifiée – pourrait fluidifier l'intégration de modules hétérogènes dans un **DSL**. Cette standardisation faciliterait également la réutilisation de bibliothèques ou l'échange de modèles entre différents domaines.

Les expériences en **multimodalité** (où l'on fusionne plusieurs types de signaux) et en **multidomaine** (où un même SCN chapeaute des entités relevant de divers secteurs) suggèrent de développer des **kernels** ou des **fonctions de synergie** plus avancées, aptes à combiner plusieurs embeddings à la fois tout en tenant compte d'éventuelles règles logiques. Il devient alors crucial de définir :

$$S_{\text{hybrid}}(\mathcal{E}_i, \mathcal{E}_j) = \alpha_1 S_{\text{sub},1}(\mathbf{x}_{i,1}, \mathbf{x}_{j,1}) + \cdots + \alpha_m S_{\text{sub},m}(\mathbf{x}_{i,m}, \mathbf{x}_{j,m}) + (1 - \alpha_\Sigma) S_{\text{sym}}(R_i, R_j),$$

où $\{\mathbf{x}_{i,k}\}$ représentent différents **embeddings** issus de modalités ou de domaines divergents, et $\{R_i\}$ décrivent leurs attributs symboliques. La détermination optimale des $\{\alpha_k\}$ constitue un enjeu de calibration, pour équilibrer la part d'influence de chacune des modalités face à la couche logique. Il s'agit d'une **piste de recherche** encore largement ouverte.

C. Lien possible avec la sécurité (Chap. 11) et la convergence (Chap. 4, 7)

Les représentations discutées ici interfèrent avec d'autres volets du **Deep Synergy Learning**. Du point de vue de la **sécurité** (Chap. 11), la robustesse d'un modèle sub-symbolique s'avère cruciale pour repérer des signaux anormaux (par exemple dans des embeddings modifiés de manière adverse), tandis qu'une base de règles symboliques permet de justifier pourquoi l'on suspecte un comportement malveillant, ou de repérer un usage qui contrevient à certaines politiques. Un DSL "hybride" renforce donc cette défense en alliant détection statistique et validation logique.

Sur le plan de la **convergence** (Chapitres 4 et 7), la présence d'une logique symbolique, parfois discontinue (une légère modification peut changer radicalement la compatibilité de deux blocs de règles), peut introduire des points de non-lisséité dans la fonction de synergie globale, ce qui soulève des problèmes de stabilité et de temps de convergence. Les algorithmes d'optimisation et les garanties de stabilisation du SCN peuvent devenir plus complexes que dans un cadre sub-symbolique lisse (où la distance euclidienne donne une fonction continue). On entrevoit ici une **piste d'investigation** consistant à examiner la dynamique $\{\omega_{i,j}(t)\}$ lorsque la composante symbolique introduit des mises à jour discrètes et non linéaires.

Conclusion :

Les limites des différentes **représentations** – sub-symboliques, symboliques ou hybrides – montrent qu'il n'existe pas de format unique à même de répondre à l'ensemble des exigences. L'efficacité, la **scalabilité**, l'**explicabilité** et la **robustesse** au bruit forment autant de pôles contradictoires qu'il convient d'équilibrer. Les **pistes** futures se situent dans la standardisation, les

algorithmes de synergie multimodale, et l'exploration de modèles plus avancés, capables de manipuler des représentations hétérogènes avec plus de fluidité. Les chapitres ultérieurs approfondiront la **dynamique** de convergence, le **maillage** avec la sécurité ou la détection d'anomalies, et la mise en place **pratique** de ces représentations dans un DSL de grande échelle, confirmant que la structure même de la représentation demeure l'un des défis centraux de l'auto-organisation au sein du **Deep Synergy Learning**.

3.7.3. Liens avec les Chapitres Suivants

Les développements du **Chapitre 3** ont mis en lumière la **diversité** des **représentations** – sub-symboliques, symboliques, hybrides, voire plus avancées (n-aires, fractales) – et la **manière** dont elles peuvent se combiner dans un **DSL** (*Deep Synergy Learning*). Il reste cependant à comprendre **comment**, à partir de ces entités ainsi représentées, se déploie toute la **dynamique** d'auto-organisation et **comment** cette architecture s'inscrit dans un cadre global multi-échelle. Les chapitres ultérieurs (4, 5, 6) poursuivront précisément dans cette direction, en s'appuyant sur les notions introduites ici.

A. Chapitre 4 : Dynamique d'Auto-Organisation

Le **Chapitre 4** approfondira la **dynamique** propre au **SCN** (Synergistic Connection Network), c'est-à-dire le **mécanisme** régissant l'évolution des pondérations $\omega_{i,j}$ au fil des itérations. Cette dimension dynamique n'a été que brièvement abordée jusqu'ici, puisque le présent chapitre (3) se focalisait surtout sur la **nature** des entités $\{\mathcal{E}_i\}$ et la **forme** de leur représentation $\mathbf{r}(i)$. Les différentes **représentations** suggèrent en effet des **fonctions de synergie** distinctes, et donc des **équations** de mise à jour différentes.

Lorsqu'un DSL recourt à des embeddings sub-symboliques, la fonction $S_{\text{sub}}(i, j)$ repose sur une distance continue (euclidienne, cosinus), engendrant une mise à jour “lisse” qui tend à éviter les discontinuités. À l'inverse, une représentation symbolique peut amener des discontinuités logiques (contradiction vs. non-contradiction), rendant la dynamique plus délicate à analyser. Le **Chapitre 4** explicitera comment chaque type de représentation influe sur la **vitesse** et la **stabilité** de la convergence ($\Delta\omega_{i,j} \approx 0$), et examinera la notion d'**attracteurs** ou de **clusters** stables qui se forment lorsque le SCN atteint l'équilibre.

B. Chapitre 5 : Architecture SCN et Exploitation Pratique

Dans le **Chapitre 5**, on s'intéressera à la **structure globale** du **Synergistic Connection Network**, c'est-à-dire la manière dont on assemble concrètement les entités dans un graphe adaptatif. Les entités que l'on a décrites ici (qu'elles soient sub-symboliques, symboliques ou hybrides) y prennent la forme de *nœuds* ou de modules, reliés par des pondérations $\omega_{i,j}$. Ce chapitre décrira plus avant les **algorithmes d'exploitation** du SCN, par exemple pour :

- Partitionner le réseau en sous-groupes (macro-clusters),
- Mettre en place un système distribué ou modulaire,
- Gérer la dynamique en parallèle (HPC, chap. 3.6.4.3).

On clarifiera aussi les **formes** que peut prendre le SCN : simple graphe non orienté, hypergraphe, architecture hiérarchique où les liens s'organisent en couches. Les choix de **représentation** exposés au Chap. 3 déterminent en partie la **manière** dont on code les attributs ou règles associées aux nœuds, et influencent la **conception** même de l'architecture SCN.

C. Chapitre 6 : Multi-Échelle, Fractalité et Hiérarchies

Le **Chapitre 6** approfondira la **dimension multi-échelle** déjà évoquée. Les représentations fractales (section 3.6.2.2) et les constructions n-aires se prêteront à une **organisation** hiérarchique ou **fractal-like**. Il sera montré que l'on peut concevoir le DSL comme un système où plusieurs **niveaux** (micro, macro) interagissent, chacun recourant aux mêmes principes de **synergie**, mais adaptés à l'échelle correspondante. On y détaillera également comment un SCN fractal, ou multi-échelle, peut renforcer l'**auto-organisation**, et comment la formation simultanée de **clusters** locaux et globaux démultiplie la puissance de l'approche.

Le lien avec les représentations décrites au **Chapitre 3** est direct : un embedding fractal, par exemple, prend tout son sens dans un réseau multi-niveaux ; une ontologie fractale trouve également sa place dans la construction d'un SCN hiérarchique. Le **Chapitre 6** montrera comment ces idées se combinent pour donner au DSL une **cohérence** à la fois locale et globale, fortement inspirée de la théorie des systèmes complexes et de la cognition distribuée.

Conclusion :

La **question** de la représentation d'une entité – qu'elle soit sub-symbolique, symbolique, ou hybride – constitue le **fondement** des synergies calculées au sein du DSL et oriente la formation de **clusters** ou de **dynamiques**. Les chapitres suivants s'appuieront sur ces bases pour :

- Étudier la dynamique de mise à jour et de convergence (Chap. 4),
- Décrire la structure SCN et son exploitation pratique (Chap. 5),
- Aborder la perspective multi-échelle et fractale (Chap. 6).

Ce **Chapitre 3** se concluait sur l'idée qu'il n'y a pas de représentation unique idéale : ces considérations se répercuteront directement dans les analyses de la **dynamique**, de la **stabilité** et du **partage** d'information à l'échelle du réseau, constituant le cœur des chapitres qui suivent.

3.7.4. Vision Globale

Lorsqu'on aborde la construction d'un **Deep Synergy Learning (DSL)** et de son **Synergistic Connection Network (SCN)**, la notion de **représentation** occupe une place centrale. Cette représentation, qu'elle soit sub-symbolique, symbolique ou hybride, agit comme la **pierre angulaire** de toute la dynamique subséquente. Les analyses précédentes ont montré que la fonction $S(i,j)$, calculée entre deux entités \mathcal{E}_i et \mathcal{E}_j , dépend directement de la manière dont ces entités sont **modélisées**. Il s'avère ainsi que la richesse, l'adaptation ou au contraire la pauvreté d'un schéma de représentation se répercutent sur la qualité des *clusters*, la stabilité de la mise à jour $\omega_{i,j}(t)$, et la capacité du DSL à s'ajuster à un environnement complexe.

A. Impact sur la qualité de la dynamique et des clusters

Les expériences de ce chapitre confirment que si la représentation initiale est imprécise ou non adaptée, les résultats d'auto-organisation peuvent se révéler fragiles ou incohérents : les $\omega_{i,j}$ se stabilisent difficilement, et l'on observe des *clusters* dont la pertinence s'avère discutable. À l'inverse, une représentation bien pensée, qu'il s'agisse d'un **embedding** sub-symbolique pertinent ou d'un **ensemble** de règles logiques cohérentes, induit une fonction de synergie plus robuste. Cette robustesse se traduit par une formation de partitions ou de groupements lisibles, et par une dynamique de stabilisation plus rapide ou plus fiable.

Pour certains types de données très bruitées (images, enregistrements audio, textes informels), un embedding neuronal s'impose souvent comme un compromis efficace, offrant une métrique continue dans \mathbb{R}^d . Dans d'autres domaines plus réglementés (domaines médical, légal, etc.), la nature symbolique d'une représentation par **règles** ou **axiomes** facilite l'explicabilité et permet un raisonnement formel, tout en résistant mal au bruit et aux variantes imprévues. Les systèmes **hybrides** puisent leur force dans la combinaison de ces deux registres : ils assurent une flexibilité face à la variabilité sub-symbolique et un contrôle conceptuel par la logique, ce qui rejoue directement sur la *cohérence* des clusters et la *lisibilité* de la carte du réseau $\{\omega_{i,j}\}$.

B. Choix de modélisation pour un DSL lisible et performant

Le panorama offert par les sections précédentes rappelle qu'il **n'existe pas de solution unique** optimisant simultanément la rapidité d'exécution, l'explicabilité, la gestion du bruit et la capacité de raisonnement. Pour des applications volumineuses, les embeddings sub-symboliques dominent souvent en pratique, mais au détriment de la traçabilité formelle. Pour des tâches exigeant un raisonnement exact, les règles symboliques priment, moyennant parfois une lourdeur de calcul ou une rigidité de mise à jour. Les solutions **hybrides** visent un équilibre, souvent gagné au prix d'une architecture plus complexe et de ressources plus importantes.

Dans une architecture DSL couvrant plusieurs **modalités** (vision, audio, textes) et intégrant une **couche** de logique (règles métier, ontologies fractales), la diversité des **outils** (fractalité, synergie n-aire, HPC) aide à maintenir un certain degré de performance, même lorsque la dimension d des embeddings ou le nombre n d'entités croît fortement. Le système se révèle alors capable de faire face à des **scénarios** réalistes, où plusieurs types de données coexistent, où l'on exige à la fois la tolérance au bruit et l'aptitude à justifier des décisions.

C. Perspectives et intégration dans les chapitres suivants

Si l'on admet que la **représentation** est la fondation d'un **DSL**, il apparaît naturel que les chapitres ultérieurs (4, 5, 6) exploitent et prolongent ce travail. Le **Chapitre 4** décrira la *dynamique* de la mise à jour $\Delta\omega_{i,j}$, montrant comment la *nature* même de la fonction $S(i,j)$ influe sur la stabilité ou la vitesse de convergence. Le **Chapitre 5** s'attachera à la *structure* d'un SCN et à sa mise en œuvre pratique, en déclinant notamment les algorithmes de construction et d'exploitation. Enfin, le **Chapitre 6** abordera la *multi-échelle* et la *fractalité* dans l'organisation du DSL, qui constituent des cas d'application sophistiqués pour les représentations hybrides ou auto-similaires évoquées dans ce chapitre.

Conclusion :

En choisissant une **représentation** adéquate, on détermine en grande partie la **qualité** finale du **Synergistic Connection Network** et la **robustesse** de la mise à jour des $\{\omega_{i,j}\}$. Les options possibles (sub-symboliques, symboliques, hybrides, fractales) possèdent chacune leurs forces et leurs faiblesses, qu'il convient de confronter aux **besoins** (explicabilité, bruit, volumétrie) et aux **ressources** (capacité HPC, temps disponible). Le **Chapitre 3** a montré que la **pierre angulaire** d'un DSL réside dans la définition soigneuse de la manière dont chaque entité \mathcal{E}_i se décrit et se projette dans l'espace de la **synergie**. Les prochains chapitres s'enracineront dans ces notions pour analyser la dynamique, l'architecture et l'éventuelle hiérarchie multi-niveau, conférant à l'**auto-organisation** tout son potentiel d'adaptation dans un réseau de données riche et varié.

4.1. Introduction Générale	500
4.1.1. Rappel du Contexte et des Objectifs	500
4.1.2. Place et Contenu du Chapitre	501
4.2. Règles de Mise à Jour Fondamentales	506
4.2.1. Formulation Additive Classique	506
4.2.1. Formulation Additive Classique	509
4.2.2. Variantes Multiplicatives, Inhibition, etc.....	513
4.2.3. Cas Multi-Entités Hétérogènes	519
4.3. Émergence de Clusters et Attracteurs	526
4.3.1. Formation Spontanée de Groupes	526
4.3.2. Attracteurs Multiples et Bascules	529
4.3.3. Stabilisation vs. Dynamisme Continu.....	536
4.4. Oscillations, Pseudo-Chaos et Méthodes de Contrôle	548
4.4.1. Pourquoi des Oscillations ou un Pseudo-Chaos ?	548
4.4.2. Stratégies pour Rompre les Oscillations	553
4.4.3. Cas Stochastiques : Recuit, Perturbations	560
4.5. Héritage de la Physique Statistique et des Systèmes Dynamiques	567
4.5. Héritage de la Physique Statistique et des Systèmes Dynamiques	567
4.5.1. Notions de Contraction, Point Fixe, Attracteurs	567
4.5.2. Énergie ou Pseudo-Énergie	575
4.5.3. Renvoi vers Chapitre 2.3 et 2.4	582
4.6. Extraction et Visualisation des Clusters Émergents	588
4.6.1. Méthodes d'Observation	588
4.6.3. Évolutions dans le Temps.....	596
4.7. Exemples Concrets et Études de Cas.....	601
4.7. Exemples Concrets et Études de Cas.....	601
4.7.2. Cas Robotique ou Multi-Agent	608
4.7.3. Liens Pratiques	615
4.8. Stabilisation Avancée : Couplages avec Chapitre 5 (Architecture SCN).....	623
4.8.1. Rôle de l'Architecture	623
4.8.2. Gestions des Ressources et Threads	628
4.8.3. Transition.....	634
4.9. Conclusion et Transition	643

4.9.1. Synthèse	643
4.9.2. Ouverture.....	644
4.9.3. Place du Chapitre 4	645

4.1. Introduction Générale

Ce **Chapitre 4** s'inscrit dans la continuité des précédents, après avoir exploré, dans le Chapitre 3, les différentes manières de **représenter** les entités (sub-symbolique, symbolique, hybride) et de **calculer** la synergie $S(i,j)$, nous allons maintenant voir **comment** ces entités et synergies **s'animent** au sein d'un **réseau** (le Synergistic Connection Network, SCN). Au cœur de ce processus se trouve la **dynamique d'auto-organisation** : un mécanisme par lequel les pondérations $\omega_{i,j}$ entre entités évoluent (augmentent ou diminuent), sous l'effet de la synergie $S(i,j)$ et d'un terme de régulation (τ, η, \dots). C'est cette **dynamique** qui fait émerger, à partir de données initiales et d'un calcul de $S(i,j)$, des **clusters** cohérents ou des **patterns** inattendus — bref, la structure finale du réseau.

4.1.1. Rappel du Contexte et des Objectifs

Le **Chapitre 2** a posé les bases théoriques du **Deep Synergy Learning** en définissant les mécanismes fondamentaux par lesquels des entités interagissent via une **fonction de synergie**, notée $S(i,j)$. Ces premiers éléments ont mis en évidence la possibilité d'associer à chaque paire (i,j) un score susceptible d'indiquer le degré de similarité, de compatibilité ou de complémentarité entre les entités \mathcal{E}_i et \mathcal{E}_j . Le **Chapitre 3** s'est ensuite concentré sur la **représentation** de ces entités, il a explicité les cadres sub-symbolique et symbolique, proposé divers modes de calcul d'**embeddings** et évoqué la façon d'agréger des règles ou des concepts dans la synergie S .

À l'issue de ces deux chapitres, le **DSL** dispose ainsi d'une **mesure** $S(i,j)$ attribuant à chaque couple $(\mathcal{E}_i, \mathcal{E}_j)$ une **valeur** susceptible de capter leur affinité ou leur pertinence mutuelle. Cependant, l'essence même du Deep Synergy Learning ne réside pas dans la seule évaluation statique de ces synergies, la véritable originalité vient du **processus d'auto-organisation**, c'est-à-dire la façon dont un réseau dynamique, appelé **Synergistic Connection Network** (SCN), fait évoluer les **pondérations** $\omega_{i,j}$ en réponse à la synergie $S(i,j)$.

Cette évolution se formalise par une **règle de mise à jour**, par exemple

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

qui instaure un **équilibre** entre la **croissance** des liaisons en cas de synergie élevée et la **décroissance** imposée par un terme $\tau \omega_{i,j}(t)$. Le but d'un tel mécanisme n'est pas simplement de calculer un score, mais de laisser le réseau s'**organiser** pour faire émerger des **clusters** ou des **motifs** particuliers, reflétant la structure latente des données ou des règles. Certains liens se renforcent alors jusqu'à atteindre une valeur stable, tandis que d'autres s'étiolent et peuvent même s'annuler si la synergie ne justifie pas leur maintien.

Cette **auto-organisation** ne mène pas toujours à un unique état final ; elle peut aboutir à une **configuration stable**, à des **oscillations** persistantes ou à plusieurs **attracteurs** coexistant selon l'initialisation et les paramètres. L'objectif de ce **Chapitre 4** est de comprendre en détail comment l'on passe de la définition de la règle de mise à jour à l'analyse de comportements variés, émergence de clusters, stabilité ou instabilité, compétition latérale, et contrôle par **recuit simulé** ou par **inhibition** plus prononcée.

Ce chapitre approfondit donc plusieurs aspects. Il décrit dans un premier temps les **formes** possibles de mise à jour (additive, multiplicative ou hybrides), montrant comment chacune traduit un certain parti pris sur la plasticité des liens. Il discute ensuite des **régimes** de convergence et des méthodes pour les caractériser. Il met en perspective le rôle central de la **synergie** S , lorsque $S(i,j)$ demeure constant, il est aisément d'analyser la convergence locale vers $\omega_{i,j}^* = S(i,j)/\tau$. Quand au contraire $S(i,j)$ varie parce que la représentation des entités se réajuste (contexte d'**apprentissage continu**), la dynamique peut échapper à la stationnarité et exiger d'autres outils d'étude.

Cette continuité théorique et algorithmique entre la **définition** de S (Chap. 3) et la **mise à jour** de ω (Chap. 4) incarne le cœur conceptuel du **DSL**. Les prochaines sections illustrent pas à pas comment, à partir d'un simple "score de similarité" entre entités, un **réseau** adaptatif finit par révéler la structure profonde d'un ensemble de données, ou se réorganiser dans le temps pour s'accorder aux changements de flux et de représentations.

4.1.2. Place et Cotenu du Chapitre

Le présent chapitre s'attache à détailler la **dynamique d'auto-organisation** mise en œuvre dans un **DSL** (Deep Synergy Learning) une fois que la **synergie** $S(i,j)$ est définie pour chaque paire d'entités. Après avoir posé les bases théoriques (Chapitre 2) et présenté (Chapitre 3) la manière dont on construit les **représentations** et dont on calcule la fonction S , il s'agit maintenant d'étudier la **loi d'évolution** des pondérations $\omega_{i,j}(t)$. Cette loi est cruciale dans la mesure où elle dicte comment, à partir de synergies statiques ou variables, le **Synergistic Connection Network (SCN)** va former, détruire ou maintenir certains liens, jusqu'à aboutir à un réseau non supervisé structuré par la **coopération** ou la **compétition** des entités.

Dans la première partie (section 4.2), l'accent est mis sur les **règles de mise à jour** fondamentales. On examine notamment la forme **additive**, considérée comme la plus classique, où l'on ajoute ou retranche un incrément proportionnel à $S(i,j)$ et à la différence $\omega_{i,j}(t)$. On étudie également les variantes **multiplicatives**, dans lesquelles la croissance ou la décroissance dépend de $\omega_{i,j}(t)$ de manière proportionnelle, et les mécanismes d'**inhibition** qui introduisent une interaction compétitive entre liaisons. À chaque étape, il sera montré comment la croissance $\Delta\omega_{i,j}$ reflète la corrélation mesurée par $S(i,j)$ et la **décroissance** imposée par un terme de pénalisation $\tau \omega_{i,j}(t)$. On verra que la structure émergente résulte d'un **équilibre** entre ces forces de renforcement et de régulation.

Dans un second temps (sections 4.3 et 4.4), l'intérêt se porte sur la façon dont cette dynamique favorise la **formation spontanée de clusters**. Des entités disposant d'une synergie mutuelle élevée se trouvent amenées à consolider leurs liaisons $\omega_{i,j}$, tandis que les connexions de moindre synergie tendent à s'affaiblir ou à disparaître. On abordera la notion d'**attracteurs multiples**, qui s'exprime par le fait que différents points fixes (ou cycles) peuvent coexister, menant à des organisations distinctes selon l'initialisation. L'analyse des **oscillations** et des régimes de **pseudo-chaos** mettra en lumière la possibilité de comportements temporels plus complexes, dont il convient de maîtriser les effets si l'on souhaite obtenir un réseau stable. Les **méthodes de contrôle** (par **recuit simulé**, **inhibition latérale** ou **saturation**) seront ainsi présentées comme des outils pour résorber ou canaliser ces fluctuations, de sorte à garantir ou à forcer une organisation plus robuste.

Dans une dernière partie (sections 4.5 à 4.7), des **exemples concrets** seront décrits pour illustrer la **mise en œuvre** de ces règles de mise à jour dans des scénarios variés. Des simulations à taille réduite montreront la constitution progressive d'un ou plusieurs **clusters**, validant expérimentalement la théorie énoncée. Par la suite, des cas plus étendus, tels qu'une population d'agents en robotique ou un jeu de données multimodales, mettront en évidence le comportement du **DSL** face à un univers plus riche, où l'on observe l'**auto-organisation** continuer d'agir même lorsque la fonction S évolue au fil du temps.

En somme, ce chapitre entend dévoiler la “**vie interne**” du **SCN** en examinant pas à pas ce qui se produit lorsqu'on laisse $\omega_{i,j}(t)$ s'ajuster en fonction de $S(i,j)$. Le propos est de comprendre comment naissent des **clusters**, des **attracteurs stables** et comment gérer les **oscillations** ou **conflits** susceptibles d'apparaître. Cet approfondissement des phénomènes d'**auto-organisation** servira de fondement pour aborder, dans les chapitres ultérieurs, l'intégration plus large du **DSL** dans des architectures complexes et dans des applications en **temps continu**.

4.2.1.1. Équation Principale

La **mise à jour** de la **pondération** $\omega_{i,j}(t)$ peut se formaliser par l'équation suivante, considérée comme une forme canonique dans la dynamique du **Deep Synergy Learning** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Dans cette expression, la variable $\omega_{i,j}(t)$ représente la **force de lien** à l'itération t , tandis que la fonction $S(i,j)$ incarne la **synergie** instantanée entre les entités \mathcal{E}_i et \mathcal{E}_j . Le terme η fait office de **taux d'apprentissage**, déterminant la vitesse à laquelle la pondération se modifie, et $\tau \omega_{i,j}(t)$ assure une **décroissance** proportionnelle à la valeur courante de $\omega_{i,j}(t)$. L'itération s'effectue de manière **additive**, c'est-à-dire qu'à chaque pas, on ajoute un incrément linéaire dépendant de la différence $[S(i,j) - \tau \omega_{i,j}(t)]$.

La présence du facteur τ contribue à une forme de **relaxation** : si la **synergie** $S(i,j)$ ne parvient pas à compenser la décroissance due au terme $\tau \omega_{i,j}(t)$, la pondération $\omega_{i,j}(t)$ tend à diminuer au fil des itérations. À l'inverse, lorsque $S(i,j)$ est suffisamment élevée, le terme $\eta S(i,j)$ amène $\omega_{i,j}(t)$ à croître progressivement, favorisant un **lien** de plus en plus fort entre i et j .

Du point de vue analytique, il est souvent utile d'examiner le **point d'équilibre** où la variation $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ s'annule. Dans ce cas, on obtient l'égalité

$$\omega_{i,j}^* = \omega_{i,j}^* + \eta[S(i,j) - \tau \omega_{i,j}^*],$$

ce qui implique

$$S(i,j) - \tau \omega_{i,j}^* = 0 \Rightarrow \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette solution $\omega_{i,j}^*$ illustre la **zone de stabilisation** autour de laquelle la pondération tend à se fixer quand le schéma de mise à jour est laissé à lui-même. Ainsi, en présence d'une **forte synergie**, la valeur $\omega_{i,j}$ s'élève et se maintient à un niveau $S(i,j)/\tau$, tandis que dans le cas d'une synergie plus faible, la décroissance prend le dessus. Du point de vue **physique**, l'équation ci-dessus évoque un

gradient ou une forme de mouvement orienté vers un attracteur local, traduisant la logique de renforcement ou d'atténuation des liens selon l'intensité de la coopération sous-jacente.

4.2.1.2. Sens Interprétatif : Lien avec la “Descente d’Énergie” et Convergence vers $\omega^* = \frac{S}{\tau}$

Le **Deep Synergy Learning** peut être éclairé par une analogie physique et un principe de **descente d’énergie**. Dans le chapitre 2.4.3, il a été proposé de conceptualiser le réseau comme un système qui tend à minimiser une **pseudo-énergie** ou “fonction potentielle”, notée $J(\Omega)$. Une formulation simplifiée de cette énergie prend généralement la forme

$$J(\Omega) = - \sum_{(i,j)} \omega_{i,j} \mathbf{S}(i,j) + \frac{\tau}{2} \sum_{(i,j)} (\omega_{i,j})^2 + \dots$$

Dans cette expression, le terme $-\omega_{i,j} \mathbf{S}(i,j)$ favorise l’augmentation de la **pondération** $\omega_{i,j}$ lorsque la **synergie** $\mathbf{S}(i,j)$ est élevée, tandis que la partie $\tau/2 (\omega_{i,j})^2$ impose une **pénalisation** qui s’accroît avec la magnitude du lien. La **règle additive** décrite précédemment,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}(i,j) - \tau \omega_{i,j}(t)],$$

peut alors s’interpréter comme un schéma de **descente** (ou de “quasi-descente”) dans le gradient négatif de J . La composante $\mathbf{S}(i,j)$ joue un rôle de “force” qui pousse $\omega_{i,j}$ à croître, tandis que le terme $\tau \omega_{i,j}$ sert de “frein” ou de “relaxation” vers des valeurs plus modestes.

Lorsque la **synergie** $\mathbf{S}(i,j)$ reste **constante** au cours du temps, il est possible d’étudier la **convergence** de la dynamique en cherchant le point fixe $\omega_{i,j}^*$ pour lequel $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$. En imposant cette condition stationnaire, on obtient

$$\omega_{i,j}^* = \omega_{i,j}^* + \eta [\mathbf{S}(i,j) - \tau \omega_{i,j}^*] \Rightarrow \mathbf{S}(i,j) = \tau \omega_{i,j}^* \Rightarrow \omega_{i,j}^* = \frac{\mathbf{S}(i,j)}{\tau}.$$

Ce résultat indique que la **valeur d’équilibre** $\omega_{i,j}^*$ est directement proportionnelle à la **synergie** $\mathbf{S}(i,j)$ et inversement proportionnelle au paramètre τ , lequel représente la “raideur” ou l’intensité du freinage imposé à la croissance du lien. Aussi longtemps que η (taux d’apprentissage) est choisi de façon à éviter les oscillations excessives, on peut montrer que $\omega_{i,j}(t)$ se rapproche de $\omega_{i,j}^*$, ce qui confère à cet équilibre le statut de **point attracteur** local.

Sur le plan **énergétique**, cette situation correspond à un **minimum** de la pseudo-énergie J dans un scénario simple où la **synergie** ne dépend pas elle-même de $\omega_{i,j}$. Au sein du SCN, chaque lien se “met en conformité” avec la valeur $\mathbf{S}(i,j)/\tau$, de sorte que les liaisons **fortes** correspondent précisément aux paires (i,j) ayant une **synergie** élevée, et les liaisons **faibles** sont observées là où $\mathbf{S}(i,j)$ demeure basse. Cette mise à niveau progressive peut être considérée comme une “relaxation” du réseau, dans laquelle les forces de renforcement (la **synergie**) et de décroissance (le terme $\tau \omega_{i,j}$) parviennent à un **compromis**.

Dans le cas plus général où $\mathbf{S}(i,j)$ varie au fil des itérations (parce que la représentation des entités se modifie ou que des règles symboliques sont adaptées en continu), l’équilibre local $\omega_{i,j}^*$ se déplace

simultanément. La **règle additive** agit alors comme un processus d'**auto-organisation** qui s'efforce de suivre ces déplacements, sans nécessairement parvenir à une véritable convergence statique. Cette faculté d'adaptation souligne la souplesse du **DSL** dans des environnements évolutifs, tout en conservant l'idée d'une "descente d'énergie" vers des configurations où la **synergie** et la **décroissance** se neutralisent mutuellement.

4.2.1.3. Cas Stationnaire vs. Cas Dynamique : $S(i,j)$ constant ou recalculé à chaque itération

La **règle additive** introduite précédemment repose sur une mise à jour des pondérations $\omega_{i,j}$ proportionnelle à la **synergie** $S(i,j)$ et à un terme de **relaxation** $\tau \omega_{i,j}(t)$. Cette formulation se décline en deux scénarios : un **cas stationnaire**, où $S(i,j)$ demeure fixe dans le temps, et un **cas dynamique**, où $S(i,j)$ est recalculé d'une itération à l'autre, par exemple lorsque les représentations ou les règles sous-jacentes se modifient.

Dans le **cas stationnaire**, on suppose que la **synergie** $S(i,j)$ ne subit aucune variation au fil du temps. Les entités \mathcal{E}_i et \mathcal{E}_j conservent alors leurs caractéristiques ou leurs représentations sans être affectées par un nouvel apprentissage ou par l'introduction de changements contextuels. Le système $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ se comporte dès lors comme une **descente d'énergie** dirigée vers un point fixe, vérifiant $\omega_{i,j}^* = S(i,j)/\tau$. Lorsque la valeur $\omega_{i,j}(t)$ atteint ce rapport, il n'existe plus d'incitation à croître ni à décroître, dans la mesure où la composante $S(i,j)$ équilibre parfaitement la pénalisation $\tau \omega_{i,j}(t)$. D'un point de vue analytique, ce régime simplifie grandement l'étude de la **convergence** : le système tend à se stabiliser, et l'on peut ainsi prouver, sous certaines conditions (en particulier sur le pas d'apprentissage η), une convergence locale vers un attracteur. Un **DSL** exploité dans ce cadre stationnaire permet de clarifier l'effet de la synergie sur la formation de liens forts, tout en évitant les complications liées à la variation de $S(i,j)$.

Dans un **cas dynamique**, la **synergie** $S(i,j)$ subit un recalcul, potentiellement à chaque itération, de sorte qu'elle devient dépendante des modifications intervenues dans le réseau. Il se peut que les représentations vectorielles des entités $\mathbf{r}(i)$ et $\mathbf{r}(j)$ soient elles-mêmes ajustées par un processus d'apprentissage sous-jacent, ou que des blocs de règles symboliques évoluent (ajout/suppression d'axiomes, reparamétrage logique). Dans ces circonstances, $S(i,j)$ prend la forme d'une fonction $S_t(i,j)$ explicitement liée au temps t . La mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_t(i,j) - \tau \omega_{i,j}(t)]$$

opère alors sur un **paysage** en mouvement. La valeur de $\omega_{i,j}(t)$ doit sans cesse s'adapter, car l'équilibre instantané $S_t(i,j)/\tau$ se déplace à mesure que $S_t(i,j)$ évolue. Un **DSL** de ce type décrit plus fidèlement un système soumis à des variations externes ou internes (arrivée de nouvelles entités, apprentissage continu, ajustement de règles symboliques). Il n'est pas toujours garanti que l'on aboutisse à un point fixe au sens strict ; la dynamique peut osciller, changer de régime ou poursuivre un déplacement lent suivant la trajectoire imposée par $S_t(i,j)$.

En conclusion, la **formulation additive** se prête tout autant au **cas stationnaire** qu'à un **cas dynamique** plus réaliste. Lorsque $S(i,j)$ reste constant, l'analyse du point fixe $\omega_{i,j}^* = S(i,j)/\tau$ permet de mettre en évidence un comportement stable et interprétable. Si, au contraire, la **synergie** varie, la boucle de mise à jour se conçoit comme un **système adaptatif** en mouvement constant,

dont la convergence peut se révéler délicate à établir. Les sections suivantes approfondissent ces notions en examinant la stabilisation, la formation de **clusters**, et les phénomènes oscillatoires dans des configurations plus complètes du **Deep Synergy Learning**.

4.2. Règles de Mise à Jour Fondamentales

Après l'introduction générale (4.1) qui présente la dynamique d'auto-organisation comme le moteur principal du **DSL** (Deep Synergy Learning), nous abordons maintenant les **règles** mathématiques qui permettent de **faire évoluer** les pondérations $\omega_{i,j}$ dans le **SCN** (Synergistic Connection Network). Ces règles traduisent la manière dont un lien $\omega_{i,j}$ se **renforce** ou se **détériore** en réponse à la **synergie** $S(i,j)$, et elles constituent le **noyau** de l'auto-organisation.

4.2.1. Formulation Additive Classique

La première formulation, très fréquente dans la littérature et déjà mentionnée en chapitres précédents (2.3, 2.4.3), est dite **additive** : on met à jour $\omega_{i,j}(t)$ en lui **ajoutant** un terme proportionnel à la différence $[S(i,j) - \tau \omega_{i,j}(t)]$.

4.2.1.1. Équation Principale

La **mise à jour** de la **pondération** $\omega_{i,j}(t)$ peut se formaliser par l'équation suivante, considérée comme une forme canonique dans la dynamique du **Deep Synergy Learning** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Dans cette expression, la variable $\omega_{i,j}(t)$ représente la **force de lien** à l'itération t , tandis que la fonction $S(i,j)$ incarne la **synergie** instantanée entre les entités \mathcal{E}_i et \mathcal{E}_j . Le terme η fait office de **taux d'apprentissage**, déterminant la vitesse à laquelle la pondération se modifie, et $\tau \omega_{i,j}(t)$ assure une **décroissance** proportionnelle à la valeur courante de $\omega_{i,j}(t)$. L'itération s'effectue de manière **additive**, c'est-à-dire qu'à chaque pas, on ajoute un incrément linéaire dépendant de la différence $[S(i,j) - \tau \omega_{i,j}(t)]$.

La présence du facteur τ contribue à une forme de **relaxation** : si la **synergie** $S(i,j)$ ne parvient pas à compenser la décroissance due au terme $\tau \omega_{i,j}(t)$, la pondération $\omega_{i,j}(t)$ tend à diminuer au fil des itérations. À l'inverse, lorsque $S(i,j)$ est suffisamment élevée, le terme $\eta S(i,j)$ amène $\omega_{i,j}(t)$ à croître progressivement, favorisant un **lien** de plus en plus fort entre i et j .

Du point de vue analytique, il est souvent utile d'examiner le **point d'équilibre** où la variation $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ s'annule. Dans ce cas, on obtient l'égalité

$$\omega_{i,j}^* = \omega_{i,j}^* + \eta [S(i,j) - \tau \omega_{i,j}^*],$$

ce qui implique

$$S(i,j) - \tau \omega_{i,j}^* = 0 \Rightarrow \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette solution $\omega_{i,j}^*$ illustre la **zone de stabilisation** autour de laquelle la pondération tend à se fixer quand le schéma de mise à jour est laissé à lui-même. Ainsi, en présence d'une **forte synergie**, la

valeur $\omega_{i,j}$ s'élève et se maintient à un niveau $S(i,j)/\tau$, tandis que dans le cas d'une synergie plus faible, la décroissance prend le dessus. Du point de vue **physique**, l'équation ci-dessus évoque un **gradient** ou une forme de mouvement orienté vers un attracteur local, traduisant la logique de renforcement ou d'atténuation des liens selon l'intensité de la coopération sous-jacente.

4.2.1.2. Sens Interprétatif : Lien avec la “Descente d’Énergie” et Convergence vers $\omega^* = S/\tau$

Il est souvent utile d'analyser la **règle additive** comme une forme de **descente de gradient** dans un **paysage d'énergie**. Au chapitre 2.4.3, il a été proposé de considérer une **fonction d'énergie** $J(\Omega)$ dont l'un des termes dominants peut s'écrire sous la forme $-\sum_{i,j} \omega_{i,j} S(i,j)$. Dans un cadre simplifié, on peut poser

$$J(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2 + \dots$$

et vérifier que la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

correspond, dans ses grandes lignes, à la **descente** du gradient de J pour peu que $S(i,j)$ ne dépende pas de $\omega_{i,j}$.

La présence du **terme** $\tau \omega_{i,j}(t)$ peut se lire comme un “coût” qui **régularise la pondération** $\omega_{i,j}$. Plus la valeur de $\omega_{i,j}(t)$ est élevée, plus le produit $\tau \omega_{i,j}(t)$ vient la ramener à la baisse, sauf si la **synergie** $S(i,j)$ est suffisamment grande pour compenser ce mouvement. Dans un scenario stationnaire, où $S(i,j)$ est constante au cours du temps, la convergence locale de $\omega_{i,j}(t)$ vers un point fixe peut être étudiée en imposant $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*$. Il en résulte l'équation

$$\omega^* = \omega^* + \eta [S(i,j) - \tau \omega^*],$$

impliquant $S(i,j) = \tau \omega^*$. Autrement dit, on obtient

$$\omega^* = \frac{S(i,j)}{\tau}.$$

Cette **valeur** ω^* peut être considérée comme la **solution d'équilibre** : si la **règle additive** est appliquée de façon itérative et que $S(i,j)$ reste fixe, la pondération $\omega_{i,j}(t)$ évolue jusqu'à se stabiliser approximativement autour de $S(i,j)/\tau$. La signification profonde de ce résultat est qu'il y a un **équilibre** entre le **gain** (lié à $S(i,j)$) et la **pénalisation** quadratique (associée à $\tau \omega_{i,j}^2$), comme si le système « descendait » la fonction d'énergie J et s'arrêtait dans une vallée correspondant à un minimum local.

Dans un contexte plus général, où la **synergie** $S(i,j)$ varie au fil du temps, il n'existe pas forcément de convergence définitive vers une valeur fixe ω^* . Il est toutefois possible de constater que la **pondération** $\omega_{i,j}$ “suit” la synergie à travers une suite de régimes quasi-stationnaires, ou qu'elle oscille si les changements de $S(i,j)$ sont trop rapides ou si l'on choisit un **taux d'apprentissage** η trop grand. Le facteur τ , agissant comme un **terme de régularisation**, garantit néanmoins que

$\omega_{i,j}(t)$ ne croîtra pas indéfiniment en l'absence de feedback négatif ; il limite donc la possibilité d'un emballlement des poids et maintient le réseau dans un régime stable ou faiblement oscillant.

4.2.1.3. Cas Stationnaire vs. Cas Dynamique : $S(i,j)$ constant ou recalculé à chaque itération

La **règle additive** introduite précédemment repose sur une mise à jour des pondérations $\omega_{i,j}$ proportionnelle à la **synergie** $S(i,j)$ et à un terme de **relaxation** $\tau \omega_{i,j}(t)$. Cette formulation se décline en deux scénarios : un **cas stationnaire**, où $S(i,j)$ demeure fixe dans le temps, et un **cas dynamique**, où $S(i,j)$ est recalculé d'une itération à l'autre, par exemple lorsque les représentations ou les règles sous-jacentes se modifient.

Dans le **cas stationnaire**, on suppose que la **synergie** $S(i,j)$ ne subit aucune variation au fil du temps. Les entités \mathcal{E}_i et \mathcal{E}_j conservent alors leurs caractéristiques ou leurs représentations sans être affectées par un nouvel apprentissage ou par l'introduction de changements contextuels. Le système $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ se comporte dès lors comme une **descente d'énergie** dirigée vers un point fixe, vérifiant $\omega_{i,j}^* = S(i,j)/\tau$. Lorsque la valeur $\omega_{i,j}(t)$ atteint ce rapport, il n'existe plus d'incitation à croître ni à décroître, dans la mesure où la composante $S(i,j)$ équilibre parfaitement la pénalisation $\tau \omega_{i,j}(t)$. D'un point de vue analytique, ce régime simplifie grandement l'étude de la **convergence** : le système tend à se stabiliser, et l'on peut ainsi prouver, sous certaines conditions (en particulier sur le pas d'apprentissage η), une convergence locale vers un attracteur. Un **DSL** exploité dans ce cadre stationnaire permet de clarifier l'effet de la synergie sur la formation de liens forts, tout en évitant les complications liées à la variation de $S(i,j)$.

Dans un **cas dynamique**, la **synergie** $S(i,j)$ subit un recalcul, potentiellement à chaque itération, de sorte qu'elle devient dépendante des modifications intervenues dans le réseau. Il se peut que les représentations vectorielles des entités $\mathbf{r}(i)$ et $\mathbf{r}(j)$ soient elles-mêmes ajustées par un processus d'apprentissage sous-jacent, ou que des blocs de règles symboliques évoluent (ajout/suppression d'axiomes, reparamétrage logique). Dans ces circonstances, $S(i,j)$ prend la forme d'une fonction $S_t(i,j)$ explicitement liée au temps t . La mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_t(i,j) - \tau \omega_{i,j}(t)]$$

opère alors sur un **paysage** en mouvement. La valeur de $\omega_{i,j}(t)$ doit sans cesse s'adapter, car l'équilibre instantané $S_t(i,j)/\tau$ se déplace à mesure que $S_t(i,j)$ évolue. Un **DSL** de ce type décrit plus fidèlement un système soumis à des variations externes ou internes (arrivée de nouvelles entités, apprentissage continu, ajustement de règles symboliques). Il n'est pas toujours garanti que l'on aboutisse à un point fixe au sens strict ; la dynamique peut osciller, changer de régime ou poursuivre un déplacement lent suivant la trajectoire imposée par $S_t(i,j)$.

En conclusion, la **formulation additive** se prête tout autant au **cas stationnaire** qu'à un **cas dynamique** plus réaliste. Lorsque $S(i,j)$ reste constant, l'analyse du point fixe $\omega_{i,j}^* = S(i,j)/\tau$ permet de mettre en évidence un comportement stable et interprétable. Si, au contraire, la **synergie** varie, la boucle de mise à jour se conçoit comme un **système adaptatif** en mouvement constant, dont la convergence peut se révéler délicate à établir. Les sections suivantes approfondissent ces notions en examinant la stabilisation, la formation de **clusters**, et les phénomènes oscillatoires dans des configurations plus complètes du **Deep Synergy Learning**.

4.2.1. Formulation Additive Classique

La première formulation, très fréquente dans la littérature et déjà mentionnée en chapitres précédents (2.3, 2.4.3), est dite **additive** : on met à jour $\omega_{i,j}(t)$ en lui **ajoutant** un terme proportionnel à la différence $[S(i,j) - \tau \omega_{i,j}(t)]$.

4.2.1.1. Équation Principale

Dans le cadre d'un **Deep Synergy Learning (DSL)**, l'évolution de la **pondération** $\omega_{i,j}$ entre deux entités \mathcal{E}_i et \mathcal{E}_j se fait de manière itérative et s'exprime par une équation de mise à jour additive qui repose sur la synergie mesurée entre ces entités et un terme de décroissance régulateur. La forme canonique de cette équation est donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où chaque symbole revêt une signification particulière. Dans cette équation, $\omega_{i,j}(t)$ représente la **pondération** (ou force de lien) entre les entités \mathcal{E}_i et \mathcal{E}_j à l'itération t . Le terme $S(i,j)$ désigne la **synergie instantanée** ou **statique** entre ces deux entités, telle que définie dans le chapitre 3, et qui est souvent calculée à partir de mesures de similarité (par exemple, la similarité cosinus ou une distance transformée par un noyau gaussien). Le paramètre η correspond au **taux d'apprentissage**, un facteur qui contrôle la vitesse avec laquelle la pondération est mise à jour, tandis que le produit $\tau \omega_{i,j}(t)$ représente un **terme de décroissance** ou de relaxation, modulé par le paramètre τ qui détermine l'intensité de cette décroissance. Ainsi, un τ élevé aura pour effet de réduire plus fortement $\omega_{i,j}(t)$ lorsque $S(i,j)$ n'est pas suffisamment grand pour compenser cette décroissance.

Mathématiquement, l'évolution de $\omega_{i,j}$ est dite **additive** dans la mesure où, à chaque itération, on ajoute (ou l'on soustrait) un delta proportionnel à la différence $S(i,j) - \tau \omega_{i,j}(t)$. Cette formulation rappelle un mécanisme de **gradient descent**, où la pondération se déplace progressivement vers un point d'équilibre. En effet, en fixant la mise à jour à zéro dans l'état stationnaire, on obtient l'équation

$$\omega_{i,j}(t+1) \approx \omega_{i,j}(t) \Rightarrow S(i,j) - \tau \omega_{i,j}(t) \approx 0,$$

ce qui implique que, à l'équilibre, la pondération satisfait approximativement

$$\omega_{i,j} \approx \frac{S(i,j)}{\tau}.$$

On peut ainsi interpréter cette équation comme une **force attractive** qui tend à renforcer le lien entre deux entités lorsque la synergie est forte, tout en assurant une décroissance progressive du lien lorsque la synergie est insuffisante. Si $\omega_{i,j}(t)$ est initialement faible et que $S(i,j)$ est grand – c'est-à-dire si la synergie mesurée est élevée – la mise à jour incrémentale va entraîner une augmentation itérative de $\omega_{i,j}$, conduisant le système vers une zone de stabilisation où la valeur de la pondération converge vers l'équilibre $\omega_{i,j} \approx \frac{S(i,j)}{\tau}$. À l'inverse, si $\omega_{i,j}(t)$ est élevé mais que

$S(i,j)$ demeure faible, le terme négatif $-\eta \tau \omega_{i,j}(t)$ prédomine, provoquant une décroissance progressive de la pondération.

Cette règle de mise à jour est souvent considérée comme la plus simple et la plus intuitive pour modéliser la dynamique des liens dans un DSL. Elle offre une représentation claire de la manière dont un lien évolue en fonction de la synergie instantanée et de la décroissance proportionnelle à son état courant, permettant ainsi de capturer à la fois l'aspect d'**apprentissage** local et l'effet de **relaxation** qui prévient une croissance incontrôlée des pondérations. En résumé, l'équation principale

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

se présente comme un modèle simple et efficace pour l'auto-organisation d'un réseau de connexions, où la pondération entre deux entités évolue de manière additive en réponse à une synergie mesurée, convergeant vers une valeur d'équilibre définie par la relation $\omega_{i,j} \approx \frac{S(i,j)}{\tau}$.

4.2.1.2. Sens Interprétatif : lien avec la “descente d'énergie” (2.4.3), convergence vers $\omega^* = \frac{S}{\tau}$

Dans le cadre du **Deep Synergy Learning (DSL)**, l'équation de mise à jour des pondérations est souvent interprétée sous l'angle d'une **descente d'énergie**. Cette interprétation repose sur l'idée qu'une fonction d'énergie, notée $\mathcal{J}(\Omega)$, définie sur l'ensemble des pondérations Ω du réseau, diminue progressivement au cours des itérations d'apprentissage. Le chapitre 2.4.3 introduisait déjà le concept d'une fonction potentielle, par exemple sous la forme

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \dots,$$

où $S(i,j)$ représente la **synergie** entre les entités \mathcal{E}_i et \mathcal{E}_j . La règle de mise à jour canonique du DSL s'exprime par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le **taux d'apprentissage** et $\tau > 0$ représente le **coefficent de décroissance** qui agit comme un terme de régularisation. On peut considérer cette équation comme une descente de gradient appliquée à la fonction d'énergie \mathcal{J} dans un cas simplifié où $S(i,j)$ est indépendant de $\omega_{i,j}$. L'analogie avec la descente d'énergie se fait apparaître en remarquant que le terme $[S(i,j) - \tau \omega_{i,j}(t)]$ joue le rôle du gradient partiel de la fonction d'énergie par rapport à $\omega_{i,j}$, ce qui conduit, en cas de convergence, à un équilibre stationnaire.

Pour étudier cet équilibre, nous considérons la situation stationnaire où la mise à jour cesse, c'est-à-dire lorsque

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*.$$

En substituant cette condition dans l'équation de mise à jour, nous obtenons

$$\omega^* = \omega^* + \eta [S(i,j) - \tau \omega^*],$$

ce qui implique nécessairement que

$$S(i,j) - \tau \omega^* = 0.$$

Il en découle alors que la pondération stationnaire satisfait

$$\omega^* = \frac{S(i,j)}{\tau}.$$

Cette relation d'équilibre est particulièrement interprétable dans le sens où elle illustre comment la **force** du lien entre deux entités se stabilise à une valeur qui est directement proportionnelle à leur synergie $S(i,j)$ et inversement proportionnelle à la décroissance imposée par τ . On peut comprendre ce mécanisme en analogie avec un système physique où une particule se déplace dans un champ de potentiel et atteint un minimum d'énergie lorsque la force nette agissant sur elle devient nulle. Ici, le terme $\tau \omega_{i,j}(t)$ agit comme une force de **régularisation** ou un « coût » associé au maintien d'un lien fort, ce qui empêche les pondérations de croître indéfiniment en l'absence d'une synergie suffisante.

Dans un scénario idéal où $S(i,j)$ est constant dans le temps, la mise à jour additive converge de manière monotone vers $\omega^* = \frac{S(i,j)}{\tau}$. Toutefois, en pratique, $S(i,j)$ peut varier d'une itération à l'autre, ce qui rend la convergence plus complexe et peut empêcher l'atteinte d'un équilibre strict. Néanmoins, l'idée centrale demeure que la pondération $\omega_{i,j}$ tend à se "caler" sur une valeur qui reflète la force moyenne de la synergie entre les entités, assurant ainsi une **auto-organisation** harmonieuse du réseau.

Le terme de décroissance $\tau \omega_{i,j}(t)$ joue également un rôle essentiel en agissant comme un mécanisme d'**anti-explosion**. Sans ce terme, si $S(i,j)$ était constamment élevé, la pondération pourrait croître de façon indéfinie, ce qui serait incompatible avec la stabilité du système. La présence de ce terme garantit que, même en cas de synergie élevée, la pondération est ramenée à des valeurs réalistes, modélisant ainsi un compromis entre la **coopération** entre entités et le **coût** de maintenir des liens trop forts.

En résumé, la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

peut être interprétée comme une descente de gradient sur une fonction d'énergie $\mathcal{J}(\Omega)$, où la convergence vers l'équilibre stationnaire $\omega^* = \frac{S(i,j)}{\tau}$ illustre la manière dont la synergie entre deux entités est équilibrée par la décroissance. Cette interprétation confère à la mise à jour un sens interprétatif profond, reliant le comportement numérique du DSL à des principes physiques tels que la minimisation de l'énergie, et offre ainsi une explication intuitive de la **stabilité** des liens dans le SCN.

4.2.1.3. Cas Stationnaire vs. Cas Dynamique : $S(i,j)$ constant ou recalculé à chaque itération

Dans l'analyse d'un **Deep Synergy Learning (DSL)**, l'évolution des **pondérations** $\omega_{i,j}$ repose sur la synergie $S(i,j)$ entre les entités, laquelle peut être considérée soit comme une valeur **constante**

au cours du temps, soit comme une grandeur **dynamique** recalculée à chaque itération. Ces deux cas présentent des comportements distincts qui influencent la convergence du système et la stabilité de l'**auto-organisation** dans le **Synergistic Connection Network (SCN)**.

Lorsqu'on adopte le **cas stationnaire**, on suppose que $S(i, j)$ demeure fixe. Dans ce contexte, les représentations des entités, qu'elles soient sub-symboliques ou symboliques, ne subissent pas de modifications significatives d'une itération à l'autre et aucune mise à jour contextuelle n'est introduite. La règle de mise à jour, qui s'exprime sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

se traduit alors par une dynamique qui converge vers un **point fixe** ou un attracteur stable. Pour démontrer ce comportement, il suffit de poser que, dans l'état stationnaire, $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega^*$. En substituant dans l'équation, on obtient

$$\omega^* = \omega^* + \eta[S(i,j) - \tau \omega^*].$$

Cette égalité est satisfaite si et seulement si

$$S(i,j) - \tau \omega^* = 0,$$

ce qui conduit immédiatement à la relation

$$\omega^* = \frac{S(i,j)}{\tau}.$$

Cette convergence vers $\omega^* = \frac{S(i,j)}{\tau}$ permet d'interpréter le système en termes d'**équilibre énergétique** ou de descente d'énergie, où la pondération atteint une valeur d'équilibre déterminée par la synergie mesurée et par le coefficient de décroissance. Dans ce cas, l'ensemble du réseau se stabilise et les transitions entre états restent fixes, ce qui facilite l'analyse théorique de la convergence et permet de servir de référence pour comparer des systèmes dans des environnements plus dynamiques.

Dans le **cas dynamique**, en revanche, la synergie $S(i, j)$ n'est pas considérée comme constante, mais est recalculée à chaque itération. Cette situation se produit lorsque les représentations des entités évoluent au fil du temps en raison d'un apprentissage continu ou de modifications contextuelles, telles que des mises à jour des embeddings ou des ajustements dans les blocs symboliques. En effet, lorsque le DSL intègre un **apprentissage adaptatif**, le score $S(i, j)$ devient une fonction temporelle, que l'on peut noter $S(i, j, t)$. La règle de mise à jour s'écrit alors

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j,t) - \tau \omega_{i,j}(t)],$$

et il apparaît un couplage itératif dans lequel la pondération $\omega_{i,j}$ s'ajuste en même temps que $S(i, j)$ évolue. En pratique, cette interaction rend le système plus complexe, car les mises à jour des représentations sub-symboliques (par exemple, via le fine-tuning ou la réorganisation des données) entraînent des variations dans $S(i, j, t)$ qui peuvent perturber la convergence du réseau. Le système se comporte alors comme un **système adaptatif** où aucune convergence stationnaire absolue n'est garantie, et l'auto-organisation devient un processus continu dans lequel la structure du réseau est sans cesse réévaluée et reconfigurée. Un tel processus peut engendrer des oscillations ou des

réajustements brusques dans les pondérations, surtout si les variations de $S(i,j,t)$ sont significatives d'une itération à l'autre. Pour pallier ces effets, il est souvent nécessaire d'introduire des mécanismes de **lissage** ou de **verrouillage partiel**, afin que la transition entre différentes valeurs de $S(i,j,t)$ soit plus progressive et que l'ensemble du système reste stable malgré des mises à jour fréquentes.

En définitive, la **formulation additive** qui sous-tend la mise à jour des pondérations dans le DSL repose sur l'équilibre entre le **renforcement** induit par $S(i,j)$ et la **relaxation** imposée par $\tau \omega_{i,j}(t)$. Dans le cas stationnaire, où $S(i,j)$ est constant, la dynamique converge de manière prévisible vers $\omega^* = \frac{S(i,j)}{\tau}$, offrant une interprétation claire en termes de descente d'énergie. Dans le cas dynamique, le recalcul constant de $S(i,j,t)$ fait du processus une boucle itérative plus complexe, qui nécessite une analyse supplémentaire de la stabilité et de la formation des clusters dans le SCN. Cette distinction fondamentale entre un $S(i,j)$ stationnaire et un $S(i,j,t)$ dynamique est au cœur de la compréhension du comportement du DSL, et constitue une base de référence pour étudier l'impact de l'évolution des représentations sur la qualité de l'auto-organisation.

Ainsi, le cas stationnaire offre une situation de référence idéale pour l'analyse théorique de la convergence, tandis que le cas dynamique reflète la réalité d'un système en apprentissage continu, dans lequel la structure du réseau se reconfigure en permanence. Ces deux cas permettent de quantifier l'influence de la **variabilité** des représentations sur la stabilité des pondérations, et servent de point de départ pour développer des stratégies visant à optimiser la **robustesse** du DSL en environnement non stationnaire.

4.2.2. Variantes Multiplicatives, Inhibition, etc.

Bien que la **formulation additive** (4.2.1) soit souvent considérée la plus "classique" pour un **DSL** (Deep Synergy Learning), il existe d'autres **règles** de mise à jour des pondérations $\omega_{i,j}$. Certaines de ces variantes cherchent à introduire plus de **dynamisme** (croissance multiplicative), d'autres à intégrer de l'**inhibition** compétitive, ou encore à imposer des **mécanismes** de saturation pour prévenir une montée incontrôlée de certains liens.

4.2.2.1. Multiplicative : $\omega_{i,j}(t+1) = \omega_{i,j}(t) [1 + \eta (\dots)]$

Dans cette variante, plutôt que d'ajouter un delta à $\omega_{i,j}(t)$ (comme dans la règle additive), on multiplie $\omega_{i,j}(t)$ par un **facteur** dépendant de la synergie $S(i,j)$ et de la décroissance τ . Autrement dit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \left[1 + \eta \left(\alpha S(i,j) - \beta \tau \omega_{i,j}(t) \right) \right],$$

ou une forme voisine (il existe plusieurs formulations possibles). L'idée générale est que la **mise à jour** se fait en **proportion** de $\omega_{i,j}(t)$, plutôt que par un **delta** absolu.

Mise à jour multiplicative : motivations, avantages et risques

Dans certains scénarios de Deep Synergy Learning (DSL), on abandonne la règle additive traditionnelle (sous la forme $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta$) au profit d'une **mise à jour multiplicative** permettant de modéliser une **croissance proportionnelle** ou quasi exponentielle. Concrètement, au lieu d'ajouter un terme Δ , on applique un **facteur** $[1 + \Delta]$, qui multiplie la valeur actuelle de $\omega_{i,j}(t)$. Ainsi, si la pondération $\omega_{i,j}(t)$ est déjà élevée, une légère variation de Δ peut se traduire par une forte augmentation, ce qui peut accélérer la structuration du réseau en mettant rapidement en évidence certains **liens dominants**.

Sur le plan analytique, on peut illustrer cette approche par la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) [1 + \eta S(i,j) - \eta \tau \omega_{i,j}(t)].$$

Dans cette équation, η représente un taux d'apprentissage, τ un terme de régulation ou d'oubli, et $S(i,j)$ la synergie calculée entre les entités \mathcal{E}_i et \mathcal{E}_j . Si $\omega_{i,j}(t)$ est déjà non négligeable, et que la partie $\eta S(i,j)$ dépasse $\eta \tau \omega_{i,j}(t)$, le produit dans la parenthèse demeure supérieur à 1, ce qui renforce massivement la pondération. C'est en cela que l'on parle de **dynamique non linéaire** : plus un lien est fort, plus il est susceptible d'augmenter vite, de sorte que l'on observe fréquemment un phénomène d'**auto-amplification**.

Cette caractéristique s'avère intéressante dès lors que l'on souhaite modéliser ou encourager une **coopération** exponentielle : par exemple, deux entités qui collaborent déjà fortement verront leur liaison s'intensifier encore, entraînant la création de **clusters** en "tout ou rien" (certains liens "explosent", tandis que d'autres chutent). Ce schéma **multiplicatif** reflète également divers comportements biologiques ou écosystémiques, où la vitesse de croissance dépend de la "valeur" présente (lois de populations, mécanismes de reproduction, etc.). En revanche, il comporte un **risque de divergence** : si $\omega_{i,j}(t)$ devient trop important et n'est pas freiné par le terme $\eta \tau \omega_{i,j}(t)$, la pondération peut croître sans limite, perturbant profondément la structure du réseau. Pour éviter ce problème, on introduit souvent des dispositifs de **saturation** (voir section 4.2.2.3) limitant la valeur maximale de $\omega_{i,j}$, ou des mécanismes d'**inhibition compétitive** (section 4.2.2.2) afin d'empêcher qu'un petit nombre de liens n'accapare l'essentiel du "poids" synergique.

En comparaison, la **règle additive** classique conduit plutôt à un point fixe $\omega^* = S(i,j)/\tau$ en l'absence d'autres rétroactions, et autorise un contrôle plus fin sur la progression des pondérations. La version multiplicative, elle, **accélère** la différenciation des liens, mais exige une **vigilance accrue** quant aux risques d'**explosion**, d'**instabilité** et de forte **sensibilité** aux variations de la synergie. En somme, choisir un schéma multiplicatif revient à opter pour un modèle dans lequel la mise en valeur rapide de certaines liaisons est un avantage, tout en s'assurant qu'un encadrement (clipping, inhibition) est bien en place pour maintenir la cohérence globale du réseau.

4.2.2.2. Inhibition Compétitive : ajout d'un terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ ou autre forme

Dans certains systèmes adaptatifs (biologiques, cognitifs, ou plus généralement en **self-organisation**), l'**inhibition compétitive** joue un rôle déterminant pour **réguler** la croissance simultanée de liens. L'idée est de **limiter** la progression de plusieurs connexions parallèles, incitant ainsi certaines à **dominer** au détriment d'autres. Cela évite un "tout ou rien" généralisé, ou la situation où trop de liens forts coexistent et saturent la dynamique. Concrètement, dans un **DSL**

(Deep Synergy Learning), on peut ajouter au terme de mise à jour un **facteur d'inhibition** qui pénalise $\omega_{i,j}$ lorsque, pour la même entité \mathcal{E}_i , d'autres pondérations $\omega_{i,k}$ (avec $k \neq j$) sont élevées.

A. Principe de l'inhibition compétitive

Une règle additive (ou multiplicative) peut être agrémentée d'un **terme** d'inhibition de la forme :

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

où $\gamma > 0$ est un coefficient d'inhibition, et la somme $\sum_{k \neq j} \omega_{i,k}(t)$ représente la “masse” totale des liens que \mathcal{E}_i entretient avec d'autres entités $k \neq j$.

En pratique, ce terme vient **s'ajouter** (avec un signe négatif) à la mise à jour de $\omega_{i,j}$. Plus \mathcal{E}_i possède des liens forts avec d'autres \mathcal{E}_k , plus la **progression** de $\omega_{i,j}$ s'en trouve **limitée**.

L'**idée intuitive** derrière ce mécanisme repose sur une forme de **compétition** : si une entité \mathcal{E}_i investit déjà fortement dans certains liens $\omega_{i,k}$, ses “ressources” disponibles pour renforcer de nouveaux liens $\omega_{i,j}$ sont de facto limitées. D'un point de vue **biologique**, ce principe évoque un phénomène de “**winner-take-all**” partiel ou d’“**inhibition latérale**”, couramment décrit dans les modèles neuronaux, où la somme d'activité globale d'un neurone (c'est-à-dire la somme de ses liens) restreint sa capacité à s'activer simultanément vers d'autres connexions.

Exemple

Supposez une règle additive standard :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Quand $\sum_{k \neq j} \omega_{i,k}(t)$ est grand, cela **déourage** l'augmentation de $\omega_{i,j}$, forçant \mathcal{E}_i à “faire un choix” parmi plusieurs liaisons concurrentes.

B. Avantages de l'inhibition compétitive

L'un des premiers bénéfices consiste à éviter toute “dispersion” : sans mécanisme d'inhibition, une entité \mathcal{E}_i risquerait de créer de nombreux liens modérément forts avec un large éventail d'autres entités.

En introduisant une compétition, on **focalise** l'attention sur quelques liaisons dominantes, tout en réduisant l'intensité des autres ; il en résulte la formation de **clusters** plus nettement délimités.

Par ailleurs, l'inhibition compétitive aide à **contrôler la croissance simultanée** des connexions : même si la synergie $S(i,j)$ se révèle positive pour plusieurs entités, le réseau ne permet pas à toutes ces liaisons de croître en parallèle, introduisant une forme de **sélection** où seules les connexions les plus pertinentes se renforcent réellement.

Enfin, cette approche s'appuie sur des principes **biomimétiques**, rappelant l'**inhibition latérale** observée dans le cerveau, où un neurone en forte activité limite celle de ses voisins.

Dans un **DSL**, ce phénomène favorise la structuration de clusters plus précis et encourage la spécialisation, chaque entité s'orientant principalement vers les connexions qui maximisent la qualité de la coopération.

C. Limites et paramétrage

L'**inhibition compétitive** dans un réseau requiert une attention particulière quant à ses limites et à son paramétrage. D'abord, il existe un **risque de désactivation excessive** : si le coefficient γ qui contrôle la force de l'inhibition est trop élevé, il peut empêcher la croissance significative de tout nouveau lien, menant ainsi à un réseau **faiblement connecté**. Il est donc crucial de fixer γ à un niveau qui ne freine pas abusivement l'établissement de connexions utiles.

Par ailleurs, la **dépendance à la taille** du voisinage pose également problème. Dans la formule $[-\gamma \sum_{k \neq j} \omega_{i,k}]$, la **somme** peut devenir très importante lorsque l'entité \mathcal{E}_i compte un grand nombre de voisins, annihilant de fait la mise à jour et rendant difficile l'émergence de nouvelles liaisons. Pour y remédier, on peut envisager de normaliser cette somme (par exemple, en la divisant par le nombre de voisins) ou d'adopter une approche "d'inhibition latérale stricte", qui retient uniquement la valeur maximale au lieu de la somme.

Enfin, il est fréquent de **combiner** l'inhibition compétitive avec un **mécanisme de saturation** : même si la synergie $S(i,j)$ incite un lien à s'intensifier, on l'empêche de dépasser un certain plafond. Cette mesure limite les phénomènes de "toute-puissance" où une entité tenterait d'activer tous ses liens simultanément, renforçant ainsi la sélectivité et la cohérence globale du réseau.

D. Conclusion

L'**inhibition compétitive** constitue un **levier** dans la mise à jour $\omega_{i,j}$ pour **forcer** un certain niveau de **compétition** entre les connexions sortant d'une même entité \mathcal{E}_i . Elle a pour effet de :

- 195. **Éviter** que de multiples liens forts cohabitent sans restriction,
- 196. **Promouvoir** des liens plus structurés ou plus "discriminants" (un "choix" se fait),
- 197. **Mieux** calquer certains phénomènes biologiques ou cognitifs (inhibition latérale), ou tout simplement améliorer la **lisibilité** des clusters en créant des sélections claires.

En pratique, elle se formule souvent comme un **terme négatif** proportionnel à la somme (ou au max) des liens $\omega_{i,k}(t)$ pour $k \neq j$. C'est un **complément** utile aux règles additive/multiplicative, afin de **rendre** la dynamique plus **sélective** et **moins** susceptible de disperser la force sur trop de connexions.

4.2.2.3. Saturation : clipping $\omega_{i,j}$ pour éviter la croissance infinie

Même dans un **DSL** (Deep Synergy Learning) recourant à une mise à jour plutôt "additive" (4.2.1) ou intégrant de l'inhibition (4.2.2.2), il peut arriver que les valeurs $\omega_{i,j}$ aient tendance à **s'emballer**. Par exemple, dans la variante **multiplicative** (4.2.2.1), un lien déjà élevé peut croître encore plus

vite, menant à un risque d'**explosion** numérique ou de “monopolisation” du réseau par quelques connexions hypertrophiées. Pour éviter cela, on introduit souvent un **mécanisme de saturation**, parfois appelé **clipping**, limitant la valeur maximale (ou minimale) que peut prendre $\omega_{i,j}$.

Le **principe du clipping** consiste à imposer, à l’issue de chaque itération de mise à jour, une **borne** maximale (et parfois minimale) à la pondération $\omega_{i,j}(t+1)$. Concrètement, cela se traduit par une opération de type :

$$\omega_{i,j}(t+1) \leftarrow \max\{0, \min\{\omega_{i,j}(t+1), \omega_{\max}\}\},$$

assurant ainsi que $\omega_{i,j}(t+1)$ ne dépasse pas la valeur ω_{\max} . Selon les besoins, on peut également l’empêcher de devenir négative, notamment si l’on considère que les liens ne doivent prendre que des valeurs positives.

Dans la pratique, ce **clipping** s’opère **après** le calcul brut de la mise à jour (qu’elle soit additive, multiplicative ou incluant un mécanisme d’inhibition). On recadre la pondération finale uniquement si elle outrepasse l’intervalle autorisé : si, par exemple, $\omega_{i,j}(t+1)$ atteignait 50 alors que $\omega_{\max} = 10$, on la ramènerait à la valeur 10.

La **détermination** de ω_{\max} varie selon la nature du problème. Certaines configurations imposent, par exemple, qu’un lien ne dépasse jamais 1 pour des raisons de normalisation. Dans d’autres contextes, on fixe $\omega_{\max} = 10$ en considérant qu’au-delà, la distinction entre pondérations serait négligeable. Par ailleurs, on peut instaurer un seuil plancher (comme $\omega_{\min} = 0$) si l’on souhaite empêcher les liens de devenir négatifs ou si un sens physique (ou logique) exige que les pondérations restent non négatives.

Les **avantages de la saturation** dans un mécanisme de mise à jour des pondérations se manifestent à plusieurs niveaux.

En premier lieu, la présence d’un **plafond** évite l’**explosion** de certaines connexions : dans les règles multiplicatives (4.2.2.1), ou même en l’absence d’un terme de décroissance convenable, quelques liens pourraient croître indéfiniment si la partie positive (par exemple $\eta S(i,j)$) l’emportait toujours sur la partie négative. Grâce à un **clipping** imposant ω_{\max} , on bride ces valeurs pour éviter qu’une minorité de liaisons n’atteigne des niveaux disproportionnés.

Ensuite, ce mécanisme contribue à **stabiliser la dynamique** globale. En empêchant $\omega_{i,j}$ de dépasser ω_{\max} , on restreint la **disparité** entre les connexions, ce qui peut faciliter l’**analyse** du réseau ou accélérer la **convergence**, puisque l’espace de recherche (celui des pondérations) demeure borné.

Enfin, la saturation favorise un **contrôle de la lisibilité** des pondérations. Lorsqu’on désire interpréter $\omega_{i,j}$ comme un simple “score” dans un intervalle [0,1], le clipping assure que ce score restera dans le **domaine** de lecture intuitif ; on n’observera ainsi pas de valeur hors échelle (telle qu’un score à 29,5), dépourvue de sens sémantique pour l’utilisateur ou l’analyste.

Les **inconvénients potentiels** de la saturation (ou points d’attention) doivent être soigneusement considérés au moment d’implémenter le clipping. En premier lieu, on peut évoquer l’**effet brutal** : dès que $\omega_{i,j}$ atteint ω_{\max} , toute mise à jour positive devient sans effet. Cela risque de **biaiser** la dynamique si, par exemple, un lien devrait continuer à croître pour signaler sa domination. Le

réseau ne percevra plus la différence entre des liens saturés et ceux proches de la borne, ce qui peut diminuer le contraste souhaité.

Un autre point concerne la **discontinuité** dans la mise à jour. Le clipping crée une **coupure** non lisse : une fois la limite atteinte, la valeur reste figée. Sur le plan mathématique, ces discontinuités peuvent compliquer certaines analyses de convergence.

Le **choix** de ω_{\max} est également crucial. Une borne trop **basse** étouffe les liens qui pourraient être significativement plus élevés que d'autres, affectant la capacité du réseau à différencier des connexions "très fortes" de connexions "modérées". À l'inverse, si ω_{\max} est fixé à un niveau **très élevé**, on ne résout pas réellement le problème de l'explosion, et on n'apporte qu'un contrôle minimal.

Exemple d'Intégration dans la Mise à Jour

Le clipping peut s'insérer de différentes manières dans le processus de mise à jour :

198. Dans la **version additive** avec clipping, on calcule d'abord :

$$1. \omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j)\tau \omega_{i,j}(t)],$$

puis on applique :

$$\omega_{i,j}(t+1) \leftarrow \min\{\max\{0, \omega_{i,j}(t+1)\}, \omega_{\max}\}.$$

On garantit ainsi que $\omega_{i,j}(t+1)$ demeure dans l'intervalle $[0, \omega_{\max}]$.

199. Dans la **version multiplicative**, une fois le calcul effectué (voir 4.2.2.1), on impose simplement $\omega_{i,j}(t+1) \leq \omega_{\max}$.

200. Lorsqu'on **combine** cette approche avec une **inhibition compétitive**, l'inhibition agit en amont (pour réduire la mise à jour) et le **clamping** final vient s'assurer qu'un lien ne dépasse pas le plafond fixé. De cette manière, même si la synergie est très élevée, on limite la croissance excessive d'un lien unique, préservant l'équilibre général du réseau.

Conclusion

Le **clipping** ou la **saturation** des pondérations $\omega_{i,j}$ est une mesure **pratique** visant à **contrôler** :

- La croissance potentielle **infinie** (notamment dans des règles multiplicatives),
- Les **excès** de la dynamique (liaisons trop fortes qui éliminent toute nuance),
- L'**interprétation** (on se limite à un range $[0, \omega_{\max}]$ pour lire facilement le "poids" d'un lien).

Cette technique, assez courante dans les algorithmes neuronaux (on pense au "gradient clipping" dans le deep learning), s'avère tout aussi utile dans un **DSL** pour **stabiliser** ou **normaliser** la dynamique d'auto-organisation. Elle peut s'intégrer à n'importe quelle version de la mise à jour

(additive, multiplicative, inhibition) en garantissant qu'on n'outrepasse pas des bornes considérées comme logiquement ou numériquement souhaitables.

4.2.3. Cas Multi-Entités Hétérogènes

Les sections précédentes (4.2.1 et 4.2.2) ont présenté diverses **règles** de mise à jour (additives, multiplicatives, avec ou sans inhibition, saturation...) pour les pondérations $\omega_{i,j}$. Cependant, elles partaient souvent de l'hypothèse qu'il existait un seul **type** d'entité ou une forme unique de $S(i,j)$. Or, dans un **DSL** (Deep Synergy Learning) plus complet (chap. 3), on peut croiser :

- Des **entités sub-symboliques** (embeddings, vecteurs),
- Des **entités symboliques** (règles logiques, blocs ontologiques),
- Des **entités hybrides** (mix sub + sym),

et chacune d'elles peut interagir sous forme de différents **types de liens**. Cette diversité amène à considérer un **multi-réseau** ou des **sous-réseaux** partiellement interconnectés, avec des règles de mise à jour possiblement **différencier**ées selon la nature des entités et des liaisons.

4.2.3.1. Sous-réseaux sub-symboliques / symboliques (cf. chapitre 3)

Dans le Deep Synergy Learning (DSL), il peut exister des **sous-réseaux sub-symboliques** (où les entités \mathcal{E}_i sont décrites par des embeddings \mathbf{x}_i , et où la synergie $S_{\text{sub}}(i,j)$ se base sur des critères vectoriels) et des **sous-réseaux symboliques** (où d'autres entités \mathcal{E}_p sont définies par des **règles** ou un **bloc** ontologique, et où $S_{\text{sym}}(p,q)$ repose sur la **compatibilité** logique). Ce modèle mixte, déjà évoqué au chapitre 3, illustre la capacité du DSL à intégrer, dans un **même cadre**, des entités numériques et des entités logiques.

Un **exemple** apparaît en **robotique** (chap. 3.6.3.3), avec des entités représentant des **perceptions** (embeddings sensoriels) et d'autres correspondant à des **actions** décrites symboliquement (précisant préconditions et effets). De même, pour un **agent conversationnel** (chap. 3.6.3.2), certaines entités sont des **segments textuels** (embedding BERT), tandis que d'autres englobent des **règles** de cohérence de dialogue.

Sur le plan **structurel**, on se retrouve souvent avec plusieurs "couches" d'entités :

- La couche "sub" manipule \mathbf{x}_i et inclut des liens sub–sub,
- La couche "sym" gère les blocs logiques et inclut des liens sym–sym,
- Et il existe parfois des **passerelles** (liens sub–sym) reliant les deux couches.

L'**enjeu** majeur réside dans la capacité de chaque type d'entité à se **mettre à jour** (par exemple, une représentation sub-symbolique qui évolue, ou des règles symboliques qui se modifient), et dans la prise en compte du fait que la **synergie** varie selon la nature des entités connectées. Par conséquent, les mécanismes de mise à jour (additifs, multiplicatifs, saturation, etc.) décrits en 4.2.1

et 4.2.2 doivent être adaptés à la catégorie de la liaison (sub–sub, sym–sym ou sub–sym) pour demeurer cohérents avec les spécificités de chaque sous-réseau.

4.2.3.2. Mise à jour distincte selon le type de lien (ex. sub–sub, sym–sym, sym–sub)

Dans le cadre d'un **Deep Synergy Learning (DSL)** hétérogène, les entités peuvent appartenir à des natures différentes, qu'elles soient **sub-symboliques** (décrites par des **embeddings** tels que $\mathbf{x}_i \in \mathbb{R}^d$), **symboliques** (décrites par un ensemble de **règles**, axiomes ou ontologies, noté R_i) ou **hybrides** (combinant à la fois des représentations vectorielles et des blocs logiques). Cette diversité des représentations se traduit par des **types de liens** distincts dans le réseau : on distingue ainsi les liens **sub–sub** (entre deux entités sub-symboliques), les liens **sym–sym** (entre deux entités symboliques) et les liens **sym–sub** (passerelles entre le domaine symbolique et le domaine sub-symbolique). Chaque type de lien requiert une **mise à jour** adaptée, car la **fonction de synergie** $S(i,j)$ utilisée pour moduler l'évolution de la pondération $\omega_{i,j}$ diffère en fonction des caractéristiques de chaque modalité. Nous détaillons ci-après les principes sous-jacents à la mise à jour pour chacun de ces types de liens.

A. Liens sub–sub

Lorsqu'un lien relie deux entités \mathcal{E}_i et \mathcal{E}_j qui sont toutes deux représentées par des **embeddings** \mathbf{x}_i et \mathbf{x}_j dans \mathbb{R}^d , la **synergie sub-symbolique** est généralement évaluée à l'aide de mesures vectorielles classiques. Par exemple, la **similarité cosinus** se définit par

$$S_{\text{sub}}(i,j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

ce qui quantifie le degré de rapprochement entre les directions de ces vecteurs. Alternativement, on peut utiliser une mesure basée sur une fonction noyau, comme un noyau gaussien exprimé par

$$S_{\text{sub}}(i,j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|^2),$$

où $\alpha > 0$ règle la sensibilité du score. La **mise à jour** standard de la pondération dans ce cas se fait selon la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{sub}}(i,j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage** et τ le **coefficient de décroissance** qui agit comme un terme de régularisation. Cette formulation, d'une complexité de l'ordre de $O(d)$ pour l'évaluation du score, permet de renforcer progressivement les liens entre des entités dont les représentations sont numériquement proches, par exemple deux images similaires ou deux segments textuels ayant des significations proches. Des variantes multiplicatives peuvent être envisagées si l'on souhaite accentuer l'effet exponentiel sur la croissance des liens, mais le principe fondamental demeure celui de la convergence vers un équilibre où, en situation stationnaire, $\omega_{i,j}^* \approx \frac{S_{\text{sub}}(i,j)}{\tau}$.

B. Liens sym–sym

Dans le cas des liens **sym–sym**, les entités \mathcal{E}_i et \mathcal{E}_j sont caractérisées par des **blocs de règles** ou des structures logiques, notées R_i et R_j . La fonction de synergie symbolique $S_{\text{sym}}(i,j)$ évalue la

compatibilité entre ces ensembles de règles, ce qui peut être formulé de manière binaire ou graduée. Par exemple, on peut définir

$$S_{\text{sym}}(R_i, R_j) = \begin{cases} 1, & \text{si } R_i \text{ et } R_j \text{ sont entièrement compatibles,} \\ 0, & \text{en cas de contradiction totale,} \\ s_{ij}, & \text{si la compatibilité est partielle, avec } s_{ij} \in (0,1). \end{cases}$$

La mise à jour des pondérations pour les liens sym–sym suit un schéma similaire à celui des liens sub–sub, à savoir

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{sym}}(i,j) - \tau \omega_{i,j}(t)],$$

ce qui permet de renforcer les connexions entre entités dont les blocs de règles se valident mutuellement. Toutefois, il faut noter que la comparaison symbolique peut s'avérer plus coûteuse en termes de calcul, surtout si l'inférence sur des systèmes logiques complexes est nécessaire. Des mécanismes d'**inhibition** ou des seuils peuvent être introduits pour limiter l'impact des incohérences et éviter une prolifération excessive des liens lorsque la logique est trop rigide.

C. Liens sym–sub

Les **liens sym–sub** constituent des passerelles entre les représentations sub-symboliques et symboliques. Dans ce cas, une entité \mathcal{E}_i peut être représentée par un **embedding** \mathbf{x}_i tandis qu'une autre entité \mathcal{E}_j est décrite par un bloc de règles R_j , ou vice versa. Pour mesurer la synergie dans un tel cas, il est courant de définir une fonction hybride qui combine la **similarité vectorielle** et la **compatibilité logique**. Une telle fonction s'exprime par

$$S_{\text{hybrid}}(i,j) = \alpha S_{\text{sub}}(i,j) + (1 - \alpha) S_{\text{sym}}(i,j),$$

où le paramètre $\alpha \in [0,1]$ ajuste l'importance relative des deux composantes. La mise à jour des pondérations pour ces liens hybrides est alors donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hybrid}}(i,j) - \tau \omega_{i,j}(t)].$$

Cette approche permet de **fusionner** les informations provenant du monde numérique (embeddings) et du monde logique (règles), ce qui est particulièrement utile dans des applications où il est nécessaire de combiner la richesse des données sub-symboliques avec une **explicabilité** issue du raisonnement formel. Toutefois, l'évaluation simultanée des deux composantes peut augmenter la **complexité** de l'algorithme, et un réglage fin du paramètre α est indispensable pour obtenir le compromis optimal entre flexibilité et précision.

D. Conclusion sur la mise à jour multi-type

La mise à jour distincte selon le type de lien dans un DSL hétérogène illustre la capacité du système à adapter ses mécanismes d'auto-organisation aux caractéristiques spécifiques des représentations des entités. Pour les liens **sub–sub**, la mise à jour repose sur des calculs vectoriels efficaces et rapides, permettant de renforcer la similarité entre entités par des opérations de complexité $O(d)$. Pour les liens **sym–sym**, la dynamique repose sur une évaluation logique qui, bien que potentiellement plus coûteuse, offre une explicabilité et une cohérence formelle appréciables. Enfin, pour les liens **sym–sub**, une mise à jour hybride combine les avantages des deux approches

en ajustant les pondérations à l'aide d'un paramètre α qui permet de peser différemment la contribution de la similarité sub-symbolique et de la compatibilité symbolique.

Ainsi, le DSL hétérogène gagne en **souplesse** et en **capacité d'adaptation** en intégrant des règles de mise à jour différencierées pour chaque type de lien, ce qui permet de préserver une **dynamique** globale stable et cohérente malgré la diversité des représentations. Le contrôle de la **cohérence** du SCN repose sur une gestion fine de la **pondération** et de la **complexité** des interactions, garantissant que chaque type de liaison contribue de manière optimale à l'auto-organisation du réseau.

4.2.3.3. Exemples de pseudo-code unifiant ces règles

Dans un **Deep Synergy Learning (DSL)** hétérogène, la gestion des liens entre entités repose sur la mise à jour de pondérations $\omega_{i,j}$ qui varient selon la **nature** des entités en interaction. En effet, les entités peuvent être purement **sub-symboliques** (représentées par des **embeddings** $\mathbf{x}_i \in \mathbb{R}^d$), purement **symboliques** (définies par un ensemble de **règles** ou d'**axiomes** noté R_i), ou encore **hybrides** (combinant à la fois un embedding et un bloc logique). Pour chaque type de lien – qu'il s'agisse de liens sub–sub, sym–sym ou sym–sub – la **fonction de synergie** $S(i, j)$ et la stratégie de mise à jour doivent être adaptées. La présente section présente un **pseudo-code** uniifié qui illustre comment, dans une même boucle d'itération, ces règles de mise à jour distinctes peuvent être intégrées de manière homogène dans l'algorithme de mise à jour des pondérations du **Synergistic Connection Network (SCN)**.

A. Présentation générale du cadre

Soit un ensemble d'entités $\{\mathcal{E}_1, \mathcal{E}_2, \dots, \mathcal{E}_n\}$ dont chacune est associée à une représentation qui peut être sub-symbolique, symbolique ou hybride. Nous disposons d'une **matrice** $\{\omega_{i,j}\}$ représentant les pondérations actuelles entre chaque paire d'entités. Le système repose sur une fonction générique, que nous appellerons $\text{ComputeSynergy}(i, j, \text{link_type})$, qui, en fonction du **type de lien** entre \mathcal{E}_i et \mathcal{E}_j (déterminé par une fonction $\text{determine_link_type}(E[i], E[j])$), renvoie le score de synergie approprié. Ainsi, dans le cas d'un lien sub–sub, ComputeSynergy renverra un score $S_{\text{sub}}(i, j)$ calculé à partir des embeddings, typiquement par une mesure comme la similarité cosinus ou un noyau gaussien, tandis que pour un lien sym–sym, la fonction renverra un score $S_{\text{sym}}(i, j)$ évalué sur la compatibilité des règles, et dans le cas d'un lien sym–sub, une combinaison des deux scores sera utilisée, par exemple via une formule hybride.

La règle de mise à jour additive, commune à tous les types, se présente sous la forme

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage** et τ le **coefficent de décroissance**. Des variantes multiplicatives peuvent également être envisagées, mais dans ce pseudo-code nous nous concentrerons sur la règle additive. Afin de contrôler la croissance excessive des pondérations, des mécanismes de **clipping** (pour borner les valeurs) et d'**inhibition** (pour tenir compte de la compétition entre liens) sont intégrés dans la mise à jour.

B. Pseudo-code unifié

Le pseudo-code suivant, de style Python, illustre comment on peut unifier les mises à jour des différents types de liens dans une boucle d'itération :

```

# Paramètres globaux
eta_sub = 0.01 # Taux d'apprentissage pour liens sub-sub
eta_sym = 0.02 # Taux pour liens sym-sym
eta_hyb = 0.015 # Taux pour liens sym-sub
tau = 0.1 # Coefficient de décroissance
gamma = 0.05 # Coefficient pour inhibition (optionnel)
omega_max = 5.0 # Valeur maximale pour le clipping
USE_INHIBITION = True # Option pour activer le mécanisme d'inhibition

# Boucle principale sur toutes les entités
for i in range(n): # Pour chaque entité i
    for j in range(n): # Pour chaque entité j
        if i == j:
            continue # Pas de mise à jour pour i=j

        # Déterminer le type de lien entre E[i] et E[j]
        link_type = determine_link_type(E[i], E[j])
        # Exemples de valeurs possibles: "sub-sub", "sym-sym", "sym-sub"

        # Calcul de la synergie S(i,j) en fonction du type de lien
        S_ij = ComputeSynergy(E[i], E[j], link_type)
        # Par exemple, pour un lien sub-sub, ComputeSynergy renvoie:
        # S_sub = (E[i].x dot E[j].x) / (||E[i].x|| * ||E[j].x||)
        # Pour un lien sym-sym, il renvoie un score basé sur la compatibilité logique
        # Pour un lien sym-sub, il combine les deux scores via S_hybrid = alpha*S_sub + (1-alpha)*S_sym

        # Calcul optionnel d'un terme d'inhibition, basé sur la somme des poids sortants de i
inhib_term = 0.0
if USE_INHIBITION:
    sum_weights = 0.0
    for k in range(n):
        if k != j:
            sum_weights += w[i][k]
    inhib_term = gamma * sum_weights

# Sélection du taux d'apprentissage spécifique selon le type de lien
if link_type == "sub-sub":
    learning_rate = eta_sub
elif link_type == "sym-sym":
    learning_rate = eta_sym
else: # pour "sym-sub"

```

```

learning_rate = eta_hyb

# Mise à jour additive de la pondération
# Formule : w_new = w(i,j) + learning_rate * (S_ij - tau*w(i,j)) - inhib_term
w_new = w[i][j] + learning_rate * (S_ij - tau * w[i][j]) - inhib_term

# Application d'un clipping pour éviter des valeurs extrêmes
if w_new < 0:
    w_new = 0
if w_new > omega_max:
    w_new = omega_max

# Mise à jour de la matrice de pondérations
w[i][j] = w_new

```

Explications détaillées :

Dans ce pseudo-code, la variable **E** représente la liste des entités du DSL. Chaque entité peut être accompagnée d'un **embedding** (pour les entités sub-symboliques), d'un **bloc de règles** (pour les entités symboliques) ou d'une combinaison des deux (pour les entités hybrides). La fonction **determine_link_type(E[i], E[j])** permet de distinguer le type de lien à établir entre deux entités en se basant sur leur nature respective. Par exemple, si deux entités possèdent toutes deux un embedding, le lien est qualifié de **sub–sub** ; si elles sont toutes deux décrites par des règles, le lien est **sym–sym** ; et si l'une est sub-symbolique et l'autre symbolique, le lien est dit **sym–sub**.

La fonction **ComputeSynergy(E[i], E[j], link_type)** renvoie le score de synergie approprié en fonction du type de lien. Pour les liens sub–sub, ce score est généralement calculé à partir de mesures classiques telles que la similarité cosinus ou une fonction de noyau gaussien, tandis que pour les liens sym–sym, il s'agit d'un score reflétant la compatibilité ou l'absence de contradictions entre les blocs logiques R_i et R_j . Pour les liens sym–sub, on combine les deux approches via une formule hybride, par exemple

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j).$$

Le **terme d'inhibition** est calculé (si activé) en sommant les pondérations sortantes d'une entité pour modérer la croissance excessive des liens et éviter une saturation du réseau. Chaque type de lien peut bénéficier d'un **taux d'apprentissage** distinct, ici symbolisé par η_{sub} , η_{sym} et η_{hyb} pour les liens sub–sub, sym–sym et sym–sub, respectivement. La mise à jour s'effectue de manière additive, comme le montre la formule utilisée, puis le **clipping** est appliqué pour garantir que les pondérations restent dans un intervalle prédéfini, par exemple $[0, \omega_{\max}]$.

Ce pseudo-code s'exécute de manière itérative sur l'ensemble des entités, et peut être répété pour de nombreuses itérations jusqu'à convergence ou pendant une durée spécifiée. Il offre une illustration concrète de la façon dont un **DSL** peut gérer simultanément des règles de mise à jour différencierées pour plusieurs types de liens, en assurant une intégration harmonieuse des informations sub-symboliques et symboliques.

Conclusion

Le présent pseudo-code constitue un exemple unifié qui intègre les différentes **règles de mise à jour** selon le type de lien dans un DSL hétérogène. En adaptant dynamiquement le taux d'apprentissage et en appliquant des mécanismes complémentaires tels que l'inhibition et le clipping, le système parvient à mettre à jour les pondérations $\omega_{i,j}$ de manière cohérente, tout en tenant compte des spécificités des interactions sub–sub, sym–sym et sym–sub. Ce schéma permet ainsi d'unifier la mise à jour des liens dans un SCN, assurant la **robustesse** et la **flexibilité** de l'auto-organisation même dans un environnement où coexistent plusieurs types de représentations.

4.3. Émergence de Clusters et Attracteurs

Dans les sections précédentes (4.2), nous avons décrit les différentes **règles** de mise à jour permettant aux pondérations $\omega_{i,j}$ d'évoluer sous l'effet de la synergie $S(i,j)$, de termes de décroissance, d'inhibition, ou de saturation. L'un des **Résultats majeurs** de cette dynamique d'auto-organisation est l'**émergence** spontanée de **clusters** — c'est-à-dire de sous-ensembles d'entités qui se lient fortement entre elles, tout en se distançant des autres.

En creusant ce phénomène, on découvre également la notion d'**attracteurs** : selon la configuration initiale et la structure de la synergie, le **SCN** (Synergistic Connection Network) peut aboutir à différents **points fixes** (clusters stabilisés), ou manifester des comportements plus riches (multi-stabilité, oscillations). Cette section (4.3) analysera donc :

201. Comment les **clusters** se forment (4.3.1),
202. Les concepts d'**attracteurs multiples** et de **bascules** (4.3.2),
203. Les scénarios de **stabilisation** ou de **dynamisme continu** (4.3.3), montrant que le réseau peut se figer ou rester en mutation constante, selon le flux de synergie et l'ajout de nouvelles données.

4.3.1. Formation Spontanée de Groupes

Lorsque l'on met en place les **mécanismes** décrits en 4.2 (mise à jour additive ou multiplicative, inhibition, etc.), les **entités** $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ ne tardent pas à voir leurs liens $\omega_{i,j}$ s'**organiser** : certains liens deviennent **forts**, d'autres s'**étiolent**, et des **groupes** se forment. C'est précisément ce qui confère au **DSL** (Deep Synergy Learning) sa **capacité** d'auto-organisation : en "laissant faire" la dynamique, on aboutit à des **clusters** d'entités hautement interconnectées, tout en limitant les connexions hors-clusters.

4.3.1.1. Notion de Clusters : pondérations fortes à l'intérieur d'un sous-ensemble \mathcal{C} , faibles à l'extérieur

Lorsqu'on examine la structure d'un **Synergistic Connection Network (SCN)**, il est courant d'observer l'**émergence** de groupes d'entités, appelés **clusters** ou **communautés**, à l'intérieur desquels les **pondérations** $\omega_{i,j}$ sont nettement plus **élèves** que celles reliant ces entités au **reste** du réseau. Autrement dit, un **sous-ensemble** $\mathcal{C} \subseteq \{1, \dots, n\}$ va exhiber des liens $\omega_{i,j}$ particulièrement **forts** pour $(i, j \in \mathcal{C})$, tandis que les connexions $\omega_{i,k}$ vers un $k \notin \mathcal{C}$ resteront **relativement faibles**.

Pour illustrer cette idée, on peut considérer un exemple simple : si $\mathcal{C} = \{2, 5, 7\}$ forme un cluster, on **attend** à l'issue de la dynamique DSL que $\omega_{2,5}$, $\omega_{5,7}$ et $\omega_{2,7}$ deviennent **hautes**, tandis que les liaisons $\omega_{2,k}$, $\omega_{5,k}$, $\omega_{7,k}$ pour $k \notin \{2, 5, 7\}$ demeurent **faibles** ou se réduisent au fil des itérations.

La **raison mathématique** de cette formation de clusters tient à la **synergie** : lorsque, pour un ensemble donné d'entités, la fonction $S(i,j)$ est (en moyenne) plus élevée entre elles qu'avec l'extérieur, la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

tend à **renforcer** progressivement $\omega_{i,j}$ pour $(i,j \in \mathcal{C})$. Symétriquement, dès lors que la synergie entre $(i \in \mathcal{C})$ et $(k \notin \mathcal{C})$ est plus faible, la pondération $\omega_{i,k}(t)$ s'**amenuise** ou reste basse, et ne concurrence pas la liaison intra-groupe.

Un aspect central de cette dynamique est l'**absence de supervision** : le système n'exige aucun label ou contrainte externe pour “dire” quelles entités devraient s'assembler. Seuls les **patterns** de synergie, qu'ils soient issus de la similarité (embeddings, co-occurrences) ou d'une logique symbolique (chap. 3.5), suffisent à produire une **dépendance mutuelle** entre entités. Si la synergie révèle des affinités nettement plus fortes au sein du sous-ensemble \mathcal{C} , la **dynamique** DSL consolide précisément ces liens.

Sur le plan **théorique**, ce processus rejoint l'idée du **clustering** en apprentissage non supervisé, à ceci près que la dynamique DSL reste **entièvement itérative** et **coopérative**. Au lieu de définir un algorithme statique (k-means ou agglomératif) qui opère en une passe sur les données, on laisse la **matrice** $\{\omega_{i,j}\}$ évoluer au fil du temps et, de manière **auto-organisée**, “dessiner” des **flots** fortement interconnectés.

Une fois l'étape de **convergence** atteinte, la matrice ω révèle explicitement l'existence de **clusters** : on peut repérer ces groupes en appliquant un **seuil** sur $\omega_{i,j}$ (les plus fortes valeurs identifient les sous-ensembles fortement liés), ou en recourant à des méthodes de **community detection** (louvain, modularité, etc.) sur le graphe formé par ω .

Sur le plan **macro**, cette **formation de clusters** marque l'un des aboutissements majeurs de la **synergie** : le SCN, parti d'entités possiblement dispersées et de pondérations initiales faibles ou aléatoires, se **structure** en regroupements cohérents, reflétant la **cohésion** intrinsèque que la fonction S a détectée dans les données. Cette propriété d'**auto-organisation** constitue l'un des piliers fondamentaux du DSL, montrant comment, à partir de **règles** locales et d'ajustements distribués, le réseau peut dégager une **partition** (partielle ou totale) du jeu d'entités en **communautés** pertinentes, sans qu'aucun paramétrage ou nombre de clusters ne soit imposé a priori.

4.3.1.2. Processus Auto-Organisé : sans supervision, certains liens se renforcent massivement (haute synergie), d'autres décroissent

Le **phénomène** de formation de **clusters** décrit en section 4.3.1.1 ne repose pas sur l'utilisation d'un algorithme de **clustering** classique ni sur la présence de **labels** ou de **catégories** prédefinies. Au contraire, il résulte d'un **processus auto-organisé** intrinsèque aux pondérations $\omega_{i,j}$, lesquelles évoluent en fonction de la **synergie** $S(i,j)$ et des règles de mise à jour (décris au paragraphe 4.2). C'est précisément ce caractère **non supervisé** et **dynamique** qui caractérise le **DSL** (Deep Synergy Learning) et qui en fait sa spécificité.

Caractère non supervisé.

Dans une configuration d'**apprentissage supervisé**, on dispose généralement de classes ou de cibles explicites pour guider la création de groupes. Ici, la démarche ne s'appuie sur **aucune** annotation : seule la **synergie** $S(i,j)$ — qu'elle provienne de similarités sub-symboliques, de compatibilités logiques ou d'autres sources (cf. chapitre 3) — oriente la **croissance** ou la **décroissance** des pondérations $\omega_{i,j}$. L'évolution du réseau reste ainsi *auto-organisée*, car elle ne fait appel à aucun signal externe imposant la façon de répartir ou de regrouper les entités.

L'**analogie** avec certains réseaux neuronaux biologiques est souvent mise en avant : dans le cerveau, les synapses voient leur force modifiée selon l'activité neuronale partagée (une certaine forme de “co-activation” ou de “synergie”), en l'absence d'un “superviseur” qui affecterait manuellement des étiquettes à chaque neurone. Le **DSL** transpose ces principes à des entités abstraites (tels que des textes, des images, des règles logiques, etc.), dont les liens se renforcent ou s'affaiblissent selon leur synergie mutuelle.

Renforcement massif de certains liens, décroissance des autres.

Imaginons qu'à l'initialisation, certaines paires (i,j) présentent une **synergie** $S(i,j)$ relativement élevée, tandis que d'autres paires (i,k) sont moins corrélées. La dynamique DSL (qu'elle soit **additive** ou **multiplicative**, avec ou sans mécanismes d'**inhibition**, de **saturation**, etc.) va **favoriser** la montée en puissance de $\omega_{i,j}$ dès lors que $S(i,j)$ se révèle important. Au fil des itérations, la pondération $\omega_{i,j}$ tend à **se stabiliser** (voire à continuer d'augmenter si les régulations le permettent), constituant ainsi un **cœur** de liaisons robustes.

De manière symétrique, les couples (i,k) pour lesquels la synergie $S(i,k)$ se montre plus faible subissent l'effet de **décroissance** lié à $-\tau \omega_{i,k}(t)$ (et éventuellement à l'inhibition compétitive). Cela se traduit par une **baisse** progressive, voire une extinction, de $\omega_{i,k}$. Au terme de ce processus, certains liens s'en trouvent **renforcés** de façon marquée, et la majorité finit par s'**affaiblir** notamment. Cette situation aboutit à des “**îlots de forte connectivité**”, constituant les futurs **clusters**.

Dans les versions dites **multiplicatives** de la mise à jour, un léger avantage initial d'une pondération peut conduire à une **croissance exponentielle**, accentuant encore la séparation entre les connexions “vainqueurs” (dotées d'une synergie forte) et les connexions “perdantes” (qui demeurent modestes ou décroissent). Même en mode **additif**, un contraste initial de synergie suffit à créer, à l'issue de la dynamique, un écart stable : l'une des liaisons converge vers $S(i,j)/\tau$, l'autre vers $S(i,k)/\tau$. Si $S(i,j) > S(i,k)$, la différence se creuse inévitablement.

Émergence de clusters.

À mesure que ces liaisons puissantes se démarquent et que la majorité des autres s'affaiblit, on observe la formation naturelle de **sous-ensembles** d'entités — c'est-à-dire de **clusters** — unis par des pondérations $\omega_{i,j}$ élevées. Au plan géométrique ou sémantique, cette différence renvoie à la **répartition** des synergies S dans l'espace des données : là où $S(i,j)$ s'avère plus grand, la pondération $\omega_{i,j}$ finit par créer des **composantes** ou **communautés** (au sein du SCN). La **matrice** $\{\omega_{i,j}\}$ se “sculpte” littéralement sous l'action de la dynamique DSL, laissant apparaître des **îlots de forte connectivité**.

En pratique, on peut suivre, au fil des itérations, la progression vers des “clusters” plus ou moins stables. Durant les premiers pas, plusieurs connexions peuvent simultanément croître, puis l'**inhibition** ou la **saturation** vont faire émerger des groupes plus définis, car un lien fort d’une entité vers un ensemble donné empêche la prolifération de liens simultanés vers d’autres ensembles. Selon les conditions initiales (valeurs de η , τ , coefficient d’inhibition γ , etc.) et la distribution des synergies, le réseau finit fréquemment par converger vers un **état stable** (ou un quasi-état stable), découvant implicitement l’ensemble des entités en **sous-groupes**. Cette séparation se produit sans affectation de labels ni contrainte externe, illustrant la pleine **auto-organisation** : la seule fonction S , intégrée à la mise à jour, produit l'**émergence** de clusters pertinents.

Conclusion : la dynamique auto-organisée comme moteur de clusterisation.

Les pondérations $\omega_{i,j}$, guidées par la **synergie** $S(i,j)$ et la **règle** de mise à jour locale, aboutissent à la consolidation d’un ensemble fini de **communautés** d’entités, sans imposer de supervision. Cette propriété joue un rôle clé dans la “découverte” de groupes cohérents : le réseau se **structure** de l’intérieur, révélant en filigrane la **topologie** ou la **sémantique** dissimulée dans la mesure de synergie. Cet **auto-appariement** de connexions “fortes” fait émerger naturellement une **partition** globale du SCN, inaugurant l’étude de la **multi-stabilité** et des **attracteurs** (section 4.3.2) qui analysent les états finaux de cette dynamique.

4.3.2. Attracteurs Multiples et Bascules

Lorsque la dynamique d’auto-organisation (chap. 4.2) agit sur les pondérations $\omega_{i,j}$, on peut observer divers **phénomènes** de stabilité ou d’instabilité. En particulier, il arrive que le **SCN** (Synergistic Connection Network) dispose de **plusieurs** configurations stables (ou quasi-stables), vers lesquelles il peut converger selon l’initialisation des $\omega_{i,j}$ ou l’évolution du **flux de synergie** $S(i,j)$. C’est ce qu’on appelle la **multi-stabilité**, ou la co-existence de **plusieurs attracteurs**.

Par “**attracteurs multiples**”, on entend que le système (la matrice $\{\omega_{i,j}\}$) peut aboutir à différentes **configurations finales** (points fixes, cycles, régimes complexes) selon les conditions initiales ou la chronologie des mises à jour. On peut alors assister à des **basclements** entre attracteurs, si le **flux** $S(i,j)$ ou d’autres paramètres changent en cours de route.

4.3.2.1. Multi-Stabilité : un SCN peut atteindre différents points fixes selon l’initialisation ou le flux de synergie (2.3.2)

A. Rappel mathématique : points fixes et stabilité

Les réseaux **DSL** (Deep Synergy Learning), ou plus largement les systèmes de type **SCN** (Synergistic Connection Network), se décrivent par un ensemble de liens $\omega_{i,j}(t)$ qui évoluent au fil du temps. De manière formelle, on peut écrire cette évolution sous la forme d’une **application** F opérant sur la configuration globale $\omega(t)$. À chaque pas d’itération, la dynamique s’écrit :

$$\omega(t+1) = F(\omega(t)),$$

où $\omega(t)$ représente le vecteur (ou la matrice) regroupant l'ensemble $\{\omega_{i,j}(t)\}$. Un **point fixe** ω^* de ce système satisfait

$$\omega^* = \mathbf{F}(\omega^*).$$

Lorsqu'on a affaire à une fonction **F non linéaire**, il peut exister plusieurs **solutions** $\omega_1^*, \omega_2^*, \dots$ satisfaisant cette même équation, chaque solution correspondant à un **arrangement stable** des pondérations $\omega_{i,j}$. Dans la théorie des systèmes dynamiques, chacun de ces états constitue un **attracteur local**. Ainsi, selon l'**initialisation** $\omega(0)$ ou les perturbations ultérieures, la trajectoire $\omega(t)$ peut aboutir à l'un ou l'autre de ces points fixes, chacun possédant son **bassin d'attraction** propre.

Dans le cadre spécifique du **DSL**, la fonction **F** dépend de la **synergie** $S(i,j)$. Si la synergie est **constante** (aucune variation au fil du temps, aucun nouveau flux de données), la multi-stabilité découle essentiellement de la **non-linéarité** (par exemple l'inhibition compétitive ou le mécanisme multiplicatif). Au contraire, si la synergie $S(i,j)$ évolue (arrivée de nouvelles entités, changement de paramètres), le système peut, pendant sa trajectoire temporelle, **changer** d'attracteur si le “paysage” dynamique se modifie suffisamment.

B. Exemples conceptuels de multi-stabilité

Pour se faire une intuition, il est possible d'imaginer un **mini réseau** de seulement trois entités {1,2,3} et de se donner une mise à jour additive ou multiplicative. Le flux de synergies $S(1,2)$, $S(2,3)$, $S(1,3)$ peut être choisi de façon à ce que deux configurations concurrentes soient toutes deux “plausibles”. Dans la première, les entités {1,2} forment un cluster, excluant la troisième, tandis que dans la seconde, ce sont {2,3} qui se solidarisent. Si l'initialisation favorise l'émergence de {1,2} (par un bruit initial en ce sens), on converge naturellement vers cette partition, tandis qu'une initialisation différente fait converger vers {2,3}.

À plus grande échelle, si l'on dispose de nombreuses entités, il peut y avoir **plusieurs partitions** tout aussi cohérentes. La nature **non linéaire** de la règle (par exemple l'inhibition compétitive) accentue cette tendance : un groupe se forme et se stabilise d'un côté, un autre se stabilise différemment d'un autre côté, et il se peut que deux exécutions nominalement équivalentes aboutissent à des **arrangements** distincts. Cela renvoie à la notion de **paysage d'énergie** comportant plusieurs “puits” locaux, chacun correspondant à un **attracteur** stable. Ces principes sont largement étudiés dans la littérature des systèmes dynamiques et rappellent, dans une certaine mesure, les réseaux de Hopfield (où plusieurs états mémorisés coexistent comme attracteurs), ou encore les chaînes de Markov possédant différents états absorbants.

C. Rôle de l'initialisation et du flux de synergie

La **dépendance** vis-à-vis de l'initialisation $\omega(0)$ se comprend en analysant la trajectoire donnée par $\omega(t+1) = \mathbf{F}(\omega(t))$. Très souvent, on part d'une matrice $\omega_{i,j}(0)$ proche de zéro (ou perturbée par un faible bruit gaussien). Mais de petites variations dans cette phase initiale peuvent se répercuter fortement, en raison des **bifurcations** qu'introduit la non-linéarité.

Si la **synergie** $S(i,j)$ reste **constante** dans le temps, le système évolue vers l'un de ses points fixes disponibles, typiquement en une seule “phase” de convergence. En revanche, si S **varie** (arrivée de nouvelles données, recalcul d'embeddings, etc.), la fonction **F** se modifie, et un attracteur qui était

stable peut cesser de l'être. Le **réseau** peut alors **basculer** vers un autre attracteur plus conforme au nouveau contexte, parfois par un phénomène de "cascade" de réajustements.

Les **perturbations** extérieures (inhibition accrue, bruit stochastique, recuit simulé) peuvent également déstabiliser l'état courant et "pousser" le réseau vers une autre configuration stable.

D. Illustration mathématique : Jacobienne et stabilité locale

Pour analyser la stabilité locale d'un point fixe, on se réfère à la **matrice jacobienne** $\mathbf{DF}(\boldsymbol{\omega}^*)$. Étant donné la dynamique

$$\boldsymbol{\omega}(t+1) = \mathbf{F}(\boldsymbol{\omega}(t)),$$

la **condition** de stabilité veut que l'opérateur linéarisé $\mathbf{DF}(\boldsymbol{\omega}^*)$ (qui décrit la sensibilité à de petites variations autour de $\boldsymbol{\omega}^*$) ait un rayon spectral (ou une norme appropriée) < 1 . Cela garantit que la trajectoire reste dans l'orbite de $\boldsymbol{\omega}^*$ et ne diverge pas.

Lorsque $\mathbf{DF}(\boldsymbol{\omega}^*)$ permet à plusieurs $\boldsymbol{\omega}^*$ d'être stables, on parle de **co-existence** d'attracteurs. Chaque attracteur dispose de son **bassin**. Dans un cadre DSL de grande dimension, ce phénomène de co-existence peut amener à une grande diversité de partitions possibles, ou à des états stables différents selon l'histoire de la trajectoire.

E. Basculement entre attracteurs

Le fait de **changer** un paramètre η, γ ou τ , de faire **arriver** un nouveau flux de données, ou de recalculer **S** peut soudainement **rendre** un attracteur obsolète et permettre à la dynamique de **s'échapper** vers une configuration plus basse en énergie. Dans la théorie des systèmes dynamiques, on parle alors de **bifurcation**, ou de "transition de phase" lorsqu'on l'analyse avec une perspective inspirée de la physique statistique.

À grande échelle, on peut **voir** ce basculement se traduire par l'évolution de la matrice $\boldsymbol{\omega}$: un bloc d'entités qui formait un cluster disparaît subitement ou se scinde en deux, tandis qu'un autre cluster émerge ou fusionne avec un tiers. Tout cela résulte d'une multitude de **petits ajustements** $\Delta\omega_{i,j}$ qui, accumulés, aboutissent à un **changement** macroscopique.

F. Conclusion : la multi-stabilité comme source de diversité

La présence de **plusieurs** solutions stables dans un SCN souligne la **richesse** des dynamiques non linéaires : non seulement il existe la possibilité de se fixer sur un arrangement unique, mais il peut y avoir **divers** arrangements également stables (ou métastables). Cette **co-existence** explique comment deux exécutions, pourtant équivalentes en apparence, produisent des **partitions** légèrement ou radicalement différentes.

Un simple décalage dans l'initialisation, ou un changement progressif du flux **S**, peut faire **sauter** le réseau d'un attracteur à l'autre. Dans certains cas, ce phénomène se révèle bénéfique, car il autorise le système à **s'adapter** aux conditions (nouveaux types d'entités, re-paramétrage).

Sur le plan de l'**auto-organisation**, la multi-stabilité illustre la **diversité** des *solutions* ou *consensus* qu'un DSL peut obtenir, reflétant divers compromis entre les synergies présentes. Elle participe aussi de la **flexibilité** d'un SCN, qui ne s'enferme pas nécessairement dans une unique forme de partition, mais peut explorer plusieurs structures cohérentes.

Dans la prochaine étape (section 4.3.2.2), on considérera que la dynamique ne se limite pas à converger vers un point fixe : il se peut qu'elle donne naissance à des **cycles** ou **oscillations**, c'est-à-dire à des attracteurs d'une nature plus mobile que le simple état stationnaire.

4.3.2.2. Cycles ou “Oscillations” : si la fonction de mise à jour favorise des rétroactions positives, on peut avoir des boucles (2.3.2.2)

Dans la continuité du phénomène de **multi-stabilité** (voir 4.3.2.1), il arrive qu'un **SCN** (Synergistic Connection Network) n'aboutisse pas à un point fixe statique mais entre dans un **cycle** récurrent ou une forme d'**oscillation**. Au lieu de converger vers une configuration unique, la dynamique parcourt successivement plusieurs états qui se répètent de manière périodique (cycle de période T) ou quasi-périodique (oscillation complexe). Cette situation survient lorsque les **rétroactions positives** dans la mise à jour de $\omega_{i,j}$ sont suffisamment fortes pour qu'aucun état stationnaire ne s'impose durablement.

A. Rappel conceptuel : rétroactions positives et non-linéarités

Lorsque, dans la règle de mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$, la contribution qui accroît $\omega_{i,j}$ se trouve amplifiée dès que $\omega_{i,j}$ est déjà élevée, on parle de **rétroaction positive**. Si cette force d'amplification n'est pas contrebalancée par un mécanisme stabilisateur (par exemple une décroissance assez puissante, ou une inhibition globale), la trajectoire de $\omega_{i,j}$ peut se mettre à fluctuer au lieu de s'ancrer à une valeur fixe.

Ces comportements de fluctuation ou de “va-et-vient” sont généralement liés à des **non-linéarités**. Dans un **DSL** (Deep Synergy Learning), plusieurs ingrédients non linéaires peuvent intervenir : la mise à jour multiplicative ($\omega_{i,j}(t+1) = \omega_{i,j}(t) \times (1 + \dots)$), l'inhibition compétitive qui ajoute un terme $-\gamma \sum_{k \neq j} \omega_{i,k}$, ou encore la saturation (ω_{\max}). Les théories des **systèmes dynamiques** enseignent que de tels couplages non linéaires suffisent à produire des solutions périodiques (cycles limites) ou, plus rarement, des régimes chaotiques.

B. Caractérisation mathématique : cycle de période T

Sur le plan purement mathématique, un **cycle** de période T pour la dynamique

$$\boldsymbol{\omega}(t+1) = \mathbf{F}(\boldsymbol{\omega}(t))$$

se définit comme une suite de configurations $\{\boldsymbol{\omega}^*(1), \dots, \boldsymbol{\omega}^*(T)\}$ telle que

$$\mathbf{F}(\boldsymbol{\omega}^*(t)) = \boldsymbol{\omega}^*((t \bmod T) + 1), \quad t = 1, \dots, T,$$

et $\boldsymbol{\omega}^*(T+1) = \boldsymbol{\omega}^*(1)$. Autrement dit, après avoir appliqué \mathbf{F} T fois, on revient à l'état initial $\boldsymbol{\omega}^*(1)$.

Dans la pratique, un **cycle de période 2** est particulièrement fréquent : la configuration oscille en “ping-pong” entre deux états. Des cycles de période 3 ou plus longs existent également. Comme pour les points fixes, on peut analyser la **stabilité** de ces cycles en examinant l'évolution locale sur une orbite complète. Si l'analyse (via la matrice dérivée sur ce tour d'orbite) montre que les perturbations y restent contenues ou se contractent, le cycle est stable ; sinon, il peut s'avérer instable et ne se maintenir qu'en conditions idéales.

C. Illustrations dans un DSL

Dans un **DSL** de dimension significative, ces cycles peuvent se traduire par des phénomènes de “fluctuation de clusters”. Par exemple, un ensemble d’entités se regroupe momentanément en un bloc cohérent, puis la dynamique se déplace pour défaire ce cluster et en bâtir un autre, et la configuration revient ensuite au premier arrangement. Cette succession décrit un **cycle** : l’état global du SCN n’est jamais figé, mais repasse par un chemin périodique dans l’espace des $\{\omega_{i,j}\}$.

Les **transitions** causées par l’inhibition compétitive sont un autre vecteur classique d’oscillations. Imaginons un triangle {1,2,3} : si l’on renforce d’abord $\omega_{1,2}$ au détriment de $\omega_{2,3}$, puis que $\omega_{2,3}$ repart à la hausse et inhibe $\omega_{1,2}$, etc. Le système tourne en boucle et ne se stabilise jamais sur un unique cluster. Des simulations minimalistes montrent aisément cette alternance.

D. Signification et implications

Un tel **cycle** s’oppose à l’idée d’une convergence vers un unique arrangement stable. La présence de rétroactions positives et de couplages non linéaires signifie que le réseau peut “tourner” dans l’espace des configurations. Sur un plan pratique, cela peut compliquer l’exploitation des résultats : s’il n’y a pas d’état final unique, on ne peut pas conclure à une partition définitive. Cependant, ces oscillations peuvent être recherchées lorsqu’on désire un comportement de type “exploration continue” ou qu’on souhaite que le SCN “réfléchisse” en alternant plusieurs configurations plausibles.

Dans les grands systèmes, on peut même entrevoir des régimes plus complexes, y compris **chaotiques**, quoique dans un DSL standard, la plupart des mécanismes (décroissance τ , saturation, etc.) limitent souvent l’apparition d’un véritable chaos. On assiste plus fréquemment à des cycles stables (limite cycles) ou des oscillations amorties. Selon les objectifs, on peut vouloir favoriser ces oscillations (pour explorer plus de partitions) ou les réduire (pour atteindre un état stable), en ajustant la force de l’inhibition γ ou de la décroissance τ .

E. Conclusion

La **dynamique** d’un SCN n’est donc pas invariablement condamnée à converger vers une partition fixe : l’**exemple** des **cycles** ou oscillations récurrentes démontre que, sous l’effet de rétroactions positives et de couplages non linéaires, l’évolution de $\{\omega_{i,j}(t)\}$ peut se maintenir dans une orbite périodique plutôt que se figer. Dans certains contextes, cette oscillation correspond à une **exploration** alternée de plusieurs regroupements concurrents ; dans d’autres, elle peut constituer un **phénomène perturbateur** à modérer. Les sections ultérieures (4.3.2.3) discuteront des stratégies pour sortir d’une oscillation indésirable (par exemple via recuit simulé) ou, au contraire, pour les exploiter lorsqu’elles sont bénéfiques à la dynamique globale du réseau.

4.3.2.3. Stratégies pour détecter / caractériser un attracteur : qu’est-ce qu’un “bassin d’attraction” ?

Dans les sections précédentes (4.3.2.1 et 4.3.2.2), nous avons vu qu’un SCN (Synergistic Connection Network) pouvait admettre divers types d’attracteurs, tels que des **points fixes** (multi-stabilité) ou des **cycles** (oscillations). Une question fondamentale se pose alors : comment **déterminer** qu’un réseau a effectivement atteint un attracteur ? Et comment **décrire** la “zone d’influence” de cet attracteur, connue sous le nom de **bassin d’attraction** ?

Il s'agit d'un sujet central en **théorie des systèmes dynamiques** et en **analyse numérique**, qui, dans le cadre d'un DSL, peut être abordé à la fois par des méthodes analytiques (examen de dérivées, jacobienes) et par des méthodes empiriques (simulation, exploration des conditions initiales).

A. Qu'est-ce qu'un attracteur ?

Un **attracteur** se définit comme un **ensemble** (point fixe, cycle limite, orbite quasi-périodique, attracteur chaotique, etc.) auquel un grand nombre de **trajectoires** convergent — ou auquel elles se maintiennent de façon récurrente. Une fois la dynamique $\{\omega_{i,j}(t)\}$ entrée dans un voisinage de cet attracteur, elle n'en sort plus (ou s'en éloigne très difficilement).

Lorsque l'attracteur est un **point fixe** ω^* , on a

$$\omega^* = \mathbf{F}(\omega^*),$$

et, à partir d'initialisations proches, les trajectoires $\omega(t)$ convergent vers ω^* . Si l'attracteur est un **cycle** de période T , un ensemble de configurations $\{\omega^*(1), \dots, \omega^*(T)\}$ se répète : après T applications de \mathbf{F} , l'état du SCN retrouve exactement sa valeur initiale, et les trajectoires voisines viennent s'« attacher » à ce cycle.

Un **exemple** en DSL pourrait être la formation définitive d'un cluster stable, ou bien une oscillation entre deux ou trois « configurations » récurrentes dans lesquelles les liaisons fortes changent tour à tour.

B. Détection et caractérisation numérique

1. Recherche d'un point fixe

En pratique, on itère la dynamique

$$\omega(t+1) = \mathbf{F}(\omega(t))$$

et on surveille l'évolution du **pas** :

$$\|\omega(t+1) - \omega(t)\|.$$

Si cette norme devient (et reste) inférieure à un seuil ε suffisamment petit, on conclut à l'approche d'un **point fixe**. Concrètement, on peut poser une condition comme

$$\|\omega(t+1) - \omega(t)\| < \varepsilon \quad \text{et} \quad \|\omega(t+\Delta t) - \omega(t)\| < \varepsilon \quad \text{sur quelques pas successifs,}$$

pour s'assurer qu'aucun ré-emballlement ne se produit. On déclare alors la convergence vers un **état stationnaire** (attracteur fixe).

2. Recherche d'un cycle

Pour découvrir un cycle de période T :

204. On enregistre les configurations $\omega(t)$ sur une certaine « fenêtre » de temps (par exemple, sur un buffer circulaire).

205. On compare $\omega(t)$ et $\omega(t + T)$ via une certaine norme (comme $\|\cdot\|$). Si, pour un T donné, on trouve que $\omega(t + T) \approx \omega(t)$ pour plusieurs valeurs de t , on en conclut l'existence d'une **périodicité** T .

Un cas particulier est la **période 2** (un « ping-pong » entre deux états) : on vérifie si

$$\omega(t + 2) \approx \omega(t).$$

Dans des systèmes plus complexes, on peut chercher des traces d'**orbites** de période 3 ou plus, ce qui nécessite des tests plus élaborés.

3. Problème de la dimension dans un DSL

Un **SCN** implique $O(n^2)$ liaisons $\omega_{i,j}$ pour n entités, et la dimension de ω est potentiellement énorme. Pour détecter un attracteur numériquement, on restreint souvent :

- Soit on **échantillonne** un sous-ensemble représentatif de liaisons,
- Soit on considère un **indicateur global** (énergie $J(\omega)$, modularité, etc.) permettant de repérer la stagnation ou la périodicité. Ces approches restent des heuristiques, mais elles fonctionnent bien en pratique pour savoir si le système s'est stabilisé (point fixe) ou est entré dans un cycle repérable.

C. Bassins d'attraction : définition et intuition

Le **bassin d'attraction** associé à un attracteur A est l'ensemble de toutes les initialisations $\omega(0)$ dont la trajectoire finit par atteindre (ou approcher) A . Formulé plus formellement,

$$\mathcal{B}(A) = \{\omega_0 \mid \lim_{t \rightarrow \infty} \text{dist}(\omega(t), A) = 0\},$$

c'est-à-dire que toute trajectoire commençant dans $\mathcal{B}(A)$ aboutira dans A . Quand le DSL présente plusieurs attracteurs (points fixes ou cycles), chacun possède son **bassin** propre, et ces bassins se partagent l'espace initial via des **frontières** potentiellement complexes.

Savoir qu'un attracteur A existe est une chose, mais **savoir** dans quel bassin se trouve l'état initial $\omega(0)$ indique vers **quel** attracteur la dynamique ira. Deux initialisations très voisines peuvent se trouver dans deux bassins différents, entraînant l'aboutissement à deux **configurations** finales radicalement différentes. Cela met en évidence la **sensibilité** aux conditions initiales possible dans un DSL non linéaire.

D. Méthodes pratiques pour étudier bassins et attracteurs

Une façon simple et courante d'**explorer** les bassins d'attraction consiste à lancer de **nombreuses** simulations, chacune avec une initialisation $\omega(0)$ distincte (p. ex. tirée aléatoirement). On fait tourner la dynamique jusqu'à la détection d'un attracteur (point fixe ou cycle). On étiquette alors chaque initialisation par l'attracteur atteint et on essaie de visualiser (en réduction de dimension ou autrement) la zone initiale associée. On obtient ainsi un **maillage** approximatif du paysage des attracteurs.

Dans certains DSL, on dispose d'une **fonction d'énergie** $J(\omega)$ (chap. 2.4) qui, bien qu'elle ne soit pas toujours strictement un potentiel, peut fournir un repère. Si J descend globalement au fil des

itérations, on remarque où elle se **stabilise** ou **oscille**. Les vallées de \mathcal{J} correspondent en général à des points fixes ou des cycles. On peut alors repérer la “profondeur” de ces vallées et apprécier la **coexistence** d’attracteurs si plusieurs minima existent.

E. Conclusion

Les stratégies pour **déetecter** un attracteur (ou un cycle) dans un **SCN** se ramènent à :

- **Observer** la trajectoire $\omega(t)$ et vérifier sa stagnation ou sa périodicité,
- **Explorer** plusieurs initialisations pour connaître les **bassins d’attraction**,
- **Mesurer** la stabilité (jacobienne locale, écart entre $\omega(t)$ et $\omega(t+1)$ ou $\omega(t+T)$, etc.).

L’idée-clé est que la **non-linéarité** d’un DSL ouvre la voie à de multiples attracteurs (points fixes, cycles, régimes complexes) et rend le comportement **dépendant** des conditions initiales et du **flux** (modifications de synergie, insertion d’entités). Cette pluralité d’attracteurs apporte un grand **potentiel** d’adaptation et de diversité, tout en nécessitant une **analyse** méthodique pour comprendre vers où la dynamique d’un SCN se dirige et dans quels **bassins** elle évolue.

4.3.3. Stabilisation vs. Dynamisme Continu

4.3.3.1. Scénario Stationnaire : Synergie Fixe, on Converge vers un Arrangement Stable

Le **Deep Synergy Learning** (DSL) suppose en général que la **synergie** $S(i,j)$ entre entités i et j peut évoluer si la représentation des entités change (embeddings, règles symboliques, etc.) ou si de nouvelles entités apparaissent. Cependant, il existe un *cas plus simple* où la **synergie** $S(i,j)$ reste **constante** au cours du temps : aucun recalculation d’embeddings, pas de nouveau nœud, pas de modification des règles symboliques. On parle alors d’un **scénario stationnaire**, dans lequel la **fonction F** (décrivant l’évolution des pondérations ω) ne dépend que de ω elle-même et non d’autres facteurs changeants. Mathématiquement, on a un **système** :

$$\omega(t+1) = \mathbf{F}(\omega(t)),$$

où **F** demeure **invariante** dans le temps. Il en résulte qu’on peut étudier la **convergence** vers un état final ω^* stable, sans se soucier de modifications extrinsèques de la synergie.

A. Équation de Mise à Jour sous Synergie Constante

Lorsqu’on parle de **scénario** stationnaire, la **synergie** $S(i,j)$ ne varie pas avec t . Cela signifie que, pour tout couple d’entités (i,j) , on a une valeur **fixe** $S(i,j)$. La **règle** DSL devient alors purement **dynamique** sur ω .

Un schéma particulièrement simple (chap. 4.2.1) est la mise à jour **additive** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

dans lequel η est un taux d'apprentissage et τ un coefficient de décroissance (pour éviter la croissance illimitée). Puisque $S(i,j)$ est **constant**, chaque $\omega_{i,j}$ s'ajuste en se rapprochant d'une “valeur-cible”.

Si $S(i,j)$ est plus **grand** que $\tau \omega_{i,j}(t)$, on ajoute un incrément positif. Inversement, si $\omega_{i,j}(t)$ dépasse $S(i,j)/\tau$, le terme devient négatif.

On peut montrer qu'un **point fixe** se situe en $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$. Une fois $\omega_{i,j}$ parvenu à cette valeur, l'update $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ s'annule. Sur le plan **opérationnel**, cela signifie que la **force** de la liaison (i,j) atteint un équilibre où la synergie (qui tend à augmenter $\omega_{i,j}$) compense exactement la décroissance imposée par $\tau \omega_{i,j}$.

B. Convergence vers un Arrangement Stable

Dans l'ensemble, la **dynamique** stationnaire produit une **stabilisation** de la matrice $\{\omega_{i,j}\}$. Les liens se rapprochent de valeurs d'équilibre ($S(i,j)/\tau$) ou de 0 (si $S(i,j) = 0$). On parle alors d'un **arrangement stable** : plus rien ne bouge, car la synergie ne change pas et la mise à jour DSL a abouti à un **point fixe**.

Sur chaque liaison $\omega_{i,j}$, si la règle est **additive** :

$$\omega_{i,j}(t+1) - \omega_{i,j}(t) = \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Cela se réécrit comme un **processus** de relaxation vers $\frac{S(i,j)}{\tau}$:

206. Tant que $\omega_{i,j}(t) < \frac{S(i,j)}{\tau}$, la différence est positive, $\omega_{i,j}$ **monte**.

207. Si $\omega_{i,j}(t) > \frac{S(i,j)}{\tau}$, la différence est négative, $\omega_{i,j}$ **baisse**.

Pourvu que $\eta \tau < 2$ et sans autres termes (ex. inhibition non linéaire), on obtient un **comportement** contractant qui converge.

Une fois **tous** les liens convergés, on dispose d'une **matrice** ω^* avec $\omega_{i,j}^* \approx \frac{S(i,j)}{\tau}$. Les **entités** i et j pour lesquelles $S(i,j)$ est élevée deviennent fortement liées, celles pour lesquelles $S(i,j) \approx 0$ ont des $\omega_{i,j} \approx 0$. Si l'on applique un **seuil** ou une forme d'inhibition, on peut voir émerger des **clusters** plus marqués.

C. Interprétation en Termes de Clusters

Dans un SCN stationnaire, un jeu de **données** (ou de “synergies” prédéterminées) aboutit à une **partition** implicite. Les liens **forts** à l'équilibre forment des **groupes** d'entités fortement interconnectées.

Si la **matrice** $\{S(i,j)\}$ reflète une structure de type “clusters” (ex. entités proches en embedding), alors la **dynamique** additive fait en sorte que les entités internes à un cluster se retrouvent avec des **liens** $\omega_{i,j}$ élevés. Entre clusters, si $S(i,j)$ est faible, $\omega_{i,j}$ reste bas.

Ce phénomène s'apparente à une forme de **clustering** non supervisé (cf. chap. 2), où le résultat final ω^* exprime la **partition** latente.

Si la règle de mise à jour est **linéaire**, comme ci-dessus, le **point fixe** est souvent unique et découle aisément de $\frac{S(i,j)}{\tau}$. Mais dès qu'on ajoute un **terme** d'inhibition plus complexe (compétition entre liens) ou des non-linéarités, il peut exister **plusieurs** attracteurs stables (chacun correspondant à un arrangement cluster différent). Dans ce cas, le **SCN** stationnaire peut se figer dans l'un ou l'autre arrangement selon l'initialisation ou des bruits aléatoires (cf. chap. 4.3.2 sur la multiplicité d'attracteurs).

D. Liens Formels avec la Descente d'Énergie

Dans la plupart des **implémentations** DSL, on peut définir une **énergie** $J(\omega)$ (ex. $-\sum_{i,j} \omega_{i,j} S(i,j) + \tau/2 \sum_{i,j} (\omega_{i,j})^2$) qui, en régime stationnaire, atteint un **minimum** local. Dans ce scénario, la synergie $S(i,j)$ ne changeant pas, la **descente** de J n'est pas contrariée par des révisions de synergie, et le **SCN aboutit** à un minimum local stable (donc un arrangement final ω^*).

On dit que $J(\omega)$ est **fixe** car la fonction $\omega \mapsto J$ ne dépend pas d'autres variables évolutives. La **descente** locale de J se fait par une dynamique de relaxation, menant au point d'équilibre.

Si on n'ajoute pas de **bruit** ou de **compétition** trop forte, la convergence est généralement assurée (chaque liaison converge vers $S(i,j)/\tau$). En revanche, des schémas multiplicatifs ou inhibiteurs peuvent générer des **oscillations** transitoires avant la stabilisation. Mais in fine, dans un cadre stationnaire, on **s'installe** dans un attracteur.

Conclusion (4.3.3.1)

Dans un **scénario** où la **synergie** $S(i,j)$ demeure **fixe** (aucune mise à jour d'embeddings, aucune apparition de nouveaux nœuds), la **dynamique** DSL (Deep Synergy Learning) se ramène à un **système** autonome sur ω . Cela aboutit typiquement à un **arrangement stable** : chaque liaison $\omega_{i,j}$ converge vers une valeur d'équilibre (ex. $S(i,j)/\tau$ pour le modèle additif), et la **matrice** ω cesse de bouger. On obtient alors une **partition** ou un **réseau** stationnaire mettant en évidence les affinités structurelles dictées par S .

Cette situation *contraste* avec les **cas dynamiques** (voir § 4.3.3.2) où la **synergie** peut se redéfinir au fil du temps : dans le scénario stationnaire, on a une **descente** d'énergie *pure* (ou une relaxation) sans que les entrées S ne perturbent l'équation au fur et à mesure. Le **SCN** se **fige** sur un "**arrangement stable**", révélant la configuration la plus compatible avec la synergie statique.

4.3.3.2. Scénario Évolutif : synergie modifiée par de nouvelles données / événements, le réseau ne se fige jamais (apprentissage continu)

Dans la section précédente (4.3.3.1), l'analyse d'un **SCN** (Synergistic Connection Network) en **scénario stationnaire** (avec une fonction de synergie $S(i,j)$ fixée) montrait comment le réseau peut converger vers un **arrangement stable**. Cependant, dans bien des cas pratiques, cette hypothèse de synergie *constante* s'avère trop restrictive. En effet, dans un système réel, de nouvelles entités peuvent faire leur apparition, des attributs ou des représentations peuvent évoluer

(embeddings ajustés, règles modifiées), et l'environnement lui-même peut changer. Ainsi, la **fonction** $S(i, j)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j devient **dépendante du temps**, ou se voit “redéfinie” à chaque étape. On se retrouve alors dans un régime d'**apprentissage continu**, où le **réseau** ne parvient pas forcément à une configuration finale unique et peut rester en **réorganisation** perpétuelle.

A. Origines de l'Évolution de la Synergie

Dans un *DSL* fonctionnant en mode “flux” (stream), de nouveaux nœuds \mathcal{E}_{n+1} apparaissent régulièrement : par exemple, de nouveaux documents, de nouveaux utilisateurs, ou encore de nouveaux capteurs. Chaque fois qu'un tel nœud est inséré, on doit :

208. Calculer ou initialiser les liens $\omega_{(n+1), i}$ pour $i = 1, \dots, n$.
209. Mettre à jour ou réévaluer la **synergie** $S(n + 1, i)$.

Ces opérations peuvent rompre l'équilibre local déjà en place : en liant \mathcal{E}_{n+1} à divers nœuds, on réoriente éventuellement la structure de **clusters** ou de **communautés** formées jusque-là.

Même si le nombre de nœuds reste le même, la *représentation* d'un nœud peut changer. Par exemple, un *embedding* sub-symbolique \mathbf{x}_i se re-entraîne et prend de nouvelles valeurs, modifiant les **distances** ou **similarités** $\|\mathbf{x}_i - \mathbf{x}_j\|$ ou encore $\mathbf{x}_i \cdot \mathbf{x}_j$. De même, un nœud symbolique peut ajouter ou retrancher une règle (logique), entraînant un **changement** dans la fonction $S_{\text{sym}}(i, j)$. On assiste alors à une **synergie** redéfinie en cours de route, ce qui oblige la mise à jour $\omega_{i,j}$ à se poursuivre en s'adaptant à cette nouvelle donne.

Dans des scénarios temps réel (ex. robotique, environnement capteur), les conditions (luminosité, bruit, configuration spatiale, etc.) varient de façon incessante. La synergie $S(i, j, t)$ — qui peut refléter la cohérence instantanée de deux signaux sensoriels — doit être recalculée en continu, empêchant le SCN de “figer” un arrangement final.

B. Impact sur la Dynamique du Réseau

Contrairement au **cas stationnaire** où l'on peut exhiber un point fixe ω^* , ici la loi

$$\boldsymbol{\omega}(t + 1) = \mathbf{F}(\boldsymbol{\omega}(t), S(\cdot, \cdot, t))$$

est *non autonome*. On ne peut plus parler de $\boldsymbol{\omega}^* = \mathbf{F}(\boldsymbol{\omega}^*, S(\cdot, \cdot))$ de manière permanente, car S varie à chaque étape. En conséquence, le réseau peut évoluer *indéfiniment* :

- Il peut se reconfigurer au gré des insertions de nœuds,
- Il peut réajuster ses liens quand un embedding \mathbf{x}_i change,
- Il peut basculer d'un type de partition à un autre selon l'arrivée de données ou l'évolution d'un paramètre (inhibition, température de recuit, etc.).

Cette logique rejoue l'idée d'un **apprentissage en ligne** : à chaque itération (ou petit batch), on :

210. Observe les changements (nouveaux nœuds, embeddings modifiés...),

211. Recalcule (au moins partiellement) la fonction $S(i, j, t)$,
212. Réalise un *delta update* $\Delta\omega_{i,j}(t)$ pour amener la pondération vers une nouvelle cohérence.

Le SCN reste ainsi *flexible* et *vivant*, réorientant ses clusters ou sous-structures au fil du temps.

Du fait que S n'est plus constant, le SCN peut parfois osciller : par exemple, un nœud dont la synergie vis-à-vis d'un cluster se renforce temporairement, puis s'affaiblit, puis se renforce, etc. On peut assister à des "phases" transitoires de stabilisation, suivies d'une "transition" quand une nouvelle *vague* de données survient.

C. Considérations Mathématiques : Systèmes Dynamiques Non Autonomes

Le système s'écrit

$$\boldsymbol{\omega}(t+1) = \mathbf{F}(\boldsymbol{\omega}(t), \mathbf{p}(t)),$$

où $\mathbf{p}(t)$ dénote la configuration (paramètres, représentations) à l'instant t . Les propriétés de stabilité ou de convergence dépendent du *trajectoire* $\{\mathbf{p}(t)\}$. Si $\mathbf{p}(t)$ varie lentement, on peut espérer un "quasi-équilibre" mobile ; si elle varie rapidement, la $\boldsymbol{\omega}(t)$ peut rester en turbulence, sans se poser.

On peut mobiliser des cadres "skew-product" en théorie des systèmes dynamiques non autonomes, ou des approches d'**adaptation** en contrôle. En pratique, la plupart des implémentations reposent sur des *simulations*, où l'on met à jour $\omega_{i,j}(t)$ de façon incrémentale, tout en recomputant $S(\cdot, \cdot, t)$. On observe l'existence ou non d'un "pseudo-ordre", de *phases* où le réseau se stabilise partiellement, puis se réadapte, etc.

Dans un système de recommandation, chaque itération introduit un *nouvel utilisateur* ou un *nouvel item*. La synergie $S(u, i)$ entre l'utilisateur u et l'item i doit être définie (ou recalculée). Cela conduit à des réajustements $\omega_{u,i}$, puis redistribue la force sur d'autres liens déjà existants. Cette dynamique se poursuit indéfiniment, car la base de données utilisateurs/items ne cesse d'évoluer.

D. Applications Pratiques

Recommandation	en	Ligne.
On ne peut jamais finaliser un "training" une fois pour toutes, car l'arrivée constante d'utilisateurs ou d'items (ou la modification des préférences) rend la matrice ω obsolète si elle n'est pas ajustée.	en	Ligne.
Le DSL apporte une plasticité native : chaque nouveau flux de données modifie la synergie S , le SCN réapprend et réorganise les clusters, tout en restant opérationnel.	en	Ligne.

Robotique.

Les flux sensoriels évoluent (conditions d'éclairage, types d'objets manipulés), les capteurs ou actions peuvent se paramétrier. Le SCN recalcule la *pertinence* des liens sensorimoteurs. Il n'existe pas d'état final, seulement une suite d'états adaptatifs.

Dialogue.

Un chatbot perfectionne ses embeddings (BERT, GPT) à mesure qu'il ingère de nouvelles

conversations. La *synergie textuelle* S n'est donc pas fixe. Le SCN en charge d'associer des thèmes ou des intentions réorganise ses liaisons ω en continu, parfois sans stabilisation.

E. Conclusion : Un Réseau en Mouvement

Dans ce **scénario évolutif**, la fonction de **synergie** $S(i,j)$ se modifie au fil du temps — parce que de *nouvelles entités* arrivent, parce que la *représentation* interne de chaque entité change, ou parce que l'environnement est *non stationnaire*. Il s'ensuit que le **SCN** (Synergistic Connection Network) est perpétuellement en **adaptation** :

- Il ne converge **pas** nécessairement vers un unique arrangement stable,
- Il peut manifester des **transitions** majeures au gré des données,
- Il s'inscrit dans un **apprentissage en ligne**, où l'on actualise ω en continu, révélant à chaque instant la configuration de clusters la plus cohérente avec l'état courant de la synergie.

Cela rend le **DSL** particulièrement adéquat pour des **contextes temps-réel** ou des **systèmes “vivants” et “dynamiques”**, où le concept de “fin d’entraînement” n’existe pas. Le *réseau* se perçoit comme un organisme “en mouvement” : toujours prêt à **réorganiser** ses groupements (clusters) selon les changements constatés dans S .

4.3.3.3. Exemples de courbes d'évolution $\omega_{i,j}(t)$ montrant la formation progressive d'un cluster

Pour mieux **visualiser** ce qui se passe durant la **dynamique** d'auto-organisation, il est souvent instructif de tracer au fil du temps l'**évolution** de quelques pondérations $\omega_{i,j}(t)$. Dans un **SCN** (Synergistic Connection Network) à grande échelle (n élevé), on se limite généralement à un sous-ensemble représentatif de liens ou à des **indicateurs** agrégés, mais le principe reste le même : observer comment certains liens $\omega_{i,j}(t)$ passent de valeurs proches de 0 (ou très faibles) à des niveaux élevés (voire saturés), tandis que d'autres se réduisent ou stagnent à des valeurs négligeables.

L'idée est de **montrer la formation progressive** d'un **cluster** : au début, les liens sont dispersés ou quasi nuls, puis la dynamique sélectionne quelques couples (i,j) pour se renforcer, révélant ainsi un **groupe cohérent**.

A. Scénario simple à cinq entités

Considérons un exemple **pédagogique** où l'on dispose de cinq entités $\{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3, \mathcal{E}_4, \mathcal{E}_5\}$. Supposons qu'en **réalité** la **synergie** $S(i,j)$ soit nettement **élevée** entre les paires du **sous-groupe** $\{1,2,3\}$ et plus **faible** pour leurs liens avec $\{4,5\}$. Autrement dit, il existe un “vrai” petit cluster $\{1,2,3\}$, alors que $\{4,5\}$ en est assez distant.

Pour amorcer la simulation, on pose :

$$\omega_{i,j}(0) \approx 0 \quad (\text{avec éventuellement un faible bruit aléatoire}).$$

Cette **condition initiale** illustre l'idée que, avant toute itération, les pondérations $\omega_{i,j}$ sont de niveau très bas (ou nuls) ; la dynamique du **DSL** va alors opérer progressivement la mise en place de liens plus ou moins forts.

Lorsque l'on **applique** la règle de mise à jour (par exemple la variante additive) :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

les paires (1,2), (1,3) et (2,3) commencent à s'**amplifier** si $S(1,2)$, $S(1,3)$, et $S(2,3)$ sont élevés. En pratique, dès les premières itérations, on observe une **croissance** notable de $\omega_{1,2}(t)$, $\omega_{1,3}(t)$, $\omega_{2,3}(t)$. La dynamique va rapidement les rapprocher de leur "point d'équilibre" $\frac{S(i,j)}{\tau}$ (hypothèse sans inhibition).

En revanche, pour les paires mixtes (c'est-à-dire entre {1,2,3} et {4,5}), la valeur de la synergie $S(i,j)$ est plus faible. De ce fait, les mises à jour successives:

$$\Delta\omega_{i,j}(t) = \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

restent proches de zéro, voire négatives si $\omega_{i,j}$ a eu un léger sursaut de bruit initial. Les liens (1,4), (2,5), etc., finissent donc par **stagner** à un niveau très faible ou décroître pour s'approcher de zéro.

Pour **visualiser** cette progression, on peut tracer, à chaque itération t , la valeur de $\omega_{i,j}(t)$ pour diverses paires (i,j) . Un exemple :

- $\omega_{1,2}(t)$: part d'environ 0 au temps $t = 0$, **monte** assez vite dans les dix ou vingt premières itérations, puis **se stabilise** dans une région proche de $S(1,2)/\tau$.
- $\omega_{1,4}(t)$: demeure très faible ; si le bruit initial la poussait légèrement, la correction $\eta [S(1,4) - \tau \omega_{1,4}(t)]$ devient vite négative ou quasi nulle, la ramenant vers zéro.

Cette différence de **trajectoire** entre paires intra-{1,2,3} et paires hors-cluster (par ex. {1,4}, {2,5}) révèle le mécanisme **sélectif** : la synergie forte enclenche un renforcement local des liens, la synergie faible induit leur quasi-extinction.

En examinant les **graphes** $\omega_{i,j}(t)$ en fonction du temps, on distingue alors deux "familles" de courbes :

- **Famille haute** : $\omega_{1,2}(t), \omega_{1,3}(t), \omega_{2,3}(t)$. Ces liens croissent puis se maintiennent à un plateau **élevé**, typique de la **cohésion** interne au cluster {1,2,3}.
- **Famille basse** : $\omega_{1,4}(t), \omega_{2,5}(t), \omega_{3,4}(t)$, etc. Celles-ci restent très proches de **zéro** ou se tassent rapidement, montrant la **séparation** d'avec {4,5}.

Après un certain nombre d'itérations, on obtient une **matrice ω** où {1,2,3} s'affiche comme un "**micro-bloc**" densément interconnecté, tandis que {4,5} n'y est lié que par des poids très bas (ou à un autre cluster si {4,5} se rapprochent entre eux).

Lorsque la **dynamique** se stabilise (ou atteint un état stationnaire), on a, pour les paires intra- $\{1,2,3\}$, des valeurs $\omega_{i,j}^* \approx \frac{S(i,j)}{\tau}$ (si le modèle est purement additif, sans inhibition). Entre $\{1,2,3\}$ et $\{4,5\}$, on constate $\omega \approx 0$.

Dans un **SCN** plus vaste, on peut observer la **même** logique se reproduire sur des groupes plus grands : si la synergie est suffisante, ils émergent comme clusters plus ou moins compacts, et la visualisation des pondérations confirme l'allure “en blocs”.

Cet **exemple** simplifié illustre la **progression** d'un DSL qui part d'une initialization “faible” $\omega(0) \approx 0$:

- Les **liens** correspondant à de **fortes** synergies se **renforcent**,
- Les autres se **réduisent** ou restent proches de zéro,
- Au bout d'un nombre d'**itérations**, on **observe** clairement la formation d'un ou plusieurs **clusters** (ici, $\{1,2,3\}$).

C'est en examinant les **courbes** $\omega_{i,j}(t)$ qu'on **saisit** la nature dynamique de la convergence : chaque liaison évolue selon la différence $[S(i,j) - \tau \omega_{i,j}(t)]$. Ce **principe** se généralise à de plus grands ensembles, avec des patterns plus complexes, mais la mécanique reste la même : fort $S(i,j) \Rightarrow$ renforcement, faible $S \Rightarrow$ liaison quasi éteinte, aboutissant à une **auto-organisation** en groupes cohérents.

B. Exemples de variations en cas d'inhibition ou de multiplicatif

Dans le cadre du **DSL** (Deep Synergy Learning), l'évolution des pondérations $\omega_{i,j}(t)$ peut prendre des profils temporels sensiblement différents, notamment lorsqu'on introduit des **mécanismes** tels que l'**inhibition compétitive** ou une **mise à jour multiplicative**. Il s'ensuit des **dynamiques** qui peuvent produire des comportements parfois plus contrastés, allant de “va-et-vient” marqués à de véritables “sauts” exponentiels.

Lorsque l'on ajoute de l'**inhibition latérale** (voir la section 4.2.2.2) dans la règle de mise à jour, on introduit un **terme** $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ qui pénalise la croissance conjointe de plusieurs liens partant de la même entité \mathcal{E}_i . Mathématiquement, la mise à jour $\omega_{i,j}(t+1)$ peut alors s'écrire :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau \omega_{i,j}(t) - \gamma \sum_{k \neq j} \omega_{i,k}(t) \right],$$

où $\gamma > 0$ représente l'intensité de l'inhibition.

Sur le plan dynamique, si un lien $\omega_{i,2}$ se met à croître rapidement (parce que la synergie $S(i,2)$ est forte), il “inhibe” ou freine la croissance d'autres liens partant de la même entité \mathcal{E}_i . Les courbes $\omega_{i,j}(t)$ peuvent alors révéler un **comportement compétitif** : tandis que $\omega_{i,2}$ se renforce, $\omega_{i,3}$ ou $\omega_{i,5}$ stagnent ou diminuent, même si leur synergie $S(i,3)$ ou $S(i,5)$ n'est pas complètement négligeable.

Il est possible d'observer des **va-et-vient** : par moment, $\omega_{i,2}(t)$ progresse, puis si la synergie $S(i, 3)$ n'est pas trop faible, $\omega_{i,3}(t)$ profite d'un relâchement d'inhibition pour se renforcer à son tour, ce qui à nouveau rabaisse $\omega_{i,2}$, etc. Ces oscillations (ou successions de prises d'avantage) sont plus ou moins marquées selon les valeurs de γ , η et τ .

Sur un graphe “ $\omega_{i,j}(t)$ vs. t ”, on discerne clairement des courbes qui croissent, puis s'infléchissent sous l'action de l'inhibition, favorisant l'émergence d'un “gagnant” local. Cette situation peut conduire à des clusters plus nets (car un nœud \mathcal{E}_i n'entretient qu'un faible nombre de liens forts), mais aussi à des **phases de compétition** prolongées.

La **règle multiplicative** (voir la section 4.2.2.1) modifie la façon dont un lien $\omega_{i,j}$ se met à jour, en introduisant une **proportionnalité** à sa valeur actuelle. Au lieu d'ajouter un terme $\eta[S(i,j) - \tau \omega_{i,j}(t)]$ (cas additif), on applique souvent une formule de la forme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \times (1 + \eta S(i,j)) - \beta \omega_{i,j}(t),$$

ou même une exponentiation directe (par exemple $\omega_{i,j}(t+1) = \omega_{i,j}(t) \exp\{\alpha[S(i,j) - \tau \omega_{i,j}(t)]\}$).

Une caractéristique majeure de la mise à jour multiplicative tient à sa **dépendance** au niveau actuel de $\omega_{i,j}$. Si un lien est déjà modérément fort, la rétroaction positive agit de manière accélérée (“the rich get richer”). Inversement, si $\omega_{i,j}$ est initialement très faible, il peut rester collé à presque zéro, car l'incrément dépend de sa valeur courante.

Les simulations montrent souvent des **sauts exponentiels** : une liaison $\omega_{i,j}$ peut demeurer quasi nulle pendant plusieurs itérations, puis, dès qu'un léger bruit ou une fluctuation l'élève au-dessus d'un petit **seuil**, elle monte subitement (de façon exponentielle) jusqu'à un plateau ou une saturation ω_{\max} . Sur un graphe “ $\omega_{i,j}(t)$ vs. t ”, la courbe paraît stable proche de 0, puis effectue un **bond** rapide.

Ce type de mise à jour produit des **clusters** très tranchés : une fois un lien enclenché (parce qu'un nœud a franchi le seuil de synergie), il grandit fortement, reléguant d'autres liaisons concurrentes à un niveau bas. Dans un DSL, on peut donc aboutir à des partitions plus brutales où quelques liens dominent massivement, tandis que d'autres restent quasi inexistantes.

Imaginons que $\omega_{1,2}(t)$ dépasse 0.02 à l'itération $t = 10$ (à cause d'une petite synergie favorable). La multiplicativité enclenche alors une **croissance** en quelques pas, menant $\omega_{1,2}$ à un niveau élevé (0.7, 0.8...). Dans le même temps, $\omega_{1,3}$ ou $\omega_{2,4}$ ne franchissent pas leur seuil, de sorte qu'ils demeurent sous 0.01. Le cluster “(1,2)” se forme ainsi quasi instantanément, tandis que (1,3) reste “ignoré”.

Conclusion générale.

Les **variations** introduites par l'**inhibition** ou la **mise à jour multiplicative** façonnent de manière singulière les **courbes** $\omega_{i,j}(t)$. L'inhibition compétitive génère souvent des **dynamiques** de compétition plus ou moins oscillantes, favorisant la formation de liens “gagnants” et la suppression d'autres liaisons sortantes. La multiplicativité, quant à elle, exacerbe le phénomène de **seuil** et d'auto-amplification, conduisant parfois à des “sauts” soudains dans la courbe d'évolution d'un

lien. Dans un **DSL**, ces mécanismes rendent la clusterisation plus marquée (voire brutale) et peuvent mener à des partitions plus segmentées, mais aussi à des **conflits** ou “résonances” selon les paramètres η , β , γ . En pratique, la visualisation des trajectoires $\omega_{i,j}(t)$ aide à **comprendre** le cheminement dynamique vers la formation (ou la disparition) de chaque **lien fort**.

C. Mise en forme de ces courbes

Lorsqu'on souhaite **visualiser** l'évolution des pondérations $\omega_{i,j}(t)$ au cours des itérations, on peut procéder en plusieurs étapes afin de tirer le meilleur parti des **courbes** tracées. Les sous-sections suivantes décrivent ce processus et mettent en évidence son intérêt pour **comprendre** la dynamique d'**auto-organisation** du réseau.

Un **premier** aspect consiste à **stocker** les valeurs de chaque liaison $\omega_{i,j}$ à chaque itération $t = 0, 1, \dots, T$. Dans la pratique, on pourra :

- Définir un tableau ou un dictionnaire pour les $\omega_{i,j}(t)$.
- Après chaque mise à jour, on mémorise $\omega_{i,j}(t)$ pour un petit nombre de paires (i,j) d'intérêt (pour ne pas trop alourdir la mémoire si le réseau est grand).
- À l'issue des itérations (par exemple au bout de 50, 100 ou 500 pas), on dispose alors de la **trajectoire** complète $\omega_{i,j}(0), \omega_{i,j}(1), \dots, \omega_{i,j}(T)$.

Pour **sélectionner** les paires (i,j) à tracer, on identifie généralement :

- **Une paire** à forte synergie a priori (ou bien dont on anticipe qu'elle va se renforcer si les entités sont très similaires).
- **Une paire** à synergie moyenne (où l'on veut vérifier si le lien monte ou descend).
- **Une paire** à synergie faible (pour constater si $\omega_{i,j}$ demeure au voisinage de 0 ou s'éteint).

On peut ainsi **comparer** les comportements de quelques liaisons représentatives, sans tracer la totalité des $O(n^2)$ courbes du SCN.

Après avoir récupéré ces **trajectoires** $\omega_{i,j}(t)$, un usage courant est de produire un **graphique** (par exemple via matplotlib ou un outil similaire en Python) où l'axe horizontal représente l'itération t et l'axe vertical la valeur de $\omega_{i,j}(t)$.

On peut tracer **plusieurs** liaisons sur un **même** diagramme :

$$\omega_{1,2}(t), \quad \omega_{1,3}(t), \quad \omega_{4,5}(t), \quad \dots$$

Ces différentes **courbes** de couleur distincte illustrent alors, sur une même figure, comment certains liens **montent** rapidement quand la synergie est forte (et donc la mise à jour DSL les renforce), tandis que d'autres **restent** très bas ou **progressent** lentement.

Cette **représentation** multi-lignes met en évidence :

- Les **phases** d'augmentation : par exemple, une courbe qui s'élève dès les 5 premières itérations, suggérant une forte cohésion entre les entités correspondantes.

- Les **palières** ou saturations : il arrive qu’après un certain temps, une courbe se fixe autour d’une valeur (ex. $\omega_{1,2}(t) \rightarrow 0.8$), marquant la stabilisation d’un lien fort.
- Les **variations plus erratiques** ou “yo-yo” : elles indiquent soit des oscillations (chap. 4.3.2.2), soit un changement de la fonction synergie (scénario évolutif, chap. 4.3.3.2).

En examinant ces **trajectoires** de $\omega_{i,j}(t)$, on obtient une **indication** très claire de la situation dynamique :

- Si, au bout d’un nombre raisonnable d’itérations (par exemple 50 ou 100), les **valeurs** $\omega_{i,j}(t)$ montrent toutes un **plateau** ou une zone presque stationnaire, c’est un signe fort de **convergence** (scénario stationnaire du chap. 4.3.3.1). On peut alors considérer que le **SCN** a “appris” sa structure finale.
- À l’inverse, si certaines courbes **oscillent** (montées et descentes successives) ou **changent** de palier à plusieurs reprises, cela indique soit un **phénomène oscillatoire** au sens du chap. 4.3.2.2, soit un **scénario évolutif** (chap. 4.3.3.2) où la synergie S se modifie (arrivée de nouvelles entités, rafraîchissement des embeddings, etc.), empêchant un état unique de s’installer définitivement.

Cette **distinction** (plateau vs. oscillations/progression continue) s’avère cruciale pour **comprendre** le régime dans lequel le **SCN** se situe :

- **Régime stationnaire** : la topologie des liens se “fige” et on peut alors extraire de manière stable les clusters.
- **Régime non stationnaire** : les courbes ne se stabilisent pas, reflétant un **réseau** “vivant” ou “en apprentissage continu”, potentiellement réactif aux nouveaux flux de données.

Un **bénéfice** direct de tracer ces courbes est d’observer, “pas à pas”, **comment** un cluster se **construit** dans le SCN.

Si l’on voit **plusieurs** liaisons $\omega_{i,j}(t)$ associées à un **même** groupe d’entités \mathcal{C} (par exemple $\{1,2,3\}$) **augmenter** quasi simultanément, c’est la **marque** qu’un bloc se **renforce**. Dans le même temps, les liens de ces entités vers l’extérieur peuvent **stagner** ou **décroître**.

Cela illustre de manière **graphique** le phénomène d’**isolement** d’un cluster : $\omega_{1,2}(t), \omega_{1,3}(t)$ deviennent grandes, alors que $\omega_{1,k}$ pour $k \notin \{2,3\}$ restent faibles. La dynamique DSL favorise les “intra-liens” cohérents et punit (ou néglige) les “extra-liens” divergents.

Exemple d’évolution

- **Étapes initiales** : $\omega_{i,j}(0) \approx 0$. Pendant quelques itérations, toutes les courbes sont proches de zéro.
- **Montée** : pour les paires à haute synergie, leurs courbes **décollent** plus ou moins vite (selon η, τ) et peuvent atteindre une zone de plateau intermédiaire (ex. 0.5–0.7).
- **Stabilisation** : si rien ne change, on voit un palier final (par ex. 0.8) pour ces liens forts, alors que les liens “faibles” conservent une valeur modeste (0.1 ou <0.05).

- **Cluster** : on interprète $\omega_{1,2}, \omega_{1,3}, \omega_{2,3}$ comme hautes, signe d'un **cluster** {1,2,3}. Les liaisons de ces entités vers d'autres indices {4,5, ... } ne dépassant pas 0.15, on obtient un "mur" séparant {1,2,3} du reste.

Selon la **présence** ou non d'inhibition (compétition pour les liaisons sortantes), quelques **tweaks** peuvent survenir :

- *Inhibition latérale* : force un **choix** plus exclusif, où un lien se stabilise un peu au détriment d'un autre. $\omega_{1,3}$ peut donc baisser de 0.5 à 0.3 si $\omega_{1,2}$ grimpe à 0.8, etc.
- *Saturation ou clipping* : empêche une courbe de dépasser ω_{\max} . Visuellement, on voit la courbe buter sur une valeur-limite.

Au final, la **lecture** de ces tracés $\omega_{i,j}(t)$ donne une "histo^{ire}" de la construction d'un cluster, au lieu de juste observer l'état final $\omega_{i,j}(T)$.

Conclusion

La **mise en forme** des courbes $\omega_{i,j}(t)$ à travers un graphique (multi-lignes) est un **outil** particulièrement instructif pour **visualiser** :

- **La discrimination** progressive des liens : les paires à forte synergie se détachent, tandis que celles à synergie faible restent ou redescendent vers zéro.
- **Le scénario** de convergence : si on voit que les valeurs aboutissent à des plateaux stables, on conclut à un régime stationnaire.
- **Les signes** d'un régime non stationnaire ou oscillant : si les courbes ne parviennent pas à un plateau, on suspecte un **scénario évolutif** ou des **oscillations** (selon la topologie, l'inhibition, la variation des synergies...).

Cette **analyse** offre en outre un **témoignage** direct de l'émergence d'un **cluster** : on voit concrètement *quand* et *comment* les liaisons associées à un groupe s'élèvent en parallèle, donnant une **compréhension** dynamique du processus d'**auto-organisation** au sein du SCN.

4.4. Oscillations, Pseudo-Chaos et Méthodes de Contrôle

Les sections précédentes (4.3) ont montré que la **dynamique** d'un SCN (Synergistic Connection Network) pouvait, dans certains cas, conduire à une **stabilisation** (formation de clusters fixes), tandis que dans d'autres, elle pouvait demeurer en mouvement (scénario évolutif). Mais il existe une autre possibilité : le réseau entre dans des **oscillations** plus ou moins régulières ou même un régime "pseudo-chaotique". Dans ce chapitre 4.4, nous allons :

- **Analyser** les **causes** de ces oscillations ou de comportements complexes (4.4.1),
- **Proposer** des **stratégies** pour les **rompre** ou les **contrôler** (4.4.2) lorsque le but est de favoriser la convergence,
- **Présenter** enfin quelques **cas stochastiques** (4.4.3) où l'injection de bruit (recuit simulé, perturbations) peut aider le SCN à échapper à des minima ou à se stabiliser de façon plus globale.

Ce thème s'inscrit dans la continuité de la question des **attracteurs** (4.3.2) : outre des points fixes ou des cycles simples, on peut rencontrer des régimes de type "oscillations prolongées" ou "pseudo-chaos" si la rétroaction est assez puissante et la non-linéarité suffisamment marquée.

4.4.1. Pourquoi des Oscillations ou un Pseudo-Chaos ?

Malgré la tendance naturelle à la stabilisation (lorsqu'un terme $-\tau \omega_{i,j}$ freine la croissance), il peut arriver que la **mise à jour** des pondérations $\omega_{i,j}$ entretienne un **cycle** ou un **mouvement** quasi cyclique. De plus, dans des réseaux de forte dimension et dotés de mécanismes non linéaires, on observe parfois des régimes encore plus complexes, assimilables à du **pseudo-chaos** (une sensibilité marquée aux conditions initiales, une trajectoire ne se répétant pas exactement mais restant dans un comportement "erratique").

4.4.1.1. Rétroactions Positives Trop Intenses, Absence d'Inhibition (2.3.2.2)

Principe Général : un Effet de Boule de Neige

Dans les **formulations** de la mise à jour (qu'elles soient additives ou multiplicatives, voir chapitre 4.2), il est fréquent de disposer d'un terme du type $+\eta S(i,j)$ qui **encourage** la croissance du lien $\omega_{i,j}$ dès lors que la **synergie** $S(i,j)$ est élevée. Une fois que ce renforcement positif l'emporte sur la décroissance $-\eta \tau \omega_{i,j}$ (parce que τ est trop faible ou parce que $S(i,j)$ dépend déjà de $\omega_{i,j}$ de manière auto-amplificatrice), on peut observer un **effet boule de neige** :

$$\omega_{i,j}(t+1) > \omega_{i,j}(t) > \dots \text{ pendant plusieurs itérations.}$$

Autrement dit, chaque itération accentue encore plus la croissance d'un lien déjà en essor, de sorte que $\omega_{i,j}$ s'élève presque sans limite. Cette **dynamique** peut conduire à un **emballlement** des pondérations et, par ricochet, à l'émergence de **cycles** ou d'**oscillations**, surtout si d'autres liens liés à $\omega_{i,j}$ se trouvent également encouragés à croître.

Absence d’Inhibition et Instabilités

Lorsqu’on n’a **pas** incorporé de mécanismes d’inhibition compétitive (chap. 4.2.2.2) ou de saturation dans la mise à jour, plusieurs liens $\omega_{i,j}$ peuvent simultanément connaître cette croissance soutenue. On se retrouve alors avec un **blocage** : la dynamique part dans un renforcement collectif, où chaque lien élevé en entraîne un autre, modifiant la synergie d’autres paires, etc. Le réseau peut **basculer** successivement d’un groupement à un autre (un cluster domine, puis un autre s’impose), s’inscrivant dans un **cycle** de transitions ou de résonances.

Dans un **système** purement linéaire, la présence d’un **gain** trop important sur les boucles de rétroaction positive suscite des phénomènes de **résonance** ou de divergence. Dans un **système** non linéaire, on peut voir apparaître des attracteurs cycliques ou chaotiques, témoignant d’un équilibre impossible à stabiliser.

Exemple de Rétroaction Positive

Considérons un **mini-système** à deux pondérations ω_1 et ω_2 . On suppose :

$$\omega_1(t+1) = \omega_1(t) + \eta [S_1(\omega_2(t))\tau \omega_1(t)], \quad \omega_2(t+1) = \omega_2(t) + \eta [S_2(\omega_1(t))\tau \omega_2(t)].$$

Si $S_1(\omega_2)$ et $S_2(\omega_1)$ sont des **fonctions croissantes** (par exemple, $S_1(\omega_2) = a \omega_2$ avec a grand), alors la croissance de ω_2 **stimule** ω_1 , et la croissance de ω_1 **stimule** ω_2 . Sans **inhibition**, ce couplage conduit aisément à un **cycle** (un lien monte pendant que l’autre descend, puis vice versa) ou à une **explosion** de leurs valeurs. Dans d’autres cas, on obtient des oscillations autour d’un point ou d’une courbe, et ces oscillations peuvent ne jamais s’amortir si τ est trop faible ou si $\eta S_1, S_2$ sont trop grands.

Ce **phénomène** explique l’**emballlement** de certaines pondérations dans le réseau si rien n’est là pour les contenir (inhibition, saturation, etc.). Une fois enclenché, la rétroaction positive se nourrit d’elle-même et empêche le système de revenir vers un état stable.

Analyse Mathématique d’Exemple Simplifié

Prenons deux pondérations ω_1, ω_2 avec mise à jour :

$$\begin{cases} \omega_1(t+1) = \omega_1(t) + \eta [A \omega_2(t) - \tau \omega_1(t)], \\ \omega_2(t+1) = \omega_2(t) + \eta [B \omega_1(t) - \tau \omega_2(t)]. \end{cases}$$

Ici, $A, B > 0$ modélisent la rétroaction positive, et $\tau \omega_i(t)$ la décroissance. Les points fixes ω_1^*, ω_2^* vérifient :

$$\begin{cases} A \omega_2^* - \tau \omega_1^* = 0, \\ B \omega_1^* - \tau \omega_2^* = 0, \end{cases}$$

d’où $\omega_1^*(A B - \tau) = 0$. On voit qu’il existe un **point fixe non trivial** seulement si $A B = \tau$. La stabilité dépendra de la jacobienne :

$$J = \begin{pmatrix} 1 - \eta \tau & \eta A \\ \eta B & 1 - \eta \tau \end{pmatrix}.$$

Si les **valeurs propres** de J sortent du cercle unité, ω_1, ω_2 peuvent **osciller** ou diverger, engendrant cycles ou chaos.

Dans un **SCN** à grande dimension, ce mécanisme s'étend et des **oscillations** complexes ou pseudo-chaotiques peuvent s'installer si la rétroaction positive est prépondérante ou si aucune forme d'inhibition globale ne vient canaliser la croissance.

Conclusion Partielle : Nécessité d'un Frein ou d'une Inhibition

Lorsque la **rétroaction positive** est **excessive**, le SCN risque de développer des **oscillations** ou un **emballlement** où certains liens se rehaussent indéfiniment, souvent en boucle avec d'autres liens croissants. Pour nombre d'applications, c'est un **problème** (on désire la stabilisation de clusters), d'où l'introduction d'un **terme** d'inhibition compétitive (chap. 4.2.2.2) ou de **saturation** (limiter $\omega_{i,j} \leq \omega_{\max}$).

À l'inverse, dans certaines approches exploratoires, on peut **vouloir** conserver un degré de mouvement oscillatoire afin d'explorer diverses partitions temporaires. Mais la plupart du temps, si le but est d'obtenir un **arrangement stable**, il est crucial de **contenir** la rétroaction positive (régler η, τ de sorte que $\eta \tau \ll 1$) ou d'implémenter un mécanisme d'inhibition. Dans la section 4.4.2, on exposera des stratégies pour **endiguer** ces oscillations et viser la **stabilité** désirée.

4.4.1.2. Couplages non linéaires (ex. synergie dépendant des ω elles-mêmes)

Lorsqu'on conçoit un **DSL** (Deep Synergy Learning) de base, on postule souvent que la **synergie** $S(i,j)$ entre deux entités i et j découle de leurs **représentations** internes (\mathbf{x}_i pour le sub-symbolique, R_i pour le symbolique, etc.) sans dépendre de la **valeur courante** du lien $\omega_{i,j}$. Toutefois, dès qu'on autorise un **coupillage** où $S(i,j)$ devient **fonction** (directe ou indirecte) de la configuration $\{\omega_{p,q}\}$, on introduit une boucle de **rétroaction** supplémentaire. Cela peut, d'une part, **enrichir** la capacité d'auto-organisation et, d'autre part, **augmenter** le risque d'oscillations ou de comportements pseudo-chaotiques.

A. *Principe : $S(i,j)$ dépend de ω*

Dans le schéma de mise à jour standard,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

on considère $S(i,j)$ comme **exogène**, c'est-à-dire qu'elle ne varie pas selon la valeur de $\omega_{i,j}$. Mais si, au contraire, $S(i,j)$ dépend de $\boldsymbol{\omega}(t)$, alors on obtient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j, \boldsymbol{\omega}(t)) - \tau \omega_{i,j}(t)].$$

La **dépendance** $S(i,j, \boldsymbol{\omega}(t))$ peut être **directe** (S explicitement lié à $\omega_{i,j}$) ou **indirecte** (ex. un module symbolique, activé seulement si certains liens sont déjà au-dessus d'un seuil, modifie la synergie).

On peut imaginer une **synergie** de la forme :

$$S(i,j) = S_0(i,j) + \alpha \sum_{k \neq i,j} \omega_{i,k} \omega_{k,j}.$$

Cela signifie que plus un *tiers* k connecte déjà i et j , plus la “coopération” potentielle entre i et j grandit. C’est un **effet** d’“auto-renforcement” de type : “Si i et k sont liés, et k et j sont liés, alors la synergie i,j augmente.”

Mathématiquement, le **DSL** prend alors la forme d’un système **non linéaire** :

$$\omega(t+1) = \omega(t) + \eta [S(\omega(t)) - \tau \omega(t)].$$

La **variable** ω agit sur $S(\cdot)$, qui *en retour* modifie ω . Sur le plan **dynamique**, cette boucle interne peut créer des régimes oscillatoires ou amplifiés si rien ne vient limiter cette rétroaction.

B. Conséquences Mathématiques : Non-linéarités et Risques d’Emballlement

Dans un **DSL** où un **lien** $\omega_{i,j}$ peut *lui-même* accroître $S(i,j)$, la “correction” $\Delta\omega_{i,j} \approx \eta [S(i,j) - \tau \omega_{i,j}]$ risque de devenir **positive** au-delà d’un simple seuil, augmentant **exponentiellement** $\omega_{i,j}$. Sans un mécanisme de **saturation** ou d’**inhibition** (voir 4.4.2), $\omega_{i,j}$ peut monter sans bound ou osciller de façon importante.

Prenons un cas simplifié où :

$$S(i,j, \omega) = S_0(i,j) + \alpha \omega_{i,j}.$$

Alors la mise à jour additive devient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_0(i,j) + \alpha \omega_{i,j}(t) - \tau \omega_{i,j}(t)].$$

On obtient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta S_0(i,j) + \eta (\alpha - \tau) \omega_{i,j}(t).$$

- Si $\alpha > \tau$, alors $\alpha - \tau > 0$, et l’on peut avoir $\eta (\alpha - \tau) > 1$, créant un **terme d’amplification** qui fait diverger $\omega_{i,j}$.
- À l’inverse, si $\eta (\tau - \alpha) > 1$, des oscillations ou un comportement erratique peut se produire selon le signe et la dynamique pas-à-pas.

Ainsi, de **petits** changements de paramétrage peuvent entraîner de grands effets.

Dans un **réseau** de grande dimension, des couplages non linéaires peuvent conduire à des **dynamiques** pseudo-chaotiques, dans lesquelles un **petit** écart initial sur $\omega(0)$ se traduit, au bout de plusieurs itérations, par de grandes différences (sensibilité exponentielle aux conditions initiales). Sur un plan théorique, ce type de comportement exige de solides **dispositifs** (inhibition, limites de croissance, etc.) pour canaliser la rétroaction positive et préserver une trajectoire “stable” ou quasi-stable.

C. Exemples d'Applications

Même si les couplages non linéaires augmentent le risque de comportements complexes, ils **intéressent** certains scénarios :

Réseaux Sociaux ou Viraux

Un lien “viral” entre i et j renforce la probabilité que d’autres liens autour s’activent, ce qui, en retour, augmente la “popularité” de $\omega_{i,j}$.

Sur un plan **DSL**, cela se traduit par $S(i,j,\omega)$ croissant avec $\omega_{i,j}$. Sans mécanismes d’inhibition, on peut voir l'**emballlement** d’un cluster “viralisé.”

Logique Symbolique Contexte-Dépendante

Certains **règles** ne s’appliquent que si un lien existant $\omega_{i,j}$ dépasse un certain **seuil**. Cela modifie **localement** la valeur de $S(i,j)$ quand la configuration $\{\omega_{k,\ell}\}$ franchit ce seuil.

Sur le plan **dynamique**, on crée des **transitions** d’état quand un lien se met à excéder ledit seuil, ce qui peut enclencher une série de modifications en cascade.

Collaboration Multi-Agent

Dans un essaim robotique, la “**coopération**” entre deux robots peut se trouver *amplifiée* si un troisième robot opère de manière synergique avec chacun d’eux. Cela se modélise par un **terme** $\sum_k \omega_{i,k} \omega_{k,j}$ venant augmenter $S(i,j)$.

Risque : **boucles** de renforcement où le trio $\{i,j,k\}$ devient instable ou quasi-oscillatoire.

D. Comment Contrôler ou Réguler ces Couplages ?

Pour éviter des **oscillations** ou une **explosion** des liens, on recourt souvent à :

- **Inhibition** (chap. 4.2.2.2) : Un coût $-\gamma \sum_k \omega_{i,k}$ empêche qu’un nœud i accroisse tous ses liens en même temps.
- **Saturation** ou “clipping” : On borne $\omega_{i,j} \leq \omega_{\max}$.
- **Seuil** adaptatif : On coupe toute $\omega_{i,j} < \theta$.

Ces **outils** limitent la rétroaction positive et bloquent la dérive exponentielle.

On peut analyser la **stabilité** via une étude du **système** dynamique :

$$\omega_{i,j}(t+1) - \omega_{i,j}(t) = \eta [S(i,j, \omega(t)) - \tau \omega_{i,j}(t)].$$

En dérivant les **conditions** sous lesquelles $\nabla_\omega S(\cdot)$ est dominée par le terme $-\tau \mathbf{I}$, on obtient des **critères** (genre $\eta \parallel \nabla S \parallel < \tau$) assurant un équilibre stable.

Dans certains cas, on *veut* cette rétroaction positive pour encourager un **cluster** à s’auto-former vigoureusement (ex. pour modéliser des “communautés fortes”). On l’associe à un frein global

(inhibition, competition). Le résultat : l'émergence de **petits** clusters internes très soudés, plutôt que de liens dispersés.

E. Conclusion

Dès lors que la **synergie** $S(i, j)$ dépend de $\omega_{i,j}$ — ou plus globalement, de la configuration ω — on bascule dans un **système** dynamique potentiellement **non linéaire** à rétroaction :

Enrichissement : la **cohérence contextuelle** se modèle plus finement (ex. plus un cluster grandit, plus sa synergie interne grandit).

Risque : cette rétroaction positive peut engendrer **oscillations, pseudo-chaos, ou instabilité**.

Pour y faire face, on introduit des **régulations** (inhibition, saturation, etc.) et on contrôle les **paramètres** de la mise à jour (taux η , coefficient τ) afin de **contenir** la rétroaction. Ainsi, on confère au DSL la possibilité d'exprimer des phénomènes émergents plus riches, tout en préservant une **stabilité** ou une **convergence** raisonnable.

4.4.2. Stratégies pour Rompre les Oscillations

Dans la section 4.4.1, nous avons décrit **pourquoi** des oscillations ou un pseudo-chaos pouvaient apparaître dans un **SCN** (Synergistic Connection Network) : rétroactions positives trop intenses, absence d'inhibition, couplages non linéaires, etc. Pour de nombreuses applications du **DSL** (Deep Synergy Learning), on souhaite cependant **stabiliser** la dynamique, car des oscillations prolongées ou un chaos déterministe peuvent rendre la structure (clusters, attracteurs) peu exploitable.

Heureusement, il existe plusieurs **méthodes** pour **rompre** ou **atténuer** ces oscillations, en introduisant notamment :

- De l'**inhibition latérale** (4.4.2.1),
- Des **règles de sparsification** (4.4.2.2),
- Une **adaptation** fine des paramètres η, τ (4.4.2.3).

Ces stratégies visent à **contenir** l'emballage de certains liens et à maintenir la rétroaction dans un domaine maîtrisé. Elles s'appliquent que la mise à jour soit **additive** ou **multiplicative**, et peuvent parfois se cumuler pour un meilleur effet stabilisateur.

4.4.2.1. Inhibition Latérale (2.4.4) : contrainte de ressources, pousse le réseau à choisir un nombre limité de liens forts

L'**inhibition latérale** – parfois appelée **inhibition compétitive** – vise à contrôler la **croissance simultanée** de multiples liens sortant d'une même entité \mathcal{E}_i . Au lieu de laisser $\omega_{i,j}$ s'élever librement lorsque la synergie $S(i, j)$ est satisfaisante, on y ajoute un **terme négatif** qui dépend de la **somme** (ou d'un autre agrégat) des connexions $\omega_{i,k}$ sortant du même noeud i . Une **forme** typique de cette mise à jour a été discutée en (4.2.2.2) et peut se formaliser, pour une règle additive, de la façon suivante :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

où $\gamma > 0$ est un **coefficent** d'inhibition, et $\sum_{k \neq j} \omega_{i,k}(t)$ désigne la **masse** totale des autres liens sortants de \mathcal{E}_i .

A. Principe et influence sur les oscillations

Lorsque \mathcal{E}_i dispose d'un **budget** de ressources limité (biologiquement, on parle d'énergie neuronale, dans un réseau technique de bande passante, etc.), l'inhibition latérale impose une **compétition interne** : si $\omega_{i,j}$ tente de s'élèver, l'effet $-\gamma \sum_{k \neq j} \omega_{i,k}$ contraint la croissance simultanée d'autres liens. Sur le plan **comportemental**, cela force \mathcal{E}_i à "choisir" un petit nombre de connexions privilégiées, évitant qu'il ne se connecte trop fortement à de multiples nœuds en parallèle.

Dans le DSL sans inhibition, un **lien** $\omega_{i,j}$ peut parfois s'amplifier fortement dès que la synergie $S(i,j)$ est supérieure à la "penalty" $\tau \omega_{i,j}$, en particulier s'il existe des boucles positives. L'inhibition latérale introduit un **terme négatif** supplémentaire proportionnel à la somme $\sum_{k \neq j} \omega_{i,k}$, ce qui limite la capacité de $\omega_{i,j}$ à croître de manière exponentielle. Ce **frein** empêche également un ensemble de liens de s'amplifier **simultanément**, l'un d'eux finissant par "dominer" les autres.

De ce fait, toute **oscillation** qui apparaît parce que plusieurs liens tenteraient d'atteindre de grandes valeurs à tour de rôle se trouve en partie **amortie** par la compétition : plus un lien s'élève, plus il pèse sur l'ensemble des autres et inversement. On observe ainsi, mathématiquement, une **dissipation** de l'énergie potentielle d'oscillation.

B. Analyse mathématique simplifiée

Pour voir comment l'inhibition latérale agit sur la **stabilité**, considérons la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Un **point fixe** $\omega_{i,j}^*$ s'obtient lorsque $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$, i.e. :

$$S(i,j) - \tau \omega_{i,j}^* - \gamma \sum_{k \neq j} \omega_{i,k}^* = 0.$$

Sur l'ensemble des entités, cela se traduit par un système d'équations linéairement couplées (ou, selon la forme de S , non linéaires), imposant une **compétition** pour qu'un nœud \mathcal{E}_i ne puisse "allouer" trop de force à plusieurs liens concurrents.

Sur le plan **stabilité**, la matrice jacobienne \mathbf{J} de la transformation $\omega_{i,j} \mapsto \omega_{i,j} + \dots$ inclut des **termes croisés** liés au $\sum_{k \neq j} \omega_{i,k}$. En un point fixe, ces dérivées croisées se traduisent par un **effet** de concurrence : quand un lien se renforce, il pèse sur les autres, ce qui rend plus difficile un emballage synchronisé. On diminue ainsi la probabilité de boucles positives incontrôlées, et,

dans la majorité des cas pratiques, on aboutit à un **régime** stable ou faiblement oscillatoire (des oscillations amorties).

C. Impact sur la formation de clusters

L'inhibition latérale incite chaque entité \mathcal{E}_i à ne conserver qu'un **nombre restreint** de liaisons fortes. Cela se rapproche d'un **modèle** "winner-takes-most" : si $\omega_{i,j}$ est très élevé pour un certain j , les autres $\omega_{i,k}$ ($k \neq j$) subissent une pénalisation accrue. On obtient donc un **réseau économe** en connexions, renforçant la **lisibilité** des structures émergentes (éviter que tout soit moyennement connecté).

Dans les algorithmes de clustering, on recherche souvent des **groupes** ("communautés") clairement définis : des entités fortement reliées **entre elles** et faiblement reliées **aux autres**. L'inhibition latérale contribue à ce "clivage" : un nœud \mathcal{E}_i ayant rejoint un cluster principal est moins susceptible de développer des liens concurrentiels vers un autre cluster, car cette croissance nouvelle se heurterait au coût $\gamma \sum_{k \neq j} \omega_{i,k}$. Cela **stabilise** les partitions émergentes.

D. Conclusion : un mécanisme crucial de stabilité

L'**inhibition latérale**, inspirée des phénomènes **biologiques** (2.4.4), remplit deux rôles essentiels :

Rôle de stabilisation :

En limitant l'**auto-amplification** de multiples liens, on **réduit** le risque d'oscillations ou de comportements chaotiques. La dynamique DSL reste **contrôlée** et converge souvent plus sûrement.

Rôle de structuration :

Chaque nœud se **focalise** sur quelques liens dominants, produisant des **clusters** plus nets et plus faciles à interpréter. La compétition incite le réseau à "faire des choix" tranchés, plutôt qu'à disperser ses connexions.

Dans les sections suivantes (4.4.2.2, 4.4.2.3) et plus loin (chap. 7.4), on verra comment compléter cette inhibition latérale par d'autres techniques (seuil adaptatif, saturation, recuit simulé) pour **approfondir** la maîtrise de la dynamique et **éviter** les fluctuations excessives. L'inhibition demeure toutefois l'un des **leviers** centraux pour maintenir un **SCN** (Synergistic Connection Network) dans un état **auto-organisé**, cohérent et **parcimonieux**.

4.4.2.2. Sparsification : imposer un maximum de k liaisons par entité (k plus forts liens), ε -radius, etc.

La **sparsification** vise à **réduire** le nombre de liaisons $\omega_{i,j}$ actives dans un **SCN** (Synergistic Connection Network), afin de **limiter** la complexité des rétroactions susceptibles de créer des oscillations ou d'autres comportements chaotiques. L'idée est qu'en empêchant chaque nœud \mathcal{E}_i d'entretenir de multiples liens à intensité moyenne, on obtient un **réseau** plus clairsemé (sparse),

ce qui **stabilise** la dynamique et **clarifie** la formation de clusters. Les méthodes les plus courantes sont :

k-liaison : on ne garde, pour chaque entité i , que les k plus forts liens $\omega_{i,j}$.

ε -seuil : on annule tout lien $\omega_{i,j}$ dont la valeur tombe sous un seuil ε .

Dans un cas, on contrôle le **degré** sortant du nœud i (exactement k liens), dans l'autre, on impose un **niveau** minimal pour conserver une liaison.

A. Principe général de la sparsification

En l'absence de **restriction**, un nœud \mathcal{E}_i pourrait entretenir des connexions non négligeables vers la quasi-totalité des autres noeuds. Dans les réseaux de grande dimension, cela crée un très grand nombre de **circuits de rétroaction** : le surplus de liens “moyens” accroît la **probabilité** d'oscillations ou de comportements complexes.

La **sparsification** rompt cette possibilité : elle force chaque nœud à “**faire des choix**”, c'est-à-dire à ne conserver qu'un **sous-ensemble** restreint de connexions jugées essentielles. En conséquence, de nombreux liens, dont les valeurs sont trop faibles ou moins utiles, sont **coupés** (fixés à 0). On obtient alors un **réseau** moins densément connecté, limitant mécaniquement la complexité dynamique et réduisant le risque d'oscillation.

B. Méthodes pratiques

Après chaque cycle de mise à jour $\omega_{i,j}(t+1)$..., on “trie” les liaisons sortantes de chaque entité \mathcal{E}_i par ordre décroissant (ou selon un critère de pertinence). Seuls les k premiers liens $\omega_{i,j}$ (ceux de plus grande intensité) sont **préservés**. Les autres, classés au-delà du rang k , sont mis à 0.

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } \omega_{i,j} \text{ est dans le top } k, \\ 0, & \text{sinon.} \end{cases}$$

Cette **stratégie** garantit que le **degré** (sortant) de \mathcal{E}_i ne dépasse pas k . On parle parfois de “k-sparsification”. Cette forme de contrainte a pour effet de **réduire** fortement le nombre de liens actifs si $k \ll n$. Cela **simplifie** la dynamique : chaque nœud ne s'occupe que d'un petit set de voisins jugés les plus forts.

Ici, on fixe un $\varepsilon > 0$. Après la mise à jour, si $\omega_{i,j} < \varepsilon$, alors on **coupe** le lien $\omega_{i,j}$ en le mettant à 0 :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } \omega_{i,j}(t+1) \geq \varepsilon, \\ 0, & \text{sinon.} \end{cases}$$

On obtient un “ ε -radius” : seuls les liens **au-dessus** de ε subsistent. Le **degré** de chaque nœud n'est pas contrôlé directement (cela peut fluctuer), mais on coupe efficacement tout petit lien demeuré trop faible, évitant la prolifération de connexions moyennes. Cette technique se rapproche du “hard thresholding” (voir 7.4.3) : on impose qu'en deçà d'un certain seuil, le lien n'a plus de raison d'exister.

On peut :

- **Appliquer** la coupe (k -liaison ou ε -seuil) à chaque itération (version stricte),
- Ou **périodiquement** (tous les p cycles, ou quand la densité du réseau dépasse un certain seuil).

De plus, on peut **faire varier** k ou ε au fil du temps : autoriser d'abord un plus grand nombre de liens, puis "resserrer" la contrainte. Cela s'apparente à un recuit "inhibiteur", où la parcimonie augmente à mesure que le réseau s'organise.

C. Impact sur la limitation des oscillations

Lorsqu'un **grand** nombre de liens coexistent dans un SCN, la **rétroaction** non linéaire peut entraîner des **oscillations** localisées ou globales (voir 4.3.2). En imposant un k -liaison ou un ε -seuil, on **casse** de nombreux arcs, réduisant ainsi drastiquement les chemins de rétroaction possibles. Moins de boucles = moins d'opportunités d'osciller.

Par ailleurs, chaque nœud n'a plus la "capacité" de faire croître simultanément un grand nombre de liens, car la plupart seront coupés s'ils sont trop faibles. La **dynamique** se concentre alors sur un ensemble restreint d'arcs — souvent, ceci rend la **trajectoire** du réseau plus lisible et moins sujette à des allers-retours.

D. Exemples d'usage et bénéfices

- **Clustering** plus franc : avec peu de liaisons conservées, les **clusters** deviennent plus évidents, car on voit nettement qui se connecte à qui.
- **Calcul plus rapide** : dans un réseau volumineux (n grand), manipuler une matrice ω dense peut être coûteux ($O(n^2)$ liens). Si, pour chaque i , on ne garde que k liens, on tombe à $O(nk)$, fréquemment $O(n)$ si k est constant. Cela accélère la mise à jour, en plus de calmer la dynamique.
- **Contrôle** de la densité : si le réseau tend à trop se connecter (souvent constaté dans l'entraînement DSL), la sparsification agit comme un "filet" coupant la majorité des petites connexions.

E. Conclusion : un atout majeur pour calmer le SCN

La **sparsification** se présente comme un **outil** essentiel, complémentaire à l'**inhibition latérale** (4.4.2.1) pour :

- **Réduire** le risque d'oscillations ou de comportements chaotiques,
- **Structurer** un SCN en clusters lisibles (éviter la dispersion de liaisons moyennes),
- **Abaïsser** le coût computationnel, surtout pour de grandes n .

Dans un **DSL**, imposer "k-liaison", " ε -seuil" ou toute autre forme de coupe (périodique ou continue) permet, au besoin, d'assurer la **stabilité** de la dynamique. Les sections suivantes (4.4.2.3, etc.) détailleront également la façon d'ajuster η et τ pour achever la **réduction** des phénomènes oscillatoires et améliorer la convergence vers un **arrangement** stable du SCN.

4.4.2.3. Adaptation de η, τ : régulation fine de la vitesse de mise à jour et de la décroissance.

La **dynamique** d'un SCN (Synergistic Connection Network) repose en grande partie sur deux **paramètres** fondamentaux : le **taux d'apprentissage** η et le **taux de décroissance** τ . Ces paramètres apparaissent dans l'équation de mise à jour (chap. 4.2) :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

ou, dans une version multiplicative, sous une forme proche de

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \times [1 + \eta(\dots)] - \dots .$$

D'un point de vue **mathématique**, η contrôle la **vitesse** à laquelle chaque lien $\omega_{i,j}$ se met à jour ; τ détermine la **force** de "rappel vers zéro", autrement dit la **décroissance** qui empêche le lien de grandir sans limite. Un mauvais choix de η ou de τ peut générer des **oscillations**, voire un **emballlement** ; à l'inverse, des valeurs trop "timides" peuvent **freiner** exagérément la formation des clusters.

Dans les sections précédentes (4.4.2.1, 4.4.2.2), nous avons décrit l'**inhibition latérale** et la **sparsification** comme des solutions pour **endiguer** les oscillations. Ici, nous montrons qu'un **règlage** précis de η et τ est un levier tout aussi crucial, souvent complémentaire à ces mécanismes.

A. Rôle de η : vitesse de mise à jour

Dans le cas **additif** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

l'analyse (4.2.1.2) révèle que pour garantir une convergence locale (sans oscillations), il faut

$$|1 - \eta \tau| < 1,$$

ou au moins $\eta \tau < 2$. Au-delà de cette borne, le système peut entrer en **oscillation** et, pour $\eta \tau$ encore plus grand, **divergir**.

Lorsque le SCN comprend **plusieurs** entités et couplages, l'analyse de stabilité se raffine par l'étude de la **matrice jacobienne** globale. Néanmoins, l'idée essentielle subsiste : un η trop fort accélère énormément la correction, risquant de "sauter" par-dessus la valeur d'équilibre, créant des "rebonds" successifs (oscillations). En revanche, un η trop faible retarde la mise en place des **clusters**, rendant la convergence (ou la pseudo-convergence) plus lente.

À l'instar de nombreuses méthodes d'apprentissage, on peut **décroître** η au fil des itérations (un "annealing" du learning rate). On commence avec un η suffisamment élevé pour **differencier** rapidement les liens (ceux qui ont une synergie élevée se renforcent plus vite), puis on réduit η afin de **stabiliser** la configuration et éviter que les fluctuations ne perdurent.

B. Rôle de τ : force de la décroissance

Le paramètre τ agit comme un **frein** : à chaque mise à jour,

$$\eta [S(i,j) - \tau \omega_{i,j}(t)]$$

inclus la composante $-\eta \tau \omega_{i,j}(t)$, qui ramène $\omega_{i,j}$ vers zéro si la synergie $S(i,j)$ n'est pas assez grande. Si τ est trop **faible**, les liens risquent de croître exagérément ; si τ est trop **fort**, la plupart des liens restent proches de zéro, le SCN échouant à **exploiter** la synergie potentielle.

- $\tau \approx 0.1$: peut être trop faible si, par exemple, $S(i,j) \approx 1$. Dans ce cas, la décroissance (multipliée par $\omega_{i,j}$) n'est pas assez forte pour empêcher un emballement (surtout si η n'est pas minime).
- $\tau \approx 10$: écrase presque toute **croissance** des liens, car $\tau \omega_{i,j}(t)$ dépasse rapidement $S(i,j)$. Les liaisons finissent par stagner très près de zéro.

Comme pour η , on peut **faire évoluer** τ : commencer avec un τ modéré (laissant le réseau former de premiers clusters), puis accroître τ pour consolider la configuration en bloquant les variations tardives. Dans un SCN évolutif, un module de surveillance détecte d'éventuelles oscillations : si elles sont trop fortes, on **relève** τ pour "amortir" davantage la dynamique.

C. Lien avec l'inhibition et la sparsification

Ces mécanismes se **superposent**. Avec **inhibition latérale** (4.4.2.1), la mise à jour inclut un terme

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

qui limite aussi la croissance simultanée de plusieurs liens. Cela peut autoriser un η un peu plus élevé ou un τ plus bas, puisque la compétition prend en charge une partie de la **stabilisation**.

Si l'on **coupe** régulièrement les liens trop faibles (ε -seuil) ou ne garde que les k plus gros liens sortants, la dimension effective du système diminue (moins de boucles de rétroaction). Par conséquent, le couple (η, τ) peut être choisi un peu plus librement. Dans un réseau très dense, il faut souvent "serrer" η et τ de manière prudente.

D. Approche algorithmique pour η, τ : essais, annealing, jacobienne

Essais empiriques : La méthode la plus fréquente : on **tente** plusieurs paires (η, τ) , on **observe** le SCN. S'il y a trop d'oscillations, on réduit η ou on monte τ . Si, au contraire, la formation de clusters est trop lente ou partielle, on augmente η ou on diminue τ . Pour des **grands** SCN, on pratique souvent un calibrage sur un **sous-problème** réduit.

Annealing : On **démarre** avec un η modérément élevé pour accélérer la **différenciation** des liaisons, puis on le **baisse** de façon linéaire ou exponentielle quand le réseau commence à se structurer. On peut également monter τ progressivement pour fortifier l'effet de décroissance au fur et à mesure que la configuration se stabilise.

Étude de la jacobienne : Sur un système jouet (quelques liens), on peut analyser la **matrice J** = $(\partial \Delta \omega / \partial \omega)$. On vise $\|J\| < 1$ ou un spectre(J) en module < 1. Cela donne des **conditions** plus formelles sur η, τ . Dans des SCN réels (grande échelle), cette étude devient trop complexe, mais la simulation permet de **surveiller** l'évolution de ω et le maintien d'une certaine stabilité.

E. Conclusion

Le couple η, τ est un **pivot** de la dynamique DSL :

- η détermine la **vitesse** de réaction des liaisons $\omega_{i,j}$. Trop grand, on risque des oscillations ; trop petit, on s'enlise dans une convergence lente ou imparfaite.
- τ fixe la **décroissance** imposée à chaque lien, empêchant un emballlement illimité si elle est bien choisie, ou brisant la croissance si elle est excessive.

Combiné à l'**inhibition latérale** (4.4.2.1) et la **sparsification** (4.4.2.2), un réglage adapté de η et τ :

- **Amortit** les oscillations,
- **Canalise** la dynamique pour qu'elle converge (ou quasi-converge) vers un état de clusters stables,
- **Équilibre** la rapidité d'adaptation et la stabilité globale du SCN.

Cette **régulation** fine de η, τ complète donc les leviers décrits, assurant que la configuration du réseau ne tombe ni dans un chaos oscillatoire ni dans une inertie excessive. On obtient ainsi un SCN plus **souple** et **maîtrisé**, à la fois **réactif** et **stable** face aux variations de \mathbf{x}_i , de $S(i, j)$, ou d'autres paramètres internes.

4.4.3. Cas Stochastiques : Recuit, Perturbations

Après avoir présenté (4.4.2) diverses stratégies **déterministes** (inhibition, sparsification, ajustement η, τ) pour rompre ou limiter les oscillations, nous abordons à présent des **approches stochastiques**. L'idée est parfois **d'injecter du bruit** (des perturbations aléatoires) dans la dynamique, afin de sortir de configurations sous-optimales ou d'éviter une oscillation stable qui ne nous satisfait pas (chap. 2.4.5.1 abordait déjà ce concept). On retrouve ici l'analogie avec le **recuit simulé** en physique statistique, qui secoue la configuration $\omega(t)$ avant de "refroidir" pour se stabiliser dans un attracteur potentiellement plus global.

4.4.3.1. Injection volontaire de "bruit" pour échapper aux minima (2.4.5.1)

Il existe des situations où le **Synergistic Connection Network** (SCN) se trouve **piégé** dans un **minimum local** ou entretient une **oscillation** persistante qui empêche la dynamique de converger vers une configuration plus satisfaisante. Dans un tel contexte, l'idée d'introduire un **terme stochastique** dans la mise à jour des pondérations peut sembler contradictoire, d'autant plus que l'on cherche souvent à **diminuer l'instabilité** ou la **non-linéarité**. Cependant, cette démarche conserve une **logique** solide : de la même manière que l'on applique un **recuit** en métallurgie pour franchir une barrière d'énergie, l'**injection de bruit** agit ici comme un **apport d'énergie** aléatoire qui peut aider le SCN à se libérer d'un attracteur local ou à rompre un **cycle d'oscillation** (cf. section 2.4.5.1).

Une **contradiction** apparente peut se lire dans le fait que l'on introduise délibérément des **perturbations aléatoires**, alors même que des mécanismes tels que l'**inhibition** ou la **sparsification** visent à stabiliser la dynamique. Pour comprendre cette démarche, il faut garder à l'esprit que l'objectif premier est de **favoriser l'exploration** de configurations nouvelles, qui seraient difficilement atteignables depuis un état trop stable mais de mauvaise qualité. Les **systèmes physiques** fournissent une analogie : un solide en cours de recuit a besoin d'un **excès** d'énergie pour franchir le **seuil** séparant deux **vallées** de potentiel. Dans le cadre d'un **SCN**, le bruit joue un rôle semblable en **déstabilisant** ponctuellement la **trajectoire** des pondérations $\omega_{i,j}$. On peut alors espérer que la structure glisse vers un autre **bassin d'attraction**, plus proche d'une configuration optimale.

Il est fréquent, dans les **réseaux** adaptatifs et non linéaires, de se trouver en présence d'**oscillations** du type "ping-pong", où quelques liaisons alternent périodiquement entre des valeurs élevées et faibles, sans jamais converger vers un point fixe stable. L'**injection** d'un **terme aléatoire** peut dans ce cas **briser** la synchronisation de l'oscillation. Si l'on modélise ce phénomène, on peut considérer qu'à chaque itération t , la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t)$$

vient remplacer la formule usuelle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$. Le **terme** $\xi_{i,j}(t)$ est un **bruit** gaussien ou uniforme, en général de moyenne nulle, dont l'**amplitude** peut être fixée ou décroissante au fil du temps. Dans certains cas, on peut aussi cibler seulement un **sous-ensemble** de paires (i,j) pour ne pas perturber l'ensemble du réseau.

Sur le **plan mathématique**, on peut relier cette injection de bruit à la **descende de gradient** perturbée, très souvent employée pour **échapper** aux minima locaux. Bien que le **DSL** et son **SCN** ne correspondent pas toujours à une **descende de gradient** globale (on parle souvent d'une mise à jour locale ne minimisant pas explicitement une fonction ; voir chapitre 2.4.3 sur la notion d'**énergie**), le **raisonnement** demeure valable : tout **puits** local où le réseau serait "coincé" dispose d'une **barrière** à franchir, et le bruit aléatoire fournit la **quantité** d'énergie nécessaire pour la surmonter. Par analogie, on peut dire que l'**état** du **SCN** suit un "processus stochastique" à l'intérieur d'un **paysage d'énergie**, et qu'en l'absence d'un tel bruit, l'état risquerait de converger dans la première vallée rencontrée, sans avoir la possibilité d'explorer un autre **bassin** à énergie plus basse.

D'un point de vue **dynamique**, on peut observer que dans des conditions de bruit modéré, la **dérive** aléatoire permet d'**éviter** la cristallisation prématuée d'un **mauvais** attracteur local. Si le bruit est trop élevé, on peut toutefois provoquer un **comportement** erratique qui empêche de **stabiliser** réellement la structure. En pratique, on paramètre la **variance** de $\xi_{i,j}(t)$ ou sa **fenêtre** d'uniformité de façon à autoriser quelques **sauts** hors des vallées, mais sans plonger le réseau dans un mouvement totalement brownien.

En **conclusion**, la **méthode** stochastique qui consiste à **injecter** volontairement du **bruit** dans la dynamique $\omega_{i,j}$ répond à deux motivations : permettre au **SCN** de **sortir** d'un **attracteur** local (ou d'une partition sous-optimale) et **rompre** des **oscillations** cycliques. Elle trouve un **équilibre** en se combinant à d'autres approches comme l'**inhibition**, la **sparsification** ou la **compétition** entre liaisons, décrites dans le cadre du **DSL** (voir chapitre 2.4.5.1). Ainsi, même si l'ajout de perturbations aléatoires est a priori contraire à la recherche de **stabilité**, il apparaît en réalité comme

un **complément** essentiel à la dynamique, pour accroître la **capacité** d’exploration et améliorer le **résultat** final lorsque le paysage d’états regorge de minima locaux ou de cycles indésirables.

4.4.3.2. Recuit simulé : un protocole pour secouer la matrice ω avant de “refroidir” et se stabiliser

Le recours à un **recuit simulé** (`\emph{simulated annealing}`) permet de structurer plus finement l’**injection de bruit** au sein d’un **Synergistic Connection Network** (SCN). Plutôt que d’ajouter des perturbations stochastiques ponctuelles ou uniformément réparties dans le temps, le **recuit** propose de “chauffer” d’abord le réseau à une **température** élevée pour lui donner la capacité d’**explorer** un vaste éventail de configurations, puis de “refroidir” graduellement, laissant le SCN se **figer** dans un état final qu’on espère “globalement optimal” ou, à tout le moins, **meilleur** qu’un minimum local trivial. Cette approche puise ses racines dans la **physique statistique**, où l’on chauffe un matériau (agitation thermique forte) avant de le refroidir lentement, de sorte à franchir des barrières d’énergie et atteindre une structure cristalline stable.

A. Analogies et principes du recuit.

Le recuit s’inspire de la métallurgie : un matériau chauffé voit ses atomes disposer d’une énergie thermique suffisante pour franchir de nombreuses barrières locales et se réarranger. En le refroidissant graduellement, on stabilise la configuration dans une cristallisation souvent plus homogène et moins sujette aux défauts. Transposé à un SCN, le “chauffage” correspond à l’adjonction d’un **terme aléatoire** de grande amplitude à la formule de mise à jour des pondérations $\omega_{i,j}$. Ceci permet au réseau de “sauter” entre plusieurs bassins d’attraction, sans se bloquer prématurément dans l’un d’eux. Le “refroidissement” opère en réduisant progressivement ce **bruit**, faisant tendre la dynamique vers une structure où les liens $\omega_{i,j}$ se figent dans une configuration stable.

Dans la perspective d’une **fonction énergie** $J(\omega)$ (voir la section 2.4.3 sur la vision “descente d’énergie”), un important niveau de bruit favorise les remontées énergétiques transitoires, que l’on peut formaliser par un facteur $\exp[-(J(\omega') - J(\omega))/T]$. Le réglage d’une **température** T régule le degré de tolérance aux hausses d’énergie : à haute température, on accepte aisément de “monter” en énergie pour fuir un puits local médiocre ; à basse température, on rejette de tels mouvements, ce qui stabilise le SCN.

B. Protocole concret dans un SCN.

Dans une formulation pratique, on pose une **température** T dont la valeur décroît au fil des itérations selon une loi typique telle que

$$T(t) = T_0 \alpha^{\lfloor t/\Delta t \rfloor}, \quad 0 < \alpha < 1,$$

avec un pas de temps Δt . On ajoute alors, à chaque mise à jour $\omega_{i,j}(t+1)$, un **bruit** $\xi_{i,j}(t)$ dont la **variance** dépend de T . Ainsi, on obtient la relation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma(t)^2)$ avec $\sigma(t) \propto T(t)$. Dans les premières itérations, $\sigma(t)$ est grande : le SCN est agité, il “saute” volontiers hors des puits locaux. Puis, à mesure que T baisse, $\sigma(t)$ diminue, ce qui fige progressivement la structure émergente.

C. Analyse mathématique du recuit.

Sur le plan formel, l’on connaît des **résultats** garantissant qu’un refroidissement suffisamment lent (c’est-à-dire un α très proche de 1) peut conduire le système à **converger** vers un minimum global de l’énergie \mathcal{J} . En pratique, on arbitre entre la qualité de la solution (un refroidissement plus lent favorise l’exploration) et le temps de calcul (un refroidissement trop lent peut être inabordable). Dans un **SCN**, on n’a pas toujours besoin de suivre l’exact protocole de la physique statistique (Markov Chain Monte Carlo, acceptation/rejet), car un simple **bruit additif** “pondéré” par $\exp[-\Delta\mathcal{J}/T]$ peut remplir la fonction d’exploration.

En tout état de cause, le recuit simulé fournit un **cadre** méthodique pour **monter** d’abord la “température”, explorer le **paysage** d’états plus librement, puis “refroidir” en limitant graduellement les perturbations.

D. Avantages et inconvénients.

Le recuit simulé présente l'**avantage** majeur de décoller le réseau d’un minimum local (ou d’un attracteur) inopportun. Là où un DSL déterministe, limité à

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta [S(i,j) - \tau \omega_{i,j}(t)],$$

risquerait de s’arrêter prématurément, la “secousse” apportée par un niveau élevé de bruit autorise l’exploration d’autres **partitions** ou **clusters** potentiels. Cette dimension d’exploration peut aussi, dans les phases “haute température”, dévoiler plusieurs organisations concurrentes, dont l’une s’avérera plus globalement satisfaisante.

Ce **bénéfice** se paie par un **coût** temporel accru. D’une part, on doit maintenir des itérations “bruyantes” en nombre suffisant pour vraiment sauter entre bassins d’attraction ; d’autre part, il faut prévoir un **refroidissement** progressif, ce qui rallonge la durée de la simulation. Un autre **risque** vient de la configuration initiale du $\sigma(t)$: si la température de départ est trop basse, on ne profitera pas assez du caractère stochastique ; si elle est trop haute, la trajectoire pourra demeurer chaotique ou écarter certaines bonnes configurations faute d’un contrôle approprié.

E. Conclusion.

Le recuit simulé se révèle un protocole pertinent pour “secouer” la matrice $\{\omega_{i,j}\}$ avant de refroidir la dynamique, cherchant ainsi à éviter les minima locaux et à **brisier** des oscillations éventuelles. Il complète les approches plus déterministes (inhibition, saturation) qui visent à **stabiliser** la structure. En veillant à la cohérence du rythme de décroissance de la température, on apporte au **Synergistic Connection Network** un instrument supplémentaire pour explorer l’espace des configurations, déjouer les verrous locaux et produire, in fine, une **organisation** potentiellement plus satisfaisante à l’échelle du réseau.

4.4.3.3. Choix de la “température” stochastique, liens avec la physique statistique

Lorsqu'un **recuit simulé** ou plus généralement une **injection de bruit** (voir 4.4.3.1 et 4.4.3.2) est implémenté dans un **Synergistic Connection Network** (SCN), la notion de **température** T joue un rôle central. Cette **température**, inspirée des concepts de la **physique statistique**, contrôle l'**ampleur** des perturbations aléatoires ajoutées aux pondérations $\omega_{i,j}$. Un bon réglage de T est donc déterminant pour que le réseau puisse, dans un premier temps, **explorer** suffisamment l'espace des configurations, puis, à l'inverse, se **figer** à la fin du processus, permettant ainsi une convergence vers un état stable.

A. Rappel : analogie avec la physique statistique.

En physique, la **température** T mesure l'**agitation** des atomes ou des particules. Un solide porté à haute température dispose de suffisamment d'**énergie** pour dépasser des barrières locales et se réorganiser librement. Dans le **SCN**, la “température” joue un rôle similaire : lorsqu'elle est élevée, on autorise l'**injection** de perturbations stochastiques de grande amplitude dans la mise à jour des pondérations. Ces perturbations permettent au réseau de “sauter” hors de puits d'énergie trop étroits ou de configurations peu satisfaisantes. À mesure qu'on réduit progressivement T , l'agitation baisse et le SCN se **stabilise**, à l'image d'un refroidissement qui fige les atomes d'un métal dans une structure cristallisée.

Il existe, dans les modèles inspirés de la physique statistique (Ising, Hopfield, etc.), une relation de type $\exp[-(\mathcal{J}(\omega') - \mathcal{J}(\omega))/T]$ régissant la probabilité d'accepter une modification augmentant l'énergie du système. Même si un **DSL** peut s'écartier de ces protocoles Markoviens classiques, la **logique** reste identique : un niveau de température suffisamment haut donne au SCN la capacité de franchir de petites “vallées” d'énergie et d'explorer son espace d'états de manière étendue.

B. Comment définir T et l'amplitude de bruit.

Le principe consiste à associer l'**injection** de bruit à la **température**. Concrètement, si $\xi_{i,j}(t)$ désigne un bruit gaussien de moyenne nulle, sa variance σ^2 peut être reliée à la température via $\sigma = \sigma_0 T$. Lorsque T est grand, les fluctuations $\xi_{i,j}(t)$ atteignent une amplitude notable ; à l'inverse, pour une température faible, elles deviennent minimes. Dans un **recuit simulé**, la température elle-même suit un schéma de décroissance contrôlé, par exemple

$$T(t) = T_0 \alpha^{\lfloor t/4 \rfloor},$$

avec $\alpha \in (0,1)$. Ainsi, au démarrage, T_0 est suffisamment élevé pour permettre une exploration étendue, puis, au fil du temps, on abaisse σ et on réduit graduellement le mouvement aléatoire autour de la configuration courante.

Cette stratégie reproduit l'idée que la **dynamique** d'un SCN doit d'abord être **libérée** (phase de haute température) pour échapper à des attracteurs locaux trop précoces, puis **ralentie** (phase de basse température) afin de stabiliser la structure. Dans certains contextes, on maintient même un petit bruit résiduel pour conserver un léger “tremblement” qui peut se révéler bénéfique, par exemple pour empêcher de retomber dans des oscillations cycliques.

C. Sélection de T_0 et vitesse de décroissance.

Le choix de la **température initiale** T_0 a un impact crucial. Si elle est trop basse, la perturbation stochastique est insuffisante pour sortir le réseau de son attracteur initial ; si elle est trop élevée, toute structuration précoce de $\omega_{i,j}$ est détruite, et le SCN passe son temps en réarrangements chaotiques. Il convient, en pratique, d'ajuster T_0 par essais, parfois en s'appuyant sur une estimation du “coût” des barrières typiques.

Le rythme de **refroidissement** (vitesse à laquelle α fait décroître T) détermine la portée de l'exploration. Un refroidissement extrêmement lent peut tendre vers un optimum global, selon les théorèmes classiques du recuit simulé, mais cela s'avère onéreux en temps de calcul. Les approches plus pragmatiques s'accommodent d'une décroissance modérée qui équilibre la qualité de la solution et l'efficacité de la simulation.

Les **critères** d'arrêt (seuil minimal de température, durée maximum d'itérations, stabilité de la matrice ω) contribuent à définir la durée du recuit. Il faut laisser au SCN le temps d'**explorer**, tout en évitant de prolonger trop inutilement une phase de bruit intense qui empirerait le coût de calcul.

D. Liens formels avec la physique statistique.

Dans un **DSL** disposant d'une fonction “énergie” $\mathcal{J}(\omega)$, on peut imaginer un protocole de type Metropolis–Hastings, où l'on calcule la différence $\Delta\mathcal{J} = \mathcal{J}(\omega') - \mathcal{J}(\omega)$ et on accepte, selon la probabilité $\min\{1, \exp[-\Delta\mathcal{J}/T]\}$, une modification qui augmente l'énergie. Cet artifice modélise la possibilité de s'extraire d'un puits local, contrôlée par la température. Bien que la mise à jour d'un SCN ne soit pas toujours décrite en ces termes, la **logique** demeure : un niveau de T élevé inflige des **sauts** plus fréquents et plus importants, tandis qu'un faible T “gèle” l'état.

En physique, on observe des **transitions** de phase (paramagnétique à ferromagnétique, par exemple). Parallèlement, dans un SCN, un fort bruit peut maintenir un ensemble de liaisons “déstructuré”, puis un abaissement de T induit une **cristallisation** sous forme de **clusters** affirmés.

E. Conclusion.

Le **choix** de la “température” T est un **levier essentiel** pour piloter l'équilibre entre l'exploration (grâce à un bruit significatif) et la stabilisation (grâce à un bruit quasi nul). En début d'apprentissage, on privilégie une “phase chaude” (élevée T_0) qui facilite les sauts hors des minima locaux et la rupture d'oscillations non désirées. Puis, en réduisant progressivement $\sigma(t) \propto T(t)$, le réseau se “fige” dans une configuration plus stable, comparable à un métal cristallisé après un recuit lent.

Cette démarche **stochastique**, inspirée de la **physique statistique**, complète les mécanismes plus déterministes (inhibition, parsimonie) déjà déployés pour maîtriser la dynamique du SCN. Le paramétrage judicieux de T_0 , de α et du temps de refroidissement offre un **contrôle** fin sur l'ensemble du processus et peut considérablement améliorer le **résultat** final, notamment dans les situations où la fonction énergie \mathcal{J} possède de nombreuses vallées locales ou où le SCN développe des **oscillations** difficiles à résorber autrement.

4.5. Héritage de la Physique Statistique et des Systèmes Dynamiques

4.5.1. Notions de Contraction, Point Fixe, Attracteurs

- 4.5.1.1. Rappels mathématiques : $\| DF(\Omega^*) \| < 1$, etc.
- 4.5.1.2. Critères de stabilité locale ou globale, cas non linéaires.
- 4.5.1.3. Illustrations : petit SCN (3 ou 4 entités) pour montrer stabilisation exponentielle ou oscillations.

4.5.2. Énergie ou Pseudo-Énergie

- 4.5.2.1. Cas simple : $J(\Omega) = -\sum \omega_{i,j} S(i,j) + \tau/2 \sum (\omega_{i,j})^2$ (2.4.3).
- 4.5.2.2. Descente locale vs. paysage évolutif si $S(i,j)$ change.
- 4.5.2.3. Cas stationnaire : $\Delta \omega_{i,j} = 0 \rightarrow \omega_{i,j}^* = \frac{S(i,j)}{\tau}$.

4.5.3. Renvoi vers Chapitre 2.3 et 2.4

- 4.5.3.1. Bref rappel des sections 2.3.2 (attracteurs multiples), 2.4.3 (notion d'énergie).
- 4.5.3.2. Comment on operationalise ces idées maintenant dans un cadre “appliqué”.

4.5. Héritage de la Physique Statistique et des Systèmes Dynamiques

Tout au long des sections précédentes (4.2, 4.3, 4.4), nous avons souligné la **dynamique** d'un **SCN** (Synergistic Connection Network) : comment les pondérations $\omega_{i,j}$ évoluent, comment des clusters se forment ou des oscillations se produisent, et comment on peut les contrôler (inhibition, sparsification, recuit simulé). Derrière ces mécanismes se cachent des **concepts** issus de la **physique statistique** (modèles d'énergie, recuit, transitions de phase) et de la **théorie des systèmes dynamiques** (points fixes, cycles, attracteurs). Cette section (4.5) met en évidence cet **héritage** et explique pourquoi on parle souvent de “descente d'énergie” ou de “contraction” locale pour caractériser la stabilité d'un point fixe.

4.5.1. Notions de Contraction, Point Fixe, Attracteurs

La grande force d'un **SCN** (Deep Synergy Learning) réside dans sa **capacité** à faire émerger des **états** (clusters, arrangements) stables ou quasi stables. Pour analyser ces états, on s'appuie sur des outils mathématiques liés aux **points fixes** et aux **attracteurs** d'une application **F** (le “règlement” qui, à chaque itération, met à jour $\omega_{i,j}(t)$).

4.5.1.1. Rappels mathématiques : $\| DF(\Omega^*) \| < 1$, etc.

Dans l'analyse des dynamiques d'auto-organisation dans un **Synergistic Connection Network (SCN)**, il est essentiel de disposer d'un cadre mathématique permettant d'étudier la stabilité des

pondérations $\omega_{i,j}$. Ce cadre s'appuie sur la notion d'**équilibre** ou de **point fixe** d'une fonction de mise à jour globale, ainsi que sur la condition de **contraction locale** caractérisée par l'opérateur dérivé de cette fonction. Nous rappelons ici les principaux éléments de ce formalisme.

A. Formalisation de la mise à jour

On considère que la mise à jour globale des pondérations peut être décrite par une fonction

$$\mathbf{F}: \mathcal{W} \rightarrow \mathcal{W},$$

où \mathcal{W} désigne l'espace des matrices $\{\omega_{i,j}\}$, souvent assimilé à $\mathbb{R}^{n \times n}$ si l'on compte n entités dans le SCN. Ainsi, à l'itération t , le vecteur (ou la matrice) des pondérations est mis à jour selon

$$\boldsymbol{\omega}(t+1) = \mathbf{F}(\boldsymbol{\omega}(t)).$$

Un **point fixe** $\boldsymbol{\omega}^*$ est défini par la condition

$$\boldsymbol{\omega}^* = \mathbf{F}(\boldsymbol{\omega}^*),$$

ce qui signifie que si le système atteint cet état, aucune mise à jour ultérieure ne le modifie. Ce concept constitue la base pour étudier la convergence et la stabilité du SCN.

B. Condition de contraction locale

Pour qu'un point fixe $\boldsymbol{\omega}^*$ soit stable, il est nécessaire que la fonction de mise à jour contracte les petites perturbations autour de ce point. En termes mathématiques, on examine la **matrice jacobienne** de \mathbf{F} évaluée en $\boldsymbol{\omega}^*$, notée $D\mathbf{F}(\boldsymbol{\omega}^*)$. On impose qu'il existe une **norme** $\|\cdot\|$ sur l'espace \mathcal{W} telle que

$$\| D\mathbf{F}(\boldsymbol{\omega}^*) \| < 1.$$

Cette inégalité indique que toute petite perturbation δ autour de $\boldsymbol{\omega}^*$ est réduite par l'application de $D\mathbf{F}$, ce qui se traduit par un retour progressif vers le point fixe. La condition de contraction, souvent résumée par $\| D\mathbf{F}(\boldsymbol{\omega}^*) \| < 1$, est donc le critère de **stabilité locale**.

C. Exemple élémentaire en dimension 1

Considérons un système unidimensionnel où la mise à jour s'exprime par

$$\omega(t+1) = F(\omega(t)).$$

Un point fixe ω^* satisfait

$$\omega^* = F(\omega^*).$$

La condition de stabilité locale pour ce système s'écrit alors

$$|F'(\omega^*)| < 1,$$

ce qui assure qu'une petite perturbation autour de ω^* sera contractée par l'itération. En dimension supérieure, cette condition se généralise par l'exigence que la norme de la matrice jacobienne $D\mathbf{F}(\boldsymbol{\omega}^*)$ soit strictement inférieure à 1.

D. Lien avec la dynamique additive

Dans le DSL, la mise à jour des pondérations se fait généralement par une règle additive de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ est la synergie entre les entités \mathcal{E}_i et \mathcal{E}_j , η le taux d'apprentissage, et τ le coefficient de décroissance. Pour analyser la stabilité autour du point fixe, nous supposons que $S(i,j)$ est constant et que la dynamique converge vers $\omega_{i,j}^*$. En posant

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*,$$

nous avons

$$\omega_{i,j}^* = \omega_{i,j}^* + \eta [S(i,j) - \tau \omega_{i,j}^*].$$

En simplifiant, nous obtenons

$$S(i,j) - \tau \omega_{i,j}^* = 0 \quad \Rightarrow \quad \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Pour étudier la stabilité locale, on examine la dérivée locale de la fonction de mise à jour. Dans ce schéma simplifié, la dérivée par rapport à $\omega_{i,j}$ s'exprime par

$$\frac{d}{d\omega_{i,j}} [\omega_{i,j} + \eta (S(i,j) - \tau \omega_{i,j})] = 1 - \eta \tau.$$

La condition de convergence locale exige que

$$|1 - \eta \tau| < 1.$$

Cette inégalité traduit la **contraction** autour du point fixe, c'est-à-dire que toute perturbation sera atténuée au fil des itérations, garantissant la stabilité du système.

Conclusion partielle

Le critère de contraction, symbolisé par l'inégalité

$$\|\mathbf{DF}(\boldsymbol{\omega}^*)\| < 1,$$

constitue la base pour vérifier la **stabilité** d'un point fixe dans un SCN. Lorsque l'on applique la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

et que l'on considère un cas stationnaire où $S(i,j)$ est constant, le système converge vers un équilibre donné par

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

La dérivée locale $1 - \eta \tau$ doit être inférieure à 1 en valeur absolue, ce qui assure que toute perturbation autour de l'équilibre est contractée, garantissant ainsi la stabilité locale du SCN. Cette interprétation en termes de **descente d'énergie** ou de **gradient** offre une vision intuitive de la manière dont le DSL équilibre le renforcement des liens avec la nécessité de réguler leur croissance, évitant ainsi une explosion des pondérations. Bien entendu, dans des cas plus complexes où $S(i, j)$ varie dans le temps, l'analyse se complique, mais le principe fondamental demeure : la stabilité du point fixe repose sur la condition de contraction de la fonction de mise à jour.

4.5.1.2. Critères de stabilité locale ou globale, cas non linéaires

Dans l'analyse des systèmes dynamiques qui sous-tendent un **Deep Synergy Learning (DSL)**, la question de la stabilité des points fixes et des attracteurs est centrale. En effet, pour garantir que les pondérations $\omega_{i,j}$ convergent vers des valeurs qui reflètent correctement la **synergie** entre entités, il convient de s'appuyer sur des critères mathématiques issus de la théorie des systèmes dynamiques. Ces critères se distinguent en deux catégories : la **stabilité locale**, qui examine le comportement du système à proximité d'un point fixe, et la **stabilité globale**, qui concerne la convergence de la dynamique depuis presque toute condition initiale. La présence de **non-linéarités** telles que l'inhibition, les couplages multiplicatifs ou les mécanismes de saturation complique l'analyse, mais le principe de base demeure celui de la contraction autour de l'attracteur.

A. Stabilité locale : La contraction au voisinage du point fixe

Pour étudier la stabilité locale d'un point fixe Ω^* dans un SCN, on considère la fonction de mise à jour globale

$$\mathbf{F}: \mathcal{W} \rightarrow \mathcal{W},$$

définie sur l'espace \mathcal{W} des matrices de pondérations. À chaque itération, le vecteur de pondérations se met à jour selon

$$\omega(t+1) = \mathbf{F}(\omega(t)).$$

Un point fixe Ω^* satisfait alors la condition

$$\Omega^* = \mathbf{F}(\Omega^*).$$

La **stabilité locale** se caractérise par la capacité du système à ramener toute perturbation infinitésimale autour de Ω^* vers ce point fixe. Pour cela, on linéarise \mathbf{F} autour de Ω^* en introduisant la **matrice jacobienne** $\mathbf{DF}(\Omega^*)$. La condition de contraction est alors exprimée par

$$\|\mathbf{DF}(\Omega^*)\| < 1,$$

où $\|\cdot\|$ désigne une norme appropriée (par exemple, la norme spectrale) et où le rayon spectral $\rho(\mathbf{DF}(\Omega^*))$ est inférieur à 1. En d'autres termes, toute perturbation δ autour du point fixe satisfait, pour une itération :

$$\omega(t+1) - \Omega^* \approx \mathbf{DF}(\Omega^*)(\omega(t) - \Omega^*),$$

et la norme de l'écart se contracte à chaque itération, garantissant que

$$\lim_{t \rightarrow \infty} \| \boldsymbol{\omega}(t) - \boldsymbol{\Omega}^* \| = 0.$$

Cette condition de contraction, analogue à la condition en dimension 1 $|F'(x^*)| < 1$, assure que dans un voisinage de $\boldsymbol{\Omega}^*$, la dynamique est **attractive** et les petites perturbations sont rapidement annulées par le processus itératif.

B. Stabilité globale : Convergence depuis presque toute condition initiale

La **stabilité globale** implique que, quel que soit l'état initial $\boldsymbol{\omega}(0)$ (à l'exception d'un ensemble de mesure nulle), la trajectoire converge vers le point fixe $\boldsymbol{\Omega}^*$, c'est-à-dire

$$\lim_{t \rightarrow \infty} \| \boldsymbol{\omega}(t) - \boldsymbol{\Omega}^* \| = 0.$$

Pour démontrer la stabilité globale, il faut que la fonction de mise à jour **F** soit **contractante** sur l'ensemble de l'espace \mathcal{W} , et non seulement dans un voisinage de $\boldsymbol{\Omega}^*$. Cette condition est beaucoup plus forte et rarement vérifiée dans des systèmes non linéaires à haute dimension, où l'on observe souvent la coexistence de plusieurs attracteurs ou de **bassins d'attraction** distincts. Dans ces cas, le système présente une **multi-stabilité** et la convergence dépend fortement de la condition initiale.

C. Cas non linéaires et phénomènes complexes

Les systèmes réels de DSL intègrent fréquemment des **non-linéarités** telles que des mécanismes d'inhibition, des couplages multiplicatifs ou des saturations. Par exemple, une mise à jour du type multiplicatif peut être exprimée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \exp(\eta [S(i,j) - \tau \omega_{i,j}(t)]),$$

ou encore des termes d'inhibition peuvent être ajoutés pour moduler la dynamique en fonction des interactions globales, donnant lieu à une mise à jour plus complexe du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Ces non-linéarités rendent l'analyse de la stabilité plus subtile. La **matrice jacobienne** de la fonction de mise à jour devient alors un opérateur qui intègre des termes croisés et dont les valeurs propres peuvent être complexes ou dépasser 1 en module, ce qui conduit à des phénomènes tels que les **oscillations**, les **bifurcations** ou même le **chaos**. Dans ces régimes, le système ne converge pas nécessairement vers un point fixe stable, mais peut osciller ou présenter des attracteurs étranges. Les méthodes de **contrôle** telles que l'inhibition, la **sparsification** ou l'ajustement dynamique des paramètres η et τ sont alors nécessaires pour maintenir une dynamique raisonnablement stable.

D. Conclusion

Les **critères de stabilité**, qu'ils soient locaux ou globaux, reposent sur l'analyse de la **contraction** de la fonction de mise à jour par le biais de la matrice jacobienne, avec la condition fondamentale

$$\| D\mathbf{F}(\boldsymbol{\Omega}^*) \| < 1.$$

En situation stationnaire, cette condition se traduit par la convergence vers un point fixe Ω^* où, dans le cas d'une règle additive simple,

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Lorsque la dynamique est **non linéaire** – en raison de mécanismes d'inhibition, de couplages multiplicatifs ou d'autres sources de non-linéarité – l'analyse devient plus complexe, et la stabilité locale peut être assurée dans un voisinage de l'attracteur, tandis que la stabilité globale n'est garantie que sous des hypothèses fortes de contraction sur tout l'espace. En pratique, ces critères, ainsi que des phénomènes comme les oscillations et les bifurcations, illustrent l'héritage des **systèmes dynamiques** et de la **physique statistique** dans l'étude du DSL. Cette approche permet non seulement de garantir la convergence locale des pondérations, mais aussi d'adapter le système en présence de perturbations, en utilisant des techniques de contrôle et de régulation pour préserver la **robustesse** et la **cohérence** de l'auto-organisation du SCN.

4.5.1.3. Illustrations : Petit SCN (3 ou 4 entités) pour montrer stabilisation exponentielle ou oscillations

Afin de rendre concrètes les notions de stabilité, d'attracteurs et de convergence évoquées dans les sections précédentes (notamment 4.5.1.1 et 4.5.1.2), il est instructif d'étudier des cas à petite échelle, par exemple un SCN constitué de trois ou de quatre entités. Ces illustrations permettent de mettre en évidence, de manière numérique et graphique, comment la dynamique de mise à jour des pondérations $\omega_{i,j}$ peut converger de manière exponentielle vers un point fixe ou, au contraire, présenter des oscillations périodiques, en fonction du choix des paramètres d'apprentissage et de décroissance.

A. Cas à 3 entités : Stabilisation exponentielle

Considérons un SCN formé de trois entités, notées \mathcal{E}_1 , \mathcal{E}_2 et \mathcal{E}_3 . Pour simplifier l'analyse, nous supposerons que la **synergie** $S(i,j)$ entre chaque paire d'entités est stationnaire, c'est-à-dire constante au cours du temps. Par exemple, nous définissons les synergies par les valeurs suivantes :

$$S(1,2) = 0.8, \quad S(1,3) = 0.7, \quad S(2,3) = 0.2,$$

et par symétrie, $S(2,1) = 0.8$, $S(3,1) = 0.7$ et $S(3,2) = 0.2$. Dans ce scénario, la mise à jour des pondérations est réalisée selon la règle additive classique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage** et τ est le **coefficent de décroissance**. Pour illustrer la stabilisation exponentielle, nous choisissons par exemple $\eta = 0.05$ et $\tau = 1.0$. L'initialisation des pondérations est faite avec des valeurs proches de zéro, par exemple, $\omega_{i,j}(0) \approx 0$ (avec une légère perturbation aléatoire pour simuler du bruit).

Au cours des itérations, les pondérations $\omega_{1,2}(t)$ et $\omega_{1,3}(t)$ augmentent progressivement car $S(1,2) = 0.8$ et $S(1,3) = 0.7$ sont relativement élevées, tandis que $\omega_{2,3}(t)$ croît plus lentement en

raison d'une synergie moindre $S(2,3) = 0.2$. Par l'analyse de la règle de mise à jour, nous pouvons observer qu'en supposant une convergence stationnaire, c'est-à-dire en posant $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$, nous obtenons :

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Ainsi, pour le lien entre \mathcal{E}_1 et \mathcal{E}_2 , on a $\omega_{1,2}^* = 0.8$; pour le lien entre \mathcal{E}_1 et \mathcal{E}_3 , $\omega_{1,3}^* = 0.7$; et pour le lien entre \mathcal{E}_2 et \mathcal{E}_3 , $\omega_{2,3}^* = 0.2$. En outre, la dynamique de convergence se caractérise par une décroissance exponentielle de l'écart, puisque la mise à jour linéaire entraîne que la différence $|\omega_{i,j}(t) - \omega_{i,j}^*|$ diminue en proportion d'un facteur constant $(1 - \eta \tau)$ à chaque itération. Graphiquement, en traçant $\omega_{1,2}(t)$ en fonction de t , on observe une courbe exponentielle ascendante se rapprochant asymptotiquement de 0.8. Ce cas illustre ainsi la **stabilité locale** dans un SCN simple, où la dynamique converge de manière prévisible vers des points fixes déterminés par la synergie stationnaire.

B. Cas à 4 entités : Apparition d'oscillations

Pour examiner un comportement différent, considérons maintenant un SCN comportant quatre entités, notées \mathcal{E}_1 , \mathcal{E}_2 , \mathcal{E}_3 et \mathcal{E}_4 . Supposons des synergies stationnaires qui varient de façon à créer des interactions compétitives, par exemple :

$$S(1,2) = 0.9, \quad S(2,3) = 0.8, \quad S(3,4) = 0.7, \quad S(4,1) = 0.8,$$

et pour certaines paires moins influentes, $S(1,3) = 0.1$ et $S(2,4) = 0.2$. Dans ce scénario, nous adoptons une **mise à jour multiplicative** afin d'exacerber les effets non linéaires, en utilisant une règle du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) [1 + \eta S(i,j) - \eta \tau \omega_{i,j}(t)].$$

Nous choisissons ici des paramètres plus agressifs, par exemple $\eta = 0.8$ et $\tau = 0.2$, et nous initialisons les pondérations avec une valeur faible, par exemple $\omega_{i,j}(0) \approx 0.01$. Dans ce régime, en raison du taux d'apprentissage élevé et de la faible décroissance, le système peut entrer dans une phase oscillatoire. Concrètement, les pondérations des liens avec des synergies élevées, telles que $\omega_{1,2}(t)$, $\omega_{2,3}(t)$, $\omega_{3,4}(t)$ et $\omega_{4,1}(t)$, vont d'abord augmenter rapidement. Cependant, lorsque l'un de ces liens atteint un niveau trop élevé, l'effet multiplicatif couplé à la rétroaction négative induite par $-\eta \tau \omega_{i,j}(t)$ provoque un sur-correction qui fait diminuer ce lien, tandis qu'un autre lien compense en augmentant, créant ainsi un cycle de **ping-pong** entre les différentes pondérations. Par exemple, $\omega_{1,2}(t)$ peut augmenter jusqu'à un certain point puis décroître brusquement lorsque $\omega_{2,3}(t)$ prend le relais, et vice versa. Ces oscillations se traduisent par des cycles quasi-périodiques dans la dynamique du SCN, et la convergence n'est pas atteinte de manière statique, mais le système évolue autour d'un attracteur cyclique.

Ce comportement oscillatoire, qui peut être observé en traçant les valeurs de $\omega_{i,j}(t)$ en fonction de t sur un graphique, démontre que le système, en l'absence d'inhibition ou de régulation supplémentaire, peut se comporter de manière non monotone, illustrant ainsi la sensibilité du DSL aux paramètres η et τ , et l'importance de mécanismes de contrôle supplémentaires (tels que l'inhibition latérale ou le clipping) pour éviter des comportements indésirables.

C. Illustration par simulation en Python

Pour mettre en évidence ces deux régimes de comportement, on peut se référer à un exemple de simulation en Python. Dans le cas à 3 entités, la mise à jour additive est appliquée avec un taux d'apprentissage modéré et un coefficient de décroissance suffisant pour assurer une convergence exponentielle. En revanche, dans le cas à 4 entités, un modèle multiplicatif avec des paramètres agressifs engendre des oscillations. Voici un extrait de pseudo-code illustratif pour le cas à 3 entités :

```

import numpy as np
import matplotlib.pyplot as plt

# Définition des synergies stationnaires pour 3 entités
S = {(1,2): 0.8, (1,3): 0.7, (2,3): 0.2}
# Assurer la symétrie
for (i, j), val in list(S.items()):
    S[(j, i)] = val

eta = 0.05 # Taux d'apprentissage
tau = 1.0 # Coefficient de décroissance
num_iterations = 30

# Initialisation des poids (petit bruit autour de 0)
np.random.seed(42)
weights = {(i, j): np.random.uniform(-0.01, 0.01) for (i, j) in S.keys()}

# Enregistrement de l'évolution des poids pour chaque paire
trajectories = {key: [] for key in weights.keys()}

for t in range(num_iterations):
    for (i, j) in weights.keys():
        omega = weights[(i, j)]
        delta = eta * (S[(i, j)] - tau * omega)
        weights[(i, j)] += delta
        trajectories[(i, j)].append(weights[(i, j)])

# Tracé des trajectoires
plt.figure(figsize=(10, 6))
for (i, j), traj in trajectories.items():
    plt.plot(traj, label=f'$\omega_{i,j}(t)$')
plt.axhline(0.8, color='blue', linestyle='--', label='$S(1,2)=0.8$')
plt.axhline(0.7, color='orange', linestyle='--', label='$S(1,3)=0.7$')
plt.axhline(0.2, color='green', linestyle='--', label='$S(2,3)=0.2$')
plt.title("Stabilisation exponentielle dans un SCN à 3 entités")
plt.xlabel("Itérations")
plt.ylabel("Pondération $\omega_{i,j}(t)$")
plt.legend()

```

```
plt.grid()
plt.show()
```

Pour le cas à 4 entités, une variante multiplicative avec des paramètres plus élevés (par exemple, $\eta = 0.8$ et $\tau = 0.2$) et l'introduction de termes non linéaires peut être utilisée pour simuler des oscillations, révélant un comportement cyclique dans la mise à jour des pondérations.

D. Conclusion

Les exemples illustratifs sur de petits SCN, comportant 3 ou 4 entités, démontrent de manière concrète comment la dynamique de mise à jour des pondérations $\omega_{i,j}$ peut soit converger de manière exponentielle vers un point fixe (dans le cas stationnaire où $S(i,j)$ est constant et les paramètres η et τ sont bien choisis), soit présenter des oscillations persistantes (dans un régime non linéaire ou avec des paramètres trop agressifs). Ces comportements illustrent l'importance des choix paramétriques et des mécanismes additionnels (comme l'inhibition et le clipping) pour garantir la stabilité et la robustesse de l'auto-organisation du SCN dans un DSL. Ces exemples à petite échelle servent de laboratoire expérimental permettant de vérifier théoriquement la condition de contraction et de comprendre, par analogie, comment la dynamique se généralisera dans des systèmes de plus grande taille comportant de nombreux attracteurs et des interactions complexes.

4.5.2. Énergie ou Pseudo-Énergie

Une façon très féconde de **comprendre** la dynamique d'un **SCN** (Synergistic Connection Network) est d'introduire l'idée d'une **fonction d'énergie** (ou **pseudo-énergie**) qui, en quelque sorte, **mesure** la qualité de la configuration ω . Cette idée, inspirée de la **physique statistique** (chap. 2.4), permet d'envisager la mise à jour $\omega(t)$ comme une forme de "descente" (ou de quasi-descente) dans un **paysage** énergétique — ou du moins comme un processus régulé par un critère qui ressemble à une minimisation.

4.5.2.1. Cas simple : $\mathcal{J}(\Omega) = -\sum \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum (\omega_{i,j})^2$ (2.4.3)

Dans cette section, nous développons de manière approfondie le modèle énergétique qui sous-tend la mise à jour des pondérations dans un **Deep Synergy Learning (DSL)**. L'idée centrale est de définir une **fonction d'énergie** ou **fonction potentielle** $\mathcal{J}(\Omega)$ sur l'ensemble des pondérations Ω du **Synergistic Connection Network (SCN)**, de telle sorte que la dynamique d'auto-organisation puisse être interprétée comme une descente vers un minimum de cette énergie. La forme simple considérée ici est

$$\mathcal{J}(\Omega) = - \sum_{(i,j)} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{(i,j)} (\omega_{i,j})^2,$$

où $S(i,j)$ désigne la **synergie** entre les entités \mathcal{E}_i et \mathcal{E}_j , $\omega_{i,j}$ représente la **pondération** ou la force du lien entre ces entités, et τ est un paramètre qui module la décroissance des pondérations. Cette fonction \mathcal{J} traduit l'intuition que l'on souhaite **maximiser** la synergie globale entre les entités tout en empêchant une croissance excessive des pondérations.

D'un point de vue interprétatif, le terme $-\sum \omega_{i,j} S(i,j)$ favorise l'augmentation des liens lorsque la synergie est élevée. En effet, plus $S(i,j)$ est grand, plus le produit $\omega_{i,j} S(i,j)$ contribue négativement à \mathcal{J} ; puisque l'objectif de la descente consiste à minimiser \mathcal{J} , la dynamique incite à augmenter $\omega_{i,j}$ pour réduire la valeur absolue de ce terme négatif, ce qui correspond à renforcer les connexions entre entités fortement synergiques. À l'inverse, le terme de régularisation $\frac{\tau}{2} \sum (\omega_{i,j})^2$ pénalise la croissance excessive des pondérations, en augmentant l'énergie si les valeurs de $\omega_{i,j}$ deviennent trop grandes. Ce compromis permet ainsi d'éviter une **croissance non contrôlée** des liens, garantissant que la solution convergera vers un équilibre.

Pour rendre cette intuition plus formelle, nous pouvons envisager une analyse variationnelle dans un cadre continu. Supposons que la dynamique de mise à jour soit modélisée par une équation différentielle approchée

$$\frac{d \omega_{i,j}}{dt} = \eta [S(i,j) - \tau \omega_{i,j}],$$

où η est le taux d'apprentissage, et où l'on suppose que $S(i,j)$ ne dépend pas de $\omega_{i,j}$. Dans ce cas, la dérivée partielle de \mathcal{J} par rapport à $\omega_{i,j}$ est donnée par

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau \omega_{i,j}.$$

La descente de gradient, qui tend à minimiser \mathcal{J} , correspond à une dynamique

$$\frac{d \omega_{i,j}}{dt} = -\eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = \eta [S(i,j) - \tau \omega_{i,j}].$$

Cette équation confirme que la dynamique de mise à jour des pondérations est en effet équivalente à une **desccente de gradient** sur \mathcal{J} . Au point fixe stationnaire, on pose $\frac{d \omega_{i,j}}{dt} = 0$ et l'on obtient

$$S(i,j) - \tau \omega_{i,j}^* = 0 \quad \Rightarrow \quad \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Ainsi, la solution d'équilibre, ou **attracteur**, est directement liée à la synergie $S(i,j)$ et régulée par le coefficient τ .

Il convient toutefois de noter que cette formulation constitue un **cas simple** dans lequel la synergie $S(i,j)$ est supposée constante au cours du temps et indépendante des pondérations. Dans des situations réelles, notamment lorsque des mécanismes non linéaires (inhibition latérale, couplages multiplicatifs, etc.) interviennent, la fonction \mathcal{J} peut inclure des termes supplémentaires, et la descente de gradient n'est plus exacte. C'est pourquoi on parle parfois de **pseudo-énergie**, car la dynamique réelle du DSL tente de réduire \mathcal{J} mais peut ne pas garantir une décroissance monotone à chaque itération en raison de la présence de rétroactions complexes.

La présence du terme de régularisation $\frac{\tau}{2} \sum (\omega_{i,j})^2$ joue un rôle essentiel, car il agit comme un **coût** associé au maintien d'un lien fort. Ce mécanisme empêche une croissance illimitée des pondérations et assure qu'en l'absence de synergie suffisante, les liens se désactivent

progressivement. Ainsi, l'équilibre atteint, $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$, représente le compromis optimal entre le renforcement des liens et la régulation nécessaire pour préserver la **stabilité** du système.

En résumé, la fonction d'énergie

$$\mathcal{J}(\boldsymbol{\Omega}) = - \sum_{(i,j)} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{(i,j)} (\omega_{i,j})^2,$$

formalise l'intuition selon laquelle le système cherche à **maximiser** la synergie globale $\sum \omega_{i,j} S(i,j)$ tout en **pénalisant** la croissance excessive des liens via le terme quadratique. La dynamique induite par la mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

correspond alors à une descente de gradient sur \mathcal{J} dans un cas simple. Cette approche permet d'atteindre un point fixe stable caractérisé par

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau},$$

ce qui offre une interprétation claire en termes d'**équilibre** entre l'activation induite par la synergie et la régulation imposée par la décroissance. Cette formulation constitue la base théorique sur laquelle s'appuient des mécanismes plus complexes dans le DSL, et sert de référence pour analyser la stabilité et la convergence des pondérations dans le réseau, même lorsque des non-linéarités supplémentaires sont introduites.

4.5.2.2. Descente locale vs. paysage évolutif si $S(i,j)$ change

Dans l'analyse de la dynamique d'un **Deep Synergy Learning (DSL)**, la fonction d'énergie $\mathcal{J}(\boldsymbol{\omega})$ joue un rôle central pour comprendre comment la mise à jour des pondérations $\omega_{i,j}$ tend à converger vers un minimum. Dans la partie précédente, nous avons étudié le cas simple où $S(i,j)$ est considéré comme constant, ce qui conduit à une descente locale dans un paysage énergétique fixe. Cependant, dans de nombreux systèmes réels, la synergie $S(i,j)$ varie au fil du temps, en raison de la mise à jour continue des représentations (embeddings) ou de l'évolution des règles logiques. Cette variabilité entraîne la formation d'un **paysage évolutif** où la fonction d'énergie se modifie continuellement, et, par conséquent, la dynamique d'auto-organisation ne se stabilise pas de manière statique. Nous détaillons ci-après les deux scénarios et leurs implications.

A. Descente locale lorsque $S(i,j)$ est constant

Lorsque la représentation des entités est stable, la synergie $S(i,j)$ est considérée comme une valeur fixe, ce qui signifie que, pour toute paire d'entités, le score $S(i,j)$ reste inchangé au cours des itérations. Dans ce cas, la fonction d'énergie

$$\mathcal{J}(\boldsymbol{\omega}) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2$$

devient indépendante du temps. La mise à jour des pondérations est alors réalisée par la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

et le système se comporte comme une descente de gradient sur \mathcal{J} . En effet, en posant l'équilibre stationnaire $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$, on obtient immédiatement

$$S(i,j) - \tau \omega_{i,j}^* = 0 \quad \Rightarrow \quad \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Dans ce contexte, toute perturbation autour de ce point fixe est contractée par le système, de sorte que la dynamique converge de manière exponentielle vers $\omega_{i,j}^*$. La vitesse de convergence est essentiellement dictée par le facteur $1 - \eta \tau$, et l'écart à l'équilibre se réduit comme $|\omega_{i,j}(t) - \omega_{i,j}^*| \sim (1 - \eta \tau)^t$. Ce cas de **descente locale** est idéal pour l'analyse théorique, car il offre un cadre simple et linéaire où le paysage énergétique \mathcal{J} est fixe, et où les mécanismes d'auto-organisation peuvent être décrits de manière analytique.

B. Paysage évolutif lorsque $S(i,j)$ change

En réalité, de nombreux **DSL** opèrent dans des environnements non stationnaires, où la synergie $S(i,j)$ est recalculée à chaque itération. Cette situation peut se produire lorsque :

- Les **embeddings** des entités sont continuellement mis à jour par des processus de fine-tuning ou d'apprentissage continu, entraînant ainsi une modification de $S(i,j)$ basée sur la proximité vectorielle.
- De nouvelles entités sont introduites dans le système, modifiant la distribution globale et redéfinissant ainsi les relations entre les entités existantes.
- Les **règles symboliques** évoluent (par exemple, par ajout ou suppression d'axiomes), ce qui modifie la composante logique du score de synergie.

Dans un tel scénario, la fonction d'énergie devient dépendante du temps et s'écrit de manière implicite comme

$$\mathcal{J}(\omega, t) = - \sum_{i,j} \omega_{i,j} S(i,j, t) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2,$$

où $S(i,j, t)$ varie à chaque itération. La dynamique de mise à jour s'adapte alors en temps réel :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j, t) - \tau \omega_{i,j}(t)].$$

Dans ce **paysage évolutif**, le minimum de \mathcal{J} n'est pas fixe mais se déplace continuellement. Le système tente alors de « suivre » ce minimum mouvant, ce qui peut conduire à des ajustements constants des pondérations. Si les variations de $S(i,j, t)$ sont **douces**, la dynamique peut suivre le déplacement du minimum local, et le SCN se réorganise progressivement sans perdre sa cohérence. Cependant, si $S(i,j, t)$ subit des variations brutales – par exemple, en cas d'insertion massive de nouvelles entités ou de modifications importantes dans les représentations – le paysage énergétique

peut se transformer de manière significative, et le système peut alors se retrouver dans une situation de **rebondissement** où les pondérations oscillent ou se réajustent continuellement, sans parvenir à une stabilisation définitive.

Ces phénomènes peuvent être analysés en considérant la dérivée temporelle de \mathcal{J} et en observant que, contrairement au cas stationnaire, le gradient $-\nabla \mathcal{J}(\boldsymbol{\omega}, t)$ varie avec t . Le système est alors contraint de s'adapter à un environnement énergétique qui évolue, et la dynamique d'auto-organisation peut présenter des comportements complexes, tels que des transitions de phase ou des oscillations persistantes.

C. Conséquences pratiques et stratégies d'adaptation

Dans un DSL opérant dans un paysage évolutif, la stabilité du SCN est moins garantie que dans le cas stationnaire. La non-stationnarité de $S(i, j, t)$ requiert des mécanismes supplémentaires pour assurer la **robustesse** du système. Parmi ces mécanismes, on peut citer :

- **Lissage des embeddings** : en réévaluant les représentations de manière graduelle, par exemple en utilisant une mise à jour du type

$$\mathbf{x}_i(t+1) \leftarrow \alpha \mathbf{x}_i(t) + (1 - \alpha) g_{t+1}(\mathbf{d}_i),$$

où g_{t+1} représente la fonction d'extraction mise à jour et $\alpha \in [0,1]$ un coefficient de lissage, on atténue les variations brusques de $S(i, j, t)$.

- **Réindexation partielle** : lorsqu'un grand changement se produit (nouveaux modèles, nouvelles entités), il peut être nécessaire de recalculer globalement ou partiellement les pondérations $\omega_{i,j}$ pour réaligner le SCN avec la nouvelle configuration du paysage énergétique.
- **Utilisation de mécanismes de contrôle** : des stratégies inspirées du recuit stochastique ou de la régulation adaptative peuvent être mises en place pour permettre au système de “suivre” le minimum local tout en évitant des oscillations excessives ou des divergences.

Ces stratégies permettent d'adapter le SCN à un environnement où $S(i, j, t)$ évolue continuellement, assurant ainsi une **adaptabilité** optimale du DSL même lorsque le paysage énergétique est en mouvement.

D. Conclusion

En résumé, lorsque la synergie $S(i, j)$ est constante, la dynamique de mise à jour des pondérations $\omega_{i,j}$ s'apparente à une **descente locale** dans un paysage énergétique fixe, conduisant à la convergence vers un point fixe défini par

$$\omega_{i,j}^* = \frac{S(i, j)}{\tau}.$$

Cette situation permet une analyse simple de la stabilité du système via la descente de gradient. En revanche, lorsque $S(i, j)$ varie au fil du temps – en raison de mises à jour des représentations ou d'insertion de nouvelles données – le paysage énergétique $\mathcal{J}(\boldsymbol{\omega}, t)$ devient mobile, et le SCN doit s'adapter continuellement. Dans ce cas, le système ne se fige pas dans un attracteur fixe, mais réorganise ses pondérations pour suivre le minimum local en évolution. Ce comportement

dynamique, qui peut se traduire par des réajustements constants voire des oscillations, souligne l'importance de mécanismes de contrôle supplémentaires pour garantir la **robustesse** du DSL dans des environnements non stationnaires. La distinction entre descente locale et paysage évolutif permet ainsi de mieux comprendre comment la variabilité de $S(i,j)$ influence la **stabilité**, la **convergence** et l'**adaptabilité** du système, fournissant une base théorique pour l'analyse des performances d'un DSL dans des contextes évolutifs et réels.

4.5.2.3. Cas stationnaire : $\Delta\omega_{i,j} = 0 \Rightarrow \omega_{i,j}^* = \frac{S(i,j)}{\tau}$

Dans ce cas, nous nous intéressons à la situation idéale dans laquelle la **synergie** $S(i,j)$ entre chaque paire d'entités est considérée comme **constante** au cours du temps, ce qui conduit à une dynamique de mise à jour des pondérations qui ne dépend que de la valeur actuelle de $\omega_{i,j}$ et des paramètres du système. Cette hypothèse simplificatrice permet de décrire le comportement stationnaire du système, c'est-à-dire le point fixe vers lequel tend la dynamique d'auto-organisation du **Synergistic Connection Network (SCN)**.

Soit la fonction d'**énergie** (ou pseudo-énergie) définie par

$$\mathcal{J}(\boldsymbol{\omega}) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2,$$

où $S(i,j)$ est un score de synergie constant pour la paire (i,j) et τ est le paramètre de décroissance. L'objectif est de minimiser \mathcal{J} afin de trouver une configuration optimale des pondérations. En effet, le premier terme, $-\sum \omega_{i,j} S(i,j)$, encourage le renforcement des liens lorsque la synergie est élevée, tandis que le terme de régularisation $\frac{\tau}{2} \sum (\omega_{i,j})^2$ pénalise une croissance excessive des pondérations. Le compromis entre ces deux contributions conduit à un minimum d'énergie qui définit l'équilibre du système.

Pour étudier ce cas stationnaire, nous utilisons la **règle additive** de mise à jour des pondérations donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η représente le **taux d'apprentissage**. Dans un état stationnaire, on suppose que la dynamique ne fait plus évoluer les pondérations, c'est-à-dire que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*.$$

En imposant la condition stationnaire, nous avons :

$$\omega_{i,j}^* = \omega_{i,j}^* + \eta [S(i,j) - \tau \omega_{i,j}^*].$$

La seule solution non triviale de cette équation est obtenue en annulant le terme entre parenthèses, ce qui conduit directement à

$$S(i,j) - \tau \omega_{i,j}^* = 0 \quad \Rightarrow \quad \omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette équation montre que, dans un environnement stationnaire, la pondération entre deux entités converge vers une valeur proportionnelle à la synergie mesurée, modulée par le paramètre de décroissance τ . En d'autres termes, si la synergie est élevée, le lien se renforce jusqu'à atteindre une valeur d'équilibre donnée par $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$. À l'inverse, si $S(i,j)$ est nul ou très faible, alors le lien tendra à disparaître (i.e., $\omega_{i,j}^*$ sera proche de 0).

Pour approfondir cette analyse, on peut interpréter la règle de mise à jour comme une **descente de gradient** sur la fonction d'énergie \mathcal{J} . En effet, la dérivée partielle de \mathcal{J} par rapport à $\omega_{i,j}$ est donnée par

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau \omega_{i,j}.$$

La dynamique de mise à jour du DSL, écrite en termes continus, s'exprime par

$$\frac{d \omega_{i,j}}{dt} = -\eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = \eta [S(i,j) - \tau \omega_{i,j}],$$

ce qui est exactement la forme de notre règle de mise à jour. Par conséquent, l'évolution des pondérations est guidée par le gradient de \mathcal{J} et tend à réduire l'énergie du système jusqu'à ce que le gradient soit nul, c'est-à-dire quand $\omega_{i,j}$ atteint l'équilibre.

Un **exemple numérique** simple peut illustrer ce point. Supposons que pour une paire d'entités, la synergie stationnaire est $S(i,j) = 0.8$ et que $\tau = 1.0$. Si l'on initialise $\omega_{i,j}(0)$ à une valeur proche de zéro, la mise à jour s'effectuera de manière à faire croître $\omega_{i,j}$ vers

$$\omega_{i,j}^* = \frac{0.8}{1.0} = 0.8.$$

On observe alors, par exemple, une convergence exponentielle telle que

$$\omega_{i,j}(t) \approx 0.8[1 - (1 - \eta\tau)^t],$$

ce qui traduit la réduction exponentielle de l'écart par rapport à l'équilibre. Graphiquement, une courbe de $\omega_{i,j}(t)$ en fonction de t montrerait une ascension rapide suivie d'un plateau asymptotique à 0.8, confirmant que la dynamique d'auto-organisation a atteint un état stationnaire.

Dans le cadre de cette approche, le terme $\tau \omega_{i,j}^2$ dans la fonction d'énergie sert de **pénalité** qui empêche la croissance excessive des pondérations. C'est ce terme régulateur qui assure la **stabilité** du système en imposant un coût à la maintenance de liens trop forts, et qui, conjointement avec le terme de synergie $\omega_{i,j} S(i,j)$, détermine l'équilibre optimal.

Conclusion

En conclusion, dans un **cas stationnaire** où la synergie $S(i,j)$ reste constante, la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

permet d'atteindre un point fixe $\omega_{i,j}^*$ donné par

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette solution correspond à un minimum local de la fonction d'énergie

$$\mathcal{J}(\boldsymbol{\omega}) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2,$$

et reflète l'équilibre entre l'effet de renforcement induit par la synergie et la pénalité qui empêche la croissance illimitée des liens. La dynamique de mise à jour agit alors comme une descente de gradient sur \mathcal{J} , assurant que toute perturbation autour du point fixe est progressivement amortie, ce qui garantit la stabilité du SCN dans un environnement stationnaire. Cette analyse fournit ainsi une base théorique solide pour comprendre la convergence et la stabilité des pondérations dans un DSL, dans des conditions idéales où le paysage énergétique reste fixe.

4.5.3. Renvoi vers Chapitre 2.3 et 2.4

Les sections précédentes (4.5.1 et 4.5.2) ont souligné l'importance de certains **concepts** (attracteurs, point fixe, énergie ou pseudo-énergie) pour comprendre la dynamique d'un SCN (Synergistic Connection Network). Mais l'origine de ces notions se situe déjà en **Chapitre 2**, lorsque nous abordions les **fondements théoriques** du DSL (Deep Synergy Learning) : la possibilité de multi-stabilité (2.3.2), l'analogie avec la physique statistique et les systèmes dynamiques (2.4.3). Cette sous-section (4.5.3) vise à **faire le lien** explicite avec ces passages et à voir **comment** on exploite concrètement (i.e. de manière “appliquée”) ces idées dans un DSL.

4.5.3.1. Bref rappel des sections 2.3.2 (attracteurs multiples) et 2.4.3 (notion d'énergie)

Dans l'analyse théorique du Deep Synergy Learning (DSL), les chapitres 2.3.2 et 2.4.3 fournissent respectivement les fondations conceptuelles concernant la co-existence d'attracteurs multiples et la formulation d'une fonction d'énergie sur l'espace des pondérations. Ces deux approches, bien que relevant de domaines différents de la théorie des systèmes dynamiques, se rejoignent pour expliquer la manière dont le Synergistic Connection Network (SCN) se structure et converge vers des configurations stables.

A. Attracteurs multiples (Chap. 2.3.2)

La notion d'attracteurs multiples trouve son origine dans la théorie des systèmes dynamiques, où l'on démontre que, pour un système non linéaire, il peut exister plusieurs points fixes ou cycles stables, c'est-à-dire plusieurs attracteurs locaux. Mathématiquement, si l'on définit la fonction de mise à jour globale du SCN par

$$\mathbf{F}: \mathcal{W} \rightarrow \mathcal{W},$$

un point fixe Ω^* vérifie

$$\boldsymbol{\Omega}^* = \mathbf{F}(\boldsymbol{\Omega}^*).$$

Dans des systèmes complexes tels que le SCN, il est souvent constaté que l'espace des pondérations se divise en différents bassins d'attraction. Ainsi, l'initialisation de $\boldsymbol{\omega}(0)$ détermine vers quel attracteur le système convergera. Cette multi-stabilité se manifeste par la formation de **clusters alternatifs** ou de configurations distinctes, chacune étant localement stable. Par analogie avec les modèles de spin (tels que le modèle d'Ising ou de Potts) et les réseaux de Hopfield, ces attracteurs multiples illustrent la possibilité pour le système d'atteindre des états globalement différents, bien que chacun satisfasse la condition d'équilibre local.

L'existence de plusieurs attracteurs implique également que la dynamique du SCN peut être sensible aux perturbations et que de légères variations dans l'état initial peuvent conduire à des résultats radicalement différents. Cette propriété, bien que complexe à analyser de manière exhaustive dans des espaces de grande dimension, constitue la base de nombreux phénomènes de **phase** et de **transition** observés dans les systèmes d'apprentissage non supervisé.

B. Notion d'énergie (Chap. 2.4.3)

Le chapitre 2.4.3 introduit l'idée d'une fonction d'énergie ou fonction potentielle $\mathcal{J}(\boldsymbol{\omega})$ définie sur l'ensemble des pondérations du SCN. Une forme simple de cette fonction est donnée par

$$\mathcal{J}(\boldsymbol{\omega}) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2,$$

où $S(i,j)$ représente la synergie entre les entités \mathcal{E}_i et \mathcal{E}_j et τ est le coefficient de décroissance, jouant ainsi le rôle d'un terme de régularisation. Cette fonction d'énergie est conçue de manière à ce que, lorsque l'on cherche à la minimiser (ou, de manière équivalente, à maximiser $\sum \omega_{i,j} S(i,j)$ tout en pénalisant la croissance excessive des pondérations), le système converge vers un état stable où les pondérations $\omega_{i,j}$ atteignent des valeurs optimales.

En effet, dans le cas stationnaire où $S(i,j)$ est constant, la descente de gradient sur \mathcal{J} conduit à la condition d'équilibre

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau \omega_{i,j} = 0,$$

ce qui implique que

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette relation fournit une interprétation élégante du compromis entre le renforcement d'un lien (favorisé par une synergie élevée) et la régulation de la croissance des pondérations (imposée par le terme quadratique). La fonction d'énergie permet également de visualiser le **paysage** dans lequel évolue le SCN. Dans ce paysage, chaque attracteur correspond à un minimum local de \mathcal{J} . Des analogies fortes peuvent être établies avec des systèmes physiques, tels que les réseaux de neurones de Hopfield, où la dynamique du système converge vers des états d'énergie minimale.

C. Lien entre attracteurs multiples et fonction d'énergie

L'héritage théorique posé dans le Chapitre 2 montre que la co-existence de plusieurs attracteurs découle naturellement de la forme de la fonction d'énergie. En effet, dans un système non linéaire, $J(\omega)$ peut présenter plusieurs vallées ou minima locaux, chacun correspondant à un ensemble stable de pondérations. Ainsi, en fonction de l'initialisation du système, le SCN convergera vers l'un de ces attracteurs, ce qui explique la **multi-stabilité** observée. Les attracteurs multiples sont alors interprétés comme des **clusters alternatifs** ou des configurations distinctes du réseau, chaque minimum local traduisant une organisation particulière des liens, qui peut être analysée en termes de similarité des entités et de la cohérence de leur interaction.

Ces concepts se retrouvent dans de nombreux modèles d'auto-organisation et d'apprentissage non supervisé. Par exemple, dans les réseaux de Hopfield, la dynamique converge vers des états mémorisés qui représentent des minima de l'énergie, tandis que dans les modèles de spin, la configuration du système est déterminée par l'équilibre entre les interactions locales et l'influence de l'environnement.

D. Résumé du lien entre Chapitre 2 et Chapitre 4

Le **Chapitre 2** pose les fondations théoriques en introduisant la notion d'attracteurs multiples et en définissant une fonction d'énergie $J(\omega)$ qui guide la dynamique d'auto-organisation du SCN. Cette approche conceptuelle permet d'interpréter la dynamique de mise à jour des pondérations comme une **descente d'énergie** qui, dans le cas stationnaire, conduit à des points fixes stables, et dans des systèmes plus complexes, à une multi-stabilité ou à des comportements oscillatoires.

Le **Chapitre 4** poursuit cette analyse en développant de manière plus "ingénierie" les mécanismes de mise à jour concrets, tels que la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

et en explorant les stratégies de contrôle (inhibition, clipping, etc.) qui permettent d'ajuster la dynamique du SCN pour éviter des comportements indésirables. Ainsi, la théorie des attracteurs et de la fonction d'énergie fournie au Chapitre 2 constitue la **base conceptuelle** sur laquelle s'appuie la mise en pratique détaillée dans le Chapitre 4, illustrant ainsi la continuité entre une vision théorique et une application pratique dans le cadre d'un DSL.

Conclusion

En somme, le rappel des sections 2.3.2 et 2.4.3 met en lumière deux aspects complémentaires de l'analyse des systèmes dynamiques dans un DSL. D'une part, la notion d'attracteurs multiples permet de comprendre comment différentes configurations stables peuvent coexister, en fonction des conditions initiales et des interactions locales entre entités. D'autre part, la fonction d'énergie $J(\omega)$ offre un cadre mathématique pour interpréter la dynamique de mise à jour des pondérations comme une descente vers un minimum, où le compromis entre renforcement par la synergie et régularisation par la décroissance est clairement établi. Ces concepts théoriques servent de socle à la mise en œuvre pratique, telle que développée dans le Chapitre 4, et illustrent comment un SCN peut évoluer de manière stable ou osciller, selon la structure du paysage énergétique et les mécanismes de contrôle appliqués.

4.5.3.2. Comment on opérationnalise ces idées maintenant dans un cadre “appliqué”

Dans cette section, nous développons en détail comment les concepts théoriques introduits dans le Chapitre 2 — tels que la notion d'énergie $\mathcal{J}(\boldsymbol{\omega})$, les attracteurs multiples et la descente d'énergie — sont traduits en procédures concrètes dans un système de **Deep Synergy Learning (DSL)**. L'objectif est de montrer comment, à travers des algorithmes implémentés en code, on peut mettre en œuvre la mise à jour des pondérations $\omega_{i,j}$ de manière à faire émerger des clusters stables ou des configurations attractives, tout en adaptant le système aux évolutions de la représentation.

A. Mise en œuvre de la descente d'énergie

L'idée fondamentale est de considérer la mise à jour des pondérations comme une descente de gradient sur une fonction d'énergie, qui dans sa forme simple est donnée par

$$\mathcal{J}(\boldsymbol{\omega}) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2,$$

où $S(i,j)$ représente la synergie entre les entités \mathcal{E}_i et \mathcal{E}_j et τ contrôle la pénalité appliquée aux grandes pondérations. Dans un cadre appliqué, cette fonction d'énergie sert à guider la mise à jour des poids par une procédure itérative. Plus précisément, l'algorithme se base sur la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

qui peut être interprétée comme une descente de gradient sur \mathcal{J} lorsque l'on observe que

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau \omega_{i,j}.$$

En pratique, on initialise le vecteur des pondérations $\boldsymbol{\omega}(0)$ (par exemple, en partant d'un bruit faible autour de zéro) et on exécute la boucle itérative suivante :

213. **Calcul du gradient local** : Pour chaque paire (i,j) , le gradient local $-S(i,j) + \tau \omega_{i,j}(t)$ est évalué. Ce terme indique la direction dans laquelle il faut modifier la pondération pour réduire l'énergie.

214. **Mise à jour** : On ajuste la pondération selon

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

215. **Clipping et régularisation** : Pour éviter des valeurs aberrantes ou une croissance non contrôlée, on applique des mécanismes tels que le clipping (par exemple, forcer $\omega_{i,j}(t+1) \in [0, \omega_{\max}]$) et éventuellement des termes d'inhibition.

L'ensemble de ces opérations s'exécute de manière itérative jusqu'à ce que la variation entre deux itérations consécutives, $\|\boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t)\|$, devienne inférieure à un seuil prédéfini, indiquant ainsi que le système a atteint un état quasi-stationnaire.

B. Gérer la multi-stabilité et l'adaptation dans un environnement évolutif

Dans des applications réelles, le système peut être amené à évoluer, que ce soit par l'insertion de nouvelles entités ou par l'actualisation des représentations (embeddings ou règles). Cela se traduit par une modification du paysage énergétique $\mathcal{J}(\omega, t)$ au fil du temps. Pour gérer ce phénomène, plusieurs stratégies opérationnelles peuvent être mises en place :

- **Mécanismes de lissage temporel** : Afin de limiter les variations brutales des synergies $S(i, j)$ dues à des mises à jour continues des représentations, on peut utiliser un schéma de lissage, par exemple en mettant à jour l'embedding \mathbf{x}_i de manière progressive selon la formule

$$\mathbf{x}_i(t+1) \leftarrow \alpha \mathbf{x}_i(t) + (1 - \alpha) g_{t+1}(\mathbf{d}_i),$$

où g_{t+1} est la fonction d'extraction mise à jour et $\alpha \in [0, 1]$ est un coefficient de lissage. Cela permet de faire évoluer $S(i, j, t)$ de manière graduelle, de sorte que la descente de gradient sur \mathcal{J} ne soit pas perturbée par des sauts trop brusques dans le paysage énergétique.

- **Réindexation et recalibration** : En cas de modification significative du système (par exemple, l'arrivée de nombreuses nouvelles entités), il peut être nécessaire de réinitialiser ou recalibrer partiellement les pondérations $\omega_{i,j}$ pour aligner le SCN avec la nouvelle configuration des représentations.
- **Utilisation de techniques de recuit stochastique** : Pour aider le système à sortir des minima locaux peu pertinents et explorer de nouveaux attracteurs plus profonds, on peut injecter du bruit contrôlé dans la mise à jour des pondérations. Ceci est analogue à un recuit stochastique où la “température” décroît progressivement afin de permettre au système d'échapper aux attracteurs peu stables tout en se stabilisant finalement dans un minimum de \mathcal{J} .

Ces mécanismes assurent que le SCN reste **adaptatif** et capable de suivre l'évolution des représentations, même lorsque le paysage énergétique est en constante mutation.

C. Mise en œuvre opérationnelle dans un cadre appliqué

Pour traduire ces idées dans un cadre appliqué, il convient de développer un pipeline algorithmique qui intègre à la fois la descente de gradient sur \mathcal{J} et les stratégies d'adaptation nécessaires à un environnement non stationnaire. Concrètement, le pipeline peut être structuré en plusieurs étapes clés :

- **Initialisation des pondérations** : Définir $\omega(0)$ avec des valeurs faibles, éventuellement avec un bruit aléatoire pour simuler des conditions réelles.
- **Calcul de la synergie** : Pour chaque paire d'entités, calculer $S(i, j)$ en fonction des représentations actuelles. Dans un environnement appliqué, ce calcul peut être basé sur des embeddings extraits par des modèles pré-entraînés ou sur la vérification de règles logiques pour des entités symboliques.
- **Boucle itérative de mise à jour** :
 - **Évaluation du gradient** : Pour chaque lien, calculer la variation $\Delta\omega_{i,j} = \eta[S(i, j) - \tau \omega_{i,j}(t)]$.

- **Application de la mise à jour :** Actualiser $\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \Delta\omega_{i,j}$.
- **Régulation :** Appliquer des mécanismes de clipping et, le cas échéant, des termes d'inhibition ou de recuit pour contrôler la croissance.
- **Surveillance de la convergence :** Calculer la différence $\|\boldsymbol{\omega}(t + 1) - \boldsymbol{\omega}(t)\|$ et suivre l'évolution de la valeur de $J(\boldsymbol{\omega}(t))$. Une fois ces valeurs stabilisées, le système est considéré comme convergent.
- **Adaptation en ligne :** En cas de changement dans les représentations (mise à jour des embeddings ou des règles), relancer le processus d'actualisation, éventuellement en réutilisant un schéma de lissage pour éviter des réajustements trop brusques.

Ces étapes peuvent être implémentées dans un cadre logiciel (par exemple en Python, avec des bibliothèques de calcul scientifique comme NumPy et des frameworks de deep learning), permettant ainsi d'opérationnaliser les concepts théoriques issus du Chapitre 2 dans un environnement appliqué.

D. Conclusion

La mise en œuvre opérationnelle des idées théoriques de la descente d'énergie et des attracteurs dans un DSL se traduit par la programmation d'un algorithme de mise à jour itérative qui cherche à minimiser une fonction d'énergie $J(\boldsymbol{\omega})$. Dans un cadre appliqué, ce processus inclut non seulement le calcul de la mise à jour des pondérations par descente de gradient, mais aussi des stratégies d'adaptation pour gérer la variabilité des représentations – par exemple, via le lissage des embeddings, la réindexation ou des techniques de recuit stochastique. Le résultat final est un SCN capable de former des clusters stables ou adaptatifs, en fonction de la stationnarité ou non stationnarité du paysage énergétique. Ainsi, les notions de multi-stabilité, de descente d'énergie et de convergence théorique posées au Chapitre 2 sont opérationnalisées dans un algorithme concret, permettant d'assurer à la fois la robustesse et l'adaptabilité du DSL en environnement réel. Cette approche permet également d'intervenir avec des mécanismes de contrôle (inhibition, clipping) pour orienter la dynamique vers des configurations optimales en fonction des besoins de l'application.

4.6. Extraction et Visualisation des Clusters Émergents

Au cours de la dynamique d'un **SCN** (Synergistic Connection Network), les pondérations $\omega_{i,j}$ évoluent et finissent souvent par révéler un **partitionnement** implicite (voir chapitres précédents). Une question cruciale consiste alors à **observer** et **visualiser** ces pondérations, de sorte à comprendre **comment** les clusters apparaissent et **quand** la structure du réseau se stabilise ou se modifie. Cette section (4.6) présente les principales **méthodes** pour :

216. **Suivre** l'évolution de la matrice $\{\omega_{i,j}\}$ en temps réel (4.6.1),
217. **Identifier** de façon automatique les clusters à partir de cette matrice (4.6.2),
218. **Analyser** les évolutions temporelles éventuelles de ces regroupements (4.6.3).

4.6.1. Méthodes d'Observation

La première étape pour extraire et comprendre les **clusters** consiste à **observer** la matrice $\{\omega_{i,j}\}$ et ses changements au fil du temps. Cette approche est particulièrement importante lorsque la dynamique est complexe (multiplicative, inhibition, etc.) ou qu'il y a un **apprentissage continu** (synergie $S(i,j)$ qui se met à jour).

4.6.1.1. Suivi de la Matrice $\{\omega_{i,j}\}$ au Fil du Temps

Dans la dynamique d'**auto-organisation** d'un **Synergistic Connection Network (SCN)**, la matrice $\{\omega_{i,j}(t)\}$ joue un rôle fondamental en reflétant l'état courant du réseau à chaque itération t . Les pondérations $\omega_{i,j}(t)$, ajustées selon une équation locale (ex. $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$), traduisent la **force de liaison** entre deux entités \mathcal{E}_i et \mathcal{E}_j . Observer leur évolution au fil du temps permet d'**analyser** la progression du SCN dans sa quête de **clusters** ou de **sous-réseaux** cohérents, tout en signalant d'éventuels déséquilibres (phénomènes d'oscillation, croissance excessive de liens, inertie). Cette démarche de *monitoring* revêt donc une valeur heuristique considérable pour comprendre si le réseau tend vers un **état stable** ou si, au contraire, il oscille ou se fragmente.

Le **suivi** de $\omega_{i,j}(t)$ consiste à archiver ou à sonder la matrice à intervalle régulier, afin de détecter comment les liens les plus significatifs se renforcent ou s'affaiblissent. Dans un SCN de taille modeste (n de l'ordre de quelques centaines), il est envisageable de **stocker** l'intégralité de la matrice $\{\omega_{i,j}(t)\}$ à chaque itération, pour reconstituer a posteriori l'évolution complète. On dispose ainsi d'un véritable "film" de la progression, ce qui facilite la mise en évidence de la formation progressive de **blocs** ou de "zones actives" où les pondérations montent en flèche. À l'inverse, dans un **réseau** plus volumineux (n allant jusqu'à plusieurs milliers ou dizaines de milliers), l'espace requis pour conserver la totalité des $O(n^2)$ liens à chaque étape s'avère prohibitif, et l'on recourt alors à un **échantillonnage** partiel (en ne conservant la matrice que toutes les 10 ou 50 itérations, ou en ne stockant qu'un sous-ensemble restreint de liens jugés pertinents). On peut aussi suivre des **indicateurs statistiques** (moyenne, variance, quantiles de $\omega_{i,j}$) pour dresser un portrait global de la dynamique, sans conserver chaque élément de la matrice.

Cette observation régulière renseigne non seulement sur la **formation** des **clusters** (par la concentration de liens forts à l'intérieur de certaines sous-parties), mais aussi sur la **déifferentiation** progressive des liens faibles et des liens forts. Les résultats typiques montrent qu'au départ, toutes les $\omega_{i,j}$ se situent dans un intervalle homogène (souvent proche de zéro si la matrice est initialisée à un faible bruit), puis la distribution se “bimodalise” : certains liens demeurent à des niveaux faibles, tandis que d'autres s'élèvent, révélant l'émergence de groupes denses. Parfois, un suivi fin permet de repérer des **oscillations** récurrentes dans la force de certains liens, symptôme d'une configuration non stabilisée ou d'un paramétrage problématique (par exemple, un η trop grand ou un τ trop faible). Dans ces cas, le *monitoring* constitue un outil de **diagnostic** : si le réseau ne converge pas ou oscille sur une large amplitude, on peut ajuster les paramètres ou vérifier que la fonction de synergie $S(i,j)$ ne présente pas de discontinuité excessive.

Le simple **observationnel** des pondérations ne suffit toutefois pas à extraire automatiquement les **communautés** ni à fournir une répartition claire de l'espace. C'est la raison pour laquelle, dans des sections ultérieures (telles que la section 4.6.2), l'on recourra à des méthodes plus **systématiques** d'extraction de clusters ou de détection de structures. Néanmoins, l'examen visuel ou statistique régulier de la matrice $\omega_{i,j}(t)$ garde tout son intérêt en phase de développement, de **test** ou de **recherche de bugs**, puisqu'il permet de diagnostiquer l'évolution globale du SCN et de repérer s'il se dirige vers un état stable, un cycle, ou un brouillard de valeurs moyennes peu discriminantes. Cette première étape de *monitoring* jette ainsi les bases d'une analyse plus poussée : sans elle, on risquerait de recourir à des algorithmes de détection de communautés ou de hiérarchisation sans s'apercevoir que le réseau ne s'est pas encore stabilisé, ou qu'une oscillation mine l'interprétabilité des clusters.

4.6.1.2. Visualisations en Heatmap, Courbes de Liaisons, ou Graphes Dynamiques

Le **Synergistic Connection Network (SCN)** met en jeu un ensemble de **pondérations** $\omega_{i,j}(t)$ dont l'évolution permet de comprendre la genèse de **clusters** ou de communautés internes. Lorsque le nombre d'entités n reste modéré, la simple consultation de la matrice $\omega(t)$ peut suffire, mais dès que l'on dépasse quelques centaines de nœuds, cette matrice devient difficile à interpréter si on se limite à un tableau numérique. Il est donc souvent essentiel d'adopter des **représentations visuelles** pour saisir le comportement du réseau, apprécier la montée ou la chute des liaisons $\omega_{i,j}(t)$ et identifier la formation progressive de blocs.

A. Heatmap (Matrice Colorée)

L'une des techniques les plus directes pour **visualiser** la matrice $\omega(t)$ consiste à en dresser une **carte colorée**. Les lignes représentent les entités \mathcal{E}_i et les colonnes \mathcal{E}_j , tandis que chaque cellule (i,j) est teintée selon la valeur de $\omega_{i,j}(t)$. Une **échelle de couleurs** (du clair au foncé, par exemple) traduit l'intensité du lien : plus la cellule est saturée, plus la pondération est élevée. Il est possible de générer cette carte pour différentes itérations t , de manière à constituer une séquence ou un “film” illustrant l'émergence ou l'extinction de régions de forte connectivité. Cette approche convient bien lorsque le **nombre** d'entités n demeure dans des limites raisonnables, comme quelques centaines de nœuds. On observe alors la formation de **blocs** compacts correspondant à des **clusters**, identifiables par des carrés colorés le long de la diagonale (ou, après réordonnement, dans d'autres zones de la matrice). Lorsque n devient plus grand, il est courant

de ne conserver qu'un **échantillonnage** ou de n'afficher que les liaisons dépassant un certain seuil. La heatmap reste alors un outil macroscopique, utile pour distinguer rapidement un état quasi homogène (aucun bloc net), une structure bipartite, ou plusieurs communautés distinctes.

B. Courbes de Liaisons

Une autre méthode consiste à **sélectionner** un sous-ensemble de liens $\omega_{i,j}(t)$ et à tracer leur valeur au cours du temps. Cette démarche permet un suivi plus **fin** d'un petit nombre de paires qui peuvent être jugées représentatives : certains liens présumés intra-cluster, d'autres inter-cluster, ou encore des liaisons ayant un rôle spécifique. On produit alors, pour chaque lien (i,j) choisi, une courbe indiquant $\omega_{i,j}(t)$ en fonction de t .

Cette représentation est particulièrement instructive pour **comprendre** la dynamique de montée ou de descente d'un lien donné. Un lien **intra-cluster** se met à croître rapidement jusqu'à un plateau, alors qu'un lien **inter-cluster** peut demeurer faible ou décroître progressivement. Parfois, on décèle des **oscillations** si $\omega_{i,j}(t)$ monte puis redescend en boucle, symptôme d'un déséquilibre dans la règle de mise à jour ou d'un paramètre mal calibré (taux d'apprentissage η trop grand, par exemple). Les *courbes de liaisons* offrent donc un moyen succinct de diagnostiquer la formation de blocs ou les difficultés de convergence, sans afficher la totalité de la matrice.

C. Graphes Dynamiques

Il est également possible de **représenter** le SCN comme un graphe dont les nœuds incarnent les entités $\{\mathcal{E}_i\}$, et où chaque arête (i,j) est dotée d'un poids $\omega_{i,j}(t)$. On peut alors faire évoluer l'épaisseur de l'arête ou sa couleur en fonction de $\omega_{i,j}(t)$, voire employer un algorithme de placement (layout) dynamique pour rapprocher les nœuds fortement reliés et éloigner ceux qui sont faiblement connectés.

Cette **visualisation** en "graphe dynamique" offre une vision très intuitive : on assiste littéralement au rassemblement progressif des nœuds en sous-ensembles denses. Un nœud peut être intégré simultanément à plusieurs blocs s'il dispose de liaisons importantes réparties, ou bien on voit apparaître des "communautés" distinctes dans différentes régions du plan si le layout *force-directed* est utilisé. Lorsque les pondérations $\omega_{i,j}(t)$ convergent, le graphe se fige en clusters stables ; si des oscillations surviennent, les liens changent périodiquement de force, et le graphe ne stabilise pas sa topologie.

Cette technique devient toutefois coûteuse pour de grands n . On doit le plus souvent filtrer les liaisons (ne garder que celles au-dessus d'un seuil, ou les k plus fortes par nœud) afin de limiter l'encombrement visuel. Dans un DSL de taille moyenne, l'animation du graphe dynamique constitue un atout pédagogique puissant pour visualiser la **coalescence** de communautés internes.

D. Intérêt pour l'Analyse de la Dynamique

Le choix de la **représentation** visuelle dépend du nombre d'entités, de la granularité qu'on veut observer et de la possibilité de filtrer ou non les liens. Les **heatmaps** indiquent la globalité de la matrice, les **courbes de liaisons** donnent un portrait plus détaillé de certaines paires, et les **graphes dynamiques** procurent une image structurale et esthétique du réseau en mutation. Toutes ces méthodes servent un même objectif : **appréhender** la progression du SCN, **déetecter** les moments critiques (croissance explosive de certaines pondérations, effondrement d'autres) et **repérer** des régimes d'oscillation ou de convergence.

Dans la suite (section 4.6.2), des outils plus formels d'**identification automatique** des clusters seront présentés. Les approches de visualisation décrites ici restent néanmoins un complément précieux pour la **compréhension** et la **validation** empirique de ce qui se produit dans un DSL. Elles confèrent une dimension exploratoire indispensable pour ajuster les hyperparamètres (η, τ) ou évaluer la pertinence de la fonction de synergie $S(i, j)$. Elles constituent en somme un pont entre l'analyse intuitive (repérer des blocs colorés, des arcs épais) et l'**extraction** algorithmique de structures.

4.6.2.1. Seuil sur $\omega_{i,j}$: si $\omega_{i,j} > \theta$, on considère l'arête comme “active” → composantes connexes

Le recours à un **seuil** θ appliqué aux pondérations $\omega_{i,j}$ représente l'une des stratégies les plus élémentaires pour extraire des **clusters** au sein d'un **Synergistic Connection Network (SCN)**. L'idée consiste à convertir la matrice $\{\omega_{i,j}\}$ en un graphe binaire en imposant une valeur critique θ : toute liaison (i, j) dont $\omega_{i,j}$ dépasse θ est qualifiée de “active” (et prend la valeur 1), tandis que les autres sont ramenées à 0. Cette binarisation aboutit à un réseau non pondéré où l'on peut aisément repérer des **composantes** connexes, chacune reflétant un **cluster** potentiel.

A. Définition du seuil et matrice binaire

La règle de transformation repose sur la relation

$$A_{i,j} = \begin{cases} 1 & \text{si } \omega_{i,j} > \theta, \\ 0 & \text{sinon.} \end{cases}$$

Ce passage d'une représentation pondérée $\omega_{i,j} \in \mathbb{R}^+$ à une **matrice** binaire **A** simplifie considérablement l'analyse. Dès qu'une liaison franchit la **barre** θ , elle est considérée comme un **arc actif** dans le graphe, sans plus distinguer les nuances de valeurs au-dessus de θ . Ce procédé reste très intuitif : seules les connexions jugées “fortes” ou “significatives” sont conservées, et on écarte la multitude de liens résiduels dont le poids, souvent faible, pourrait brouiller la lisibilité de la structure.

B. Rôle de θ et formation des clusters

Ce paramètre θ joue un rôle crucial, car il influe directement sur la **taille** et le **nombre** de composantes qu'on verra émerger dans le graphe binaire. Si l'on choisit un seuil trop élevé, la plupart des $\omega_{i,j}$ seront en deçà de θ ; on risque alors d'obtenir de nombreuses petites composantes, voire des nœuds totalement isolés. Au contraire, un seuil trop faible a pour effet de conserver trop de liaisons, transformant le graphe en un réseau quasi complet qui ne laisse apparaître aucune partition claire. La distribution des valeurs $\{\omega_{i,j}\}$ peut guider l'ajustement de θ : on peut par exemple fixer θ au percentile 80 ou 90 % des valeurs non-nulles, ou bien étudier l'histogramme pour repérer un coude permettant de trancher entre des liaisons fortes et des liaisons faibles.

C. Détection des composantes connexes

Une fois la matrice binaire établie, le **réseau** se conçoit comme un graphe simple où $\omega_{i,j} \in \{0,1\}$. L'extraction des **composantes** connexes devient alors un problème classique en théorie des graphes. On peut recourir à un algorithme de parcours en profondeur (*Depth-First Search*) ou en

largeur (*Breadth-First Search*) : pour chaque nœud, on visite les nœuds qui lui sont reliés par des arêtes actives, puis les nœuds reliés à ceux-ci, et ainsi de suite. L'ensemble de nœuds atteints de la sorte forme une **composante**, souvent qualifiée de cluster dans ce contexte d'auto-organisation. Il suffit alors de relancer cette procédure pour chaque nœud non encore visité afin de recenser la totalité des composantes. La **cohésion** interne d'une composante reflète la densité d'arêtes ayant dépassé le seuil θ . Ainsi, si θ est bien choisi, on obtient en général une partition reflétant très fidèlement les groupes de nœuds fortement reliés (intra-cluster), tout en isolant les liens inter-cluster jugés insuffisants.

D. Avantages et limites

La principale **force** de cette méthode réside dans sa **simplicité** et sa **transparence** : il suffit de couper tous les liens en deçà d'un certain poids. Les *clusters* détectés comme composantes connexes sont alors très lisibles, ne requièrent pas de calcul sophistiqué (un simple algorithme DFS ou BFS), et assurent une séparation nette entre les liens censés traduire une véritable affinité et ceux considérés comme parasites ou négligeables. Toutefois, cette binarisation implique qu'on ne nuance plus les liaisons supérieures au seuil : un lien à 0,95 est traité de la même manière qu'un lien à 0,55, alors que leur intensité réelle peut différer de manière significative. En outre, l'existence d'un **unique** paramètre θ rend la solution très sensible à ce choix : si ce seuil n'est pas adapté à la distribution réelle des $\omega_{i,j}$, on risque soit un trop grand nombre de micro-clusters, soit un unique bloc, ce qui n'est pas nécessairement conforme à la structure sous-jacente du SCN.

E. Conclusion

La méthode fondée sur un **seuil** θ appliqué aux pondérations $\omega_{i,j}$ constitue l'un des procédés les plus rudimentaires et rapides pour **identifier** des **clusters** dans un SCN. L'on retire tous les liens dont la valeur n'atteint pas ce niveau critique, puis on recherche les **composantes** connexes du graphe binaire obtenu. Cette approche révèle souvent la partition macroscopique que la dynamique d'auto-organisation cherchait à faire émerger, à condition de sélectionner judicieusement le seuil θ . Malgré l'absence de prise en compte des différences de poids au-dessus de θ , cette technique assure une lecture intuitive de l'organisation en groupes fortement soudés. Lorsque l'on désire un partitionnement plus *graduel*, il reste possible de se tourner vers des **méthodes pondérées** ou vers des indices de **modularité** (section 4.6.2.2) qui exploitent toute la richesse des valeurs $\omega_{i,j}$.

4.6.2.2. Mesures de Modularity (Inspirées des Graphes) ou “Community Detection” dans le SCN

Les **méthodes** décrites précédemment (en particulier le recours à un **seuil** θ pour binariser la matrice $\omega_{i,j}$) constituent des approches rapides pour **délimiter** des blocs dans un **Synergistic Connection Network** (SCN). Cependant, elles peinent parfois à exploiter la **graduation** subtile des valeurs $\omega_{i,j}$. Dès lors que l'on souhaite distinguer des liens à 0.95 de ceux à 0.55, il est naturel de se tourner vers des **outils** de *community detection* issus de la **théorie des graphes pondérés**. Cette famille d'algorithmes recourt souvent à une **mesure de modularité**, initialement conçue pour des graphes non pondérés, mais aisément étendue à des poids réels.

L'idée générale est de **maximiser** un indicateur, la **modularité** Q , qui évalue dans quelle mesure une partition en communautés s'écarte d'une configuration purement aléatoire. Appliquée à un

SCN, où $\{\omega_{i,j}\}$ traduit la **synergie** entre entités, la modularité fournit un critère solide pour **valider** la cohésion de chaque bloc interne tout en préservant l'hétérogénéité des liens.

A. Rappel de la Notion de Modularité dans un Graphe Pondéré

Dans un graphe non pondéré, la modularité (Q) se définit par le formalisme de Newman-Girvan. On compare la fraction de liens **intra-communautés** réelle d'une partition à la fraction **attendue** si la distribution des degrés était maintenue de façon aléatoire. En notation succincte, pour un graphe non pondéré, on a

$$Q = \frac{1}{2m} \sum_{i,j} \left[A_{i,j} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j),$$

où $A_{i,j}$ décrit la matrice d'adjacence binaire ($A_{i,j} = 1$ s'il existe une arête, 0 sinon), m est le nombre total d'arêtes, et k_i le degré du nœud i . Le terme $\delta(c_i, c_j)$ vaut 1 si les nœuds i et j appartiennent à la **même** communauté, 0 dans le cas contraire.

Dans le cadre **pondéré** pertinent pour un **SCN**, on remplace $\{A_{i,j}\}$ par $\{\omega_{i,j}\}$, et on définit le degré pondéré comme $d_i = \sum_j \omega_{i,j}$. La somme totale de poids W s'identifie alors à la moitié de $\sum_{i,j} \omega_{i,j}$ pour un graphe non orienté. La **modularité** prend la forme

$$Q = \frac{1}{2W} \sum_{i,j} \left[\omega_{i,j} - \frac{d_i d_j}{2W} \right] \delta(c_i, c_j).$$

Le raisonnement demeure identique : on quantifie la somme des poids “intra-communauté” qui excède la somme que l'on obtiendrait dans un régime aléatoire respectant les degrés $\{d_i\}$.

B. Application dans un Synergistic Connection Network

Dans un **SCN**, chaque entité \mathcal{E}_i possède une somme pondérée $d_i = \sum_j \omega_{i,j}$. La totalité des poids est $W = 1/2 \sum_{i,j} \omega_{i,j}$. Considérer $\omega_{i,j}$ comme la matrice de poids d'un *graphe pondéré* rend possible l'emploi direct des algorithmes de *community detection* tels que Louvain, Leiden ou d'autres techniques spectrales ou basées sur la betweenness.

Leur but est de **maximiser** la modularité Q . Une “communauté” au sens modulaire se conçoit donc comme un ensemble de nœuds (entités) dont les liens internes $\omega_{i,j}$ sont significativement supérieurs à ce que proposerait un modèle d'attachement aléatoire. Intuitivement, si des entités forment un cluster dans le SCN, on doit y constater un renforcement local $\{\omega_{i,j}\}$, ce qui se traduira par un niveau élevé de modularité intra-communauté.

C. Avantages pour le SCN

Comparer cette méthode de *community detection* à la binarisation via un **seuil** θ souligne deux atouts majeurs. D'abord, la modularité prend **pleinement** en compte les variations de poids entre différents liens, et ne les réduit pas à un simple 0/1. Ensuite, de nombreux **algorithmes** éprouvés – comme Louvain ou Leiden – sont disponibles dans des bibliothèques spécialisées (NetworkX, igraph, etc.), avec une **implémentation** optimisée même pour des graphes de grande taille.

Si l'on souhaite confirmer la cohérence d'un découpage donné (par exemple, si on a une partition candidate trouvée par un autre moyen), on peut en calculer la **modularité** Q . Plus la valeur de Q s'approche de 1, plus la structuration en communautés est considérée “prononcée”. Dans les cas extrêmes, un Q très proche de 0 ou négatif indique une absence de véritables communautés au sens de la pondération interne.

D. Limites et Alternatives

La **modularité** est victime d'un phénomène connu sous le nom de “résolution limit”, qui tend à faire émerger des communautés parfois plus grandes que la “réalité” de terrain ou, au contraire, à rater certains clusters de taille réduite. D'autres métriques (comme la *partition density*, la *surprise*, ou des méthodes de *infomap*) peuvent pallier partiellement ces écueils, mais la modularité demeure un **standard** pour la *community detection* dans de nombreux scénarios.

En sus, on peut envisager des **méthodes** alternatives pour repérer des blocs pondérés : par exemple, un **algorithme spectral** construit la Laplacienne du graphe $\mathbf{L} = \mathbf{D} - \mathbf{W}$, où \mathbf{W} est la matrice $\{\omega_{i,j}\}$ et \mathbf{D} la matrice diagonale des degrés pondérés. Les *vecteurs propres* associés aux plus petites valeurs propres de \mathbf{L} permettent d'identifier, via un k -means, des ensembles fortement connectés. D'autres algorithmes, comme *Girvan–Newman* adapté aux poids, suppriment itérativement les liens de forte betweenness jusqu'à l'apparition de groupes.

E. Conclusion

Les **mesures de modularité**, et plus largement les approches de **community detection** pondérées, constituent une avancée majeure pour l'**identification** de **clusters** dans un SCN. Contrairement à la simple binarisation par θ , on conserve la **graduation** des pondérations $\omega_{i,j}$. Cette nuance se révèle décisive pour des scénarios où la distance entre 0.95 et 0.60 importe, et où l'on ne veut pas se contenter d'un coup de ciseau tranchant. Grâce à des algorithmes tels que Louvain ou Leiden, on peut tirer parti de la **structuration** que la dynamique du DSL a créée, et faire émerger une partition en communautés reflétant véritablement la répartition des poids dans le réseau. L'étude de la modularité Q donne alors un **indicateur** quantitatif du caractère “naturel” ou “forcé” de la segmentation, tout en s'appuyant sur la logique de la pondération globale $\{\omega_{i,j}\}$.

4.6.2.3. Lien avec le Clustering “Offline” (**k-means**, **DBSCAN**), mais ici c'est issu d'une Dynamique Interne

Le **clustering** en tant que méthode de segmentation de données se rencontre fréquemment sous forme d'algorithmes “offline” tels que **k-means**, **DBSCAN** ou encore l'approche **agglomérative**. Ces techniques s'appliquent habituellement à un **ensemble** de points $\{x_i\} \subset \mathbb{R}^d$, soumis à une distance explicite (euclidienne, cosinus, densité, etc.), dans le but de **découper** l'ensemble en blocs suffisamment cohérents pour un critère global.

Dans un **Synergistic Connection Network** (SCN), le repérage des **clusters** repose sur une **dynamique interne** : la mise à jour itérative de $\omega_{i,j}(t)$ construit progressivement une structure de poids, révélant des **liaisons** fortes entre entités $\mathcal{E}_i, \mathcal{E}_j$. Il est alors légitime de se demander dans quelle mesure ces approches de clustering classiques recoupent, ou se différencient, des blocs obtenus par la **lecture** de la matrice $\{\omega_{i,j}\}$ (comme décrit en 4.6.2.1 ou 4.6.2.2).

A. Rappel : k-means, DBSCAN et Clustering Offline

Lorsque l'on applique **k-means** à un ensemble de points $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$, l'algorithme procède par minimisation d'une fonction d'énergie, généralement la somme intra-cluster des distances au centroïde. La taille k de la partition est fixée au préalable, et le résultat final dépend de la position initiale des centroïdes et de la distribution des données. **DBSCAN**, à l'inverse, se fonde sur une notion de **densité** (rayon ε et minPts) afin de regrouper les points "denses" et laisser les autres comme du "bruit".

Ces algorithmes sont dits "**offline**" : un **lot** de données est fourni, et l'on cherche le partitionnement optimal en se basant sur une **distance** statique. Toute correction (changement de paramètres) exige de relancer l'algorithme.

B. Dynamique Interne des Pondérations dans un SCN

Dans un **SCN**, au contraire, les entités \mathcal{E}_i ne sont pas placées dans un espace fixe ; elles se relient par des pondérations $\omega_{i,j}(t)$ qui évoluent au fil d'une **règle** de mise à jour dépendant de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Le SCN ne procède pas par minimisation explicite d'un critère global comme k-means ; il s'appuie plutôt sur un **processus local** qui renforce ou atténue chaque liaison. Les *clusters* émergent alors naturellement, et l'on peut distinguer :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)].$$

Les *clusters* sont donc un **résultat de l'auto-organisation** et non l'objet d'un algorithme de partition. Il est possible de considérer qu'une fois la matrice $\{\omega_{i,j}\}$ stabilisée (ou lue à un instant donné), on détient un "**graph pondéré**" dont on peut extraire des communautés via différentes méthodes (seuil, modularité, etc.).

C. Possibilité de Relier l'Approche "Offline" et "Online"

Un **DSL** présente d'autres subtilités : l'on peut, de temps à autre, prélever la **matrice** $\{\omega_{i,j}(t)\}$ pour en faire un **traitement** "offline". Par exemple, on peut procéder ainsi :

219. Considérer les vecteurs $\boldsymbol{\omega}_i \in \mathbb{R}^n$ définis par

$$\boldsymbol{\omega}_i = (\omega_{i,1}, \omega_{i,2}, \dots, \omega_{i,n}).$$

On obtient ainsi un portrait complet de la manière dont la **synergie** s'exprime entre \mathcal{E}_i et l'ensemble des autres entités.

220. Appliquer un **clustering** typique (ex. k-means) sur l'ensemble $\{\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_n\}$. Les entités qui possèdent un "profil de connexion" similaire (pondérations internes et externes proches) se retrouvent alors dans la même classe.

Cette technique revient à considérer que la *distance* entre deux entités $\mathcal{E}_i, \mathcal{E}_j$ peut être mesurée via une norme $\|\boldsymbol{\omega}_i - \boldsymbol{\omega}_j\|$. Par ce biais, l'on fait un pont entre la **dynamique interne** (qui a fait croître ou décroître les $\omega_{i,j}$) et un **algorithme** de segmentation standard (k-means, DBSCAN) appliqué "après coup".

D. Intérêt et Différences

Il existe un **intérêt** certain à comparer une **partition** issue de cette démarche (ou via la modularité sur le graph pondéré) à celle qu'on obtiendrait en prenant directement les **données brutes** de \mathcal{E}_i (ex. leurs embeddings ou leurs attributs) et en appliquant k-means/DBSCAN. Si les deux partitions coïncident, on peut conclure que la **fonction** de synergie $S(i, j)$ et la **dynamique** du SCN n'ont fait que reproduire la structure spatiale originale. S'il y a un écart notable, cela indique que la logique interne du SCN (éventuellement imprégnée de symbolique, ou d'une synergie non triviale) a façonné une structuration différente, possiblement plus riche.

La différence essentielle est que dans le **clustering offline**, on cherche à **minimiser** explicitement un critère global (somme des distances intra-cluster, densité, etc.), tandis qu'un **SCN** exécute un **processus local d'auto-organisation**. Ce processus peut faire intervenir différents paramètres (symboliques ou sub-symboliques), permettant d'autres formes de “convergence” qu'une simple structure euclidienne dans \mathbb{R}^d .

Ainsi, si un utilisateur se demande : “Ne pourrais-je pas simplement appliquer k-means (ou DBSCAN) sur mes données plutôt que de passer par la mise à jour des $\omega_{i,j}$?” La réponse tient en la **finalité** : un **DSL** permet de gérer un flux évolutif, une synergie multimodale, ou un couplage symbolique–sub-symbolique. Les **clusters** ne sont pas seulement un partitionnement statique ; ils résultent d'un **processus** vivant, sensible aux interactions internes (renforcements/désactivations).

E. Conclusion

Il existe un **parallèle** entre la volonté de décomposer un ensemble de points en **clusters** (via k-means, DBSCAN) et la recherche de **communautés** à l'intérieur d'un **SCN**. Les deux aboutissent à des **groupes** d'entités. Cependant, dans un clustering traditionnel “offline”, on part d'un dataset statique et on applique un **critère** fixé (variance, densité, etc.), alors que dans un SCN, les **liens** $\omega_{i,j}$ proviennent d'une **dynamique** auto-organisée. Les algorithmes de partition basés sur $\{\omega_{i,j}\}$ (seuil, modularité, community detection) tirent avantage d'un “réseau pondéré” engendré par la synergie, et non d'une distance spatiale imposée a priori.

Un **SCN** peut donc être vu comme une solution “**online**” ou “**dynamique**”, là où k-means/DBSCAN constituent des méthodes “**offline**” fixes. Les deux approches demeurent complémentaires : il est possible de confronter les **résultats** (partitions) pour mieux comprendre l'agencement de ses données, ou même d'adopter un schéma hybride (ex. utiliser k-means pour initialiser ω , puis laisser la règle d'auto-organisation opérer). Cette **interaction** entre la dynamique interne du SCN et les techniques de clustering classiques illustre la souplesse du DSL, capable de revisiter ou d'étendre la notion même de partition au sein d'un réseau adaptatif.

4.6.3. Évolutions dans le Temps

Une fois les **clusters** identifiés (4.6.2), il est également pertinent de **suivre** leur **évolution** au fil des itérations ou des mises à jour (particulièrement si $S(i, j)$ n'est pas figé). Dans un **SCN** (Synergistic Connection Network), les changements de pondérations $\omega_{i,j}$ peuvent amener un cluster à **apparaître**, à **fusionner** avec un autre, ou encore à se **scinder** en deux ou plusieurs sous-groupes. L'observation de ces **dynamiques** se révèle cruciale pour comprendre la façon dont le réseau

s’auto-organise ou se réorganise lorsque les entités, les synergies ou les paramètres du modèle varient.

4.6.3.1. Les Clusters Peuvent Apparaître, Fusionner, se Scinder

Les **clusters** formés au sein d’un **Synergistic Connection Network (SCN)** ne constituent pas nécessairement des entités fixes et immuables. Sous l’effet de la **dynamique** de mise à jour des pondérations $\omega_{i,j}(t)$ et de l’évolution potentielle de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (dans un scénario de données ou de règles changeantes), ces regroupements peuvent se **créer**, se **réunifier** ou se **dissocier** au fil du temps.

A. Apparition de nouveaux clusters

Un groupe restreint d’entités, initialement dispersées au sein du SCN, est susceptible de voir leurs liaisons $\omega_{i,j}$ progresser de manière conjointe. Les liens intra-groupe s’élèvent peu à peu, se stabilisant au-dessus d’un certain **niveau**. À mesure que ces pondérations franchissent un seuil critique (éventuellement implicite) ou qu’elles surpassent nettement les liens externes, ce **groupe** se distingue nettement du reste du réseau. Un nouveau **cluster** “apparaît” alors, témoignant de la coalescence progressive de ses membres autour d’affinités fortes. Sur le plan mathématique, si les poids $\{\omega_{i,j}\}$ pour (i, j) dans un certain sous-ensemble \mathcal{C} atteignent des valeurs significatives, alors les entités $\mathcal{E}_i \in \mathcal{C}$ forment un bloc cohésif. Il suffit par exemple qu’à l’itération t les liaisons $\omega_{i,j}(t)$ dépassent successivement un seuil θ , rendant évident le fait que $\{i, j\}$ forment désormais un **noyau** ou qu’un triplet se renforce.

B. Fusion de clusters préexistants

Il arrive également que deux ou plusieurs **communautés** déjà consolidées se rapprochent l’une de l’autre. Dans ce cas, un sous-ensemble d’entités appartenant à deux blocs différents se met à voir ses pondérations inter-blocs croître de manière notable. Les liens $\omega_{i,j}$ entre ces deux agrégats, autrefois faibles, s’intensifient à la suite de modifications dans la **synergie** $S(i, j)$.

Lorsqu’un nombre suffisant de liaisons entre ces groupes dépasse un niveau appréciable, le système peut converger vers une situation où les deux ensembles précédents ne forment plus qu’un seul **cluster** fusionné. Sur le plan **graphique**, on interprète la scène comme deux composantes qui se **rejoignent**, des arêtes inter-blocs qui se renforcent, et une nouvelle macro-communauté qui émerge.

C. Scission d’un cluster en plusieurs

Inversement, un **cluster** jusque-là stable peut se **scinder** si, dans sa structure interne, certaines liaisons se mettent à décroître ou si un sous-groupe d’entités voit ses pondérations internes s’intensifier au détriment des liens avec le reste. Le sous-groupe finit par se “détacher” du noyau originel.

Sur un plan formel, on suppose qu’un cluster \mathcal{C} se fragmente en $\mathcal{C}_1 \cup \mathcal{C}_2 = \mathcal{C}$ parce que les valeurs $\{\omega_{i,j}\}_{i \in \mathcal{C}_1, j \in \mathcal{C}_2}$ baissent progressivement, tandis que les pondérations $\{\omega_{i,j}\}_{(i,j) \in \mathcal{C}_1^2}$ demeurent

élevées. La discontinuité se révèle lorsqu'on observe la formation d'un véritable sous-cluster \mathcal{C}_1 qui ne conserve presque plus d'attaches fortes avec \mathcal{C}_2 .

D. Causes et conditions de ces évolutions

Les **transformations** qui mènent à l'apparition, la fusion ou la scission de clusters peuvent s'expliquer par plusieurs facteurs. L'évolution de la **synergie** $S(i, j)$ constitue une première cause : dans un **DSL** multimodal ou symbolique, un changement d'environnement ou la mise à jour des caractéristiques d'une entité entraînent une redistribution de valeurs de similarité ou de compatibilité, modifiant les $\{\omega_{i,j}\}$. Les **paramètres** internes (le taux d'apprentissage η , le coefficient τ , ou encore l'ajout d'une inhibition latérale γ) peuvent aussi favoriser la ségrégation en petits blocs ou, au contraire, la coalescence en ensembles plus vastes. Enfin, des **perturbations** volontaires (recuit, injection de bruit) peuvent redéfinir la topologie du SCN et engendrer de nouvelles partitions.

E. Observation empirique et diagnostic

Sur le plan **pratique**, on suit la matrice $\{\omega_{i,j}(t)\}$ (section 4.6.1) ou on en extrait des communautés (section 4.6.2). À chaque itération ou après un certain nombre de pas, on compare la partition courante à la partition précédente, ce qui met en évidence l'apparition ou la fusion de blocs, la fragmentation d'une communauté, etc. Des mesures telles que la **NMI** (Normalized Mutual Information) ou l'indice de Rand offrent un moyen quantitatif de comparer deux partitions successives. On peut ainsi repérer les instants où un cluster a “disparu” (scission) ou où deux communautés se sont unies.

Cette **observation** permet également de comprendre si le **SCN** se stabilise ou maintient un régime oscillatoire. Lorsqu'un cluster apparaît mais se délite quelques itérations plus tard, l'auto-organisation demeure dans un état transitoire, suggérant qu'il n'existe pas encore d'équilibre.

Conclusion (4.6.3.1)

Les **clusters** au sein d'un **Synergistic Connection Network** ne sont pas fixes, mais reflètent la **dynamique** interne de la mise à jour $\Delta\omega_{i,j}$. Ils peuvent naître de la consolidation de quelques entités jusque-là dispersées, se **fusionner** en un bloc unique lorsque leurs sous-ensembles développent des liaisons suffisamment fortes, ou se **scinder** lorsque certaines liaisons internes s'affaiblissent. Cette plasticité témoigne de la nature adaptative d'un **DSL**, exposé à des modifications de la synergie ou des paramètres du réseau. Les sections ultérieures poursuivront cette analyse en considérant comment suivre et “tracker” ces variations de communautés (section 4.6.3.2), afin de les consigner ou de les exploiter dans des scénarios applicatifs.

4.6.3.2. Notation d'un “Cluster Tracking” : Indexer les Entités qui Changent de Communauté

La dynamique d'un **Synergistic Connection Network** (SCN) peut conduire à des **transformations** répétées de la structure de **clusters** (ou communautés) : un groupe d'entités peut naître, deux communautés peuvent fusionner ou, au contraire, un bloc jusqu'alors soudé se scinder en plusieurs parties. Dans l'optique d'analyser un tel système en évolution, il est souvent crucial d'exploiter un mécanisme de **cluster tracking**, c'est-à-dire un dispositif consignant dans quelle

communauté chaque entité se situe à divers instants. Ce mécanisme autorise un suivi temporel des recompositions et, en cas de changements, il rend possible l'évaluation du degré d'instabilité ou de transition dans le SCN.

A. Principe Général du “Cluster Tracking”

Il est fréquent de ne pas recalculer la partition en communautés à chaque itération t , surtout si la dynamique est potentiellement rapide ou coûteuse. On définit plutôt une suite de **paliers** $\{t_0, t_1, \dots, t_K\}$ où l'on exécute une procédure d'**extraction** de clusters (comme décrit en 4.6.2). Chaque entité \mathcal{E}_i reçoit un label de communauté $\text{clusterID}_i(t_k)$ pour chaque instant de référence t_k . La confrontation des partitions successives donne alors des informations précises : si une entité \mathcal{E}_i change de label, on note que cette entité migre d'une communauté à une autre ; si un cluster se scinde, on repère dans la partition au temps t_{k+1} que le groupe qui s'appelait “C1” au temps t_k est maintenant réparti dans deux nouveaux labels.

B. Comparaison Formelle Entre Deux Partitions

Pour comparer la partition $\mathcal{P}(t_k)$ à la partition $\mathcal{P}(t_{k+1})$, il est souvent utile de recourir à des **indices** standard tels que l'**indice de Rand** ou la **NMI** (Normalized Mutual Information). Ces mesures quantifient la similarité entre deux partitions, de sorte qu'un indice élevé indique une partition presque inchangée alors qu'un indice bas reflète une fragmentation ou une fusion importante. Sur le plan **algorithmique**, on doit parfois résoudre un **matching** entre les labels des deux partitions, puisque le cluster 3 dans $\mathcal{P}(t_k)$ peut correspondre majoritairement au cluster 2 dans $\mathcal{P}(t_{k+1})$. Il se peut qu'un algorithme de *Hungarian matching* (ou un équivalent) soit utilisé pour trouver un mapping optimal, minimisant les désaccords entre les deux partitions.

C. Changements Possibles et Suivi dans le Temps

Les changements majeurs identifiés sont :

- **Apparition d'un nouveau cluster** : un sous-ensemble d'entités jusqu'alors éparpillées ou relevant d'autres communautés se regroupent en un bloc cohésif.
- **Fusion de clusters** : deux blocs établis voient leurs liens inter-communautés $\{\omega_{i,j}\}$ croître au point de se confondre en un unique cluster.
- **Scission** : un bloc relativement stable se fragmente en deux sous-blocs distincts, dont les liens internes respectifs se renforcent et se détachent de l'ensemble initial.
- **Migration partielle** : quelques entités se déplacent d'une communauté à une autre, sans que la communauté source ni la communauté cible ne disparaissent ou ne fusionnent.

La connaissance de ces changements successifs permet, par exemple, de constituer un **journal** retracant l'historique de l'évolution du SCN. Cela se conçoit sous la forme d'une table où l'on indique à quel pas de temps \mathcal{E}_i quitte un label pour un autre, ou sous forme de diagrammes qui représentent visuellement la “trajectoire” de chaque bloc dans un plan temporel, un peu à la manière d'un diagramme de Sankey.

D. Avantages et Limites

Le *cluster tracking* éclaire la **stabilité** du SCN en révélant quelles entités demeurent ancrées dans le même cluster sur une longue période et quelles entités se montrent plus volatiles. Cela facilite

un diagnostic de la **convergence** : si, itération après itération, la partition fluctue très peu, on en déduit une stabilisation. Inversement, si une proportion non négligeable de nœuds changent de label à chaque palier, on soupçonne soit des oscillations internes, soit une évolution rapide de la synergie $\{\omega_{i,j}\}$.

Sur le plan computationnel, le *cluster tracking* exige d'**exécuter** ou de **répéter** régulièrement un algorithme de détection de communautés (ou un seuil, ou une modularité). Cette opération ne doit pas nécessairement être effectuée à chaque itération : on peut opter pour un pas plus lâche, réduisant la charge. Il subsiste un flou potentiel lorsqu'une communauté se scinde en plusieurs morceaux de taille comparable : les algorithmes de matching entre labels peuvent alors décider arbitrairement d'un label pour la majorité, et on perd une correspondance limpide pour l'autre fragment.

Conclusion (4.6.3.2)

La mise en place d'un **mécanisme** de “*cluster tracking*” joue un rôle capital dans un **DSL évolutif**, où la matrice $\omega_{i,j}(t)$ et par conséquent la structure de groupes subissent de potentielles modifications à chaque étape. En consignant la partition $\mathcal{P}(t_k)$ à divers paliers, puis en comparant ces partitions entre elles, on identifie précisément les **migrations** d'entités d'un cluster à l'autre, les **fusions** et **scissions** de communautés. Cette démarche outrepasse la simple photographie à un instant donné : elle offre une vision **dynamique** et **longitudinale** du SCN, révélant non seulement le “qui est avec qui” final, mais aussi la façon dont on y parvient et les délais ou perturbations éventuels. Elle sert ainsi à détecter et à **documenter** le *turn-over* communautaire, informant de la robustesse, de la stabilité et de l'adaptabilité du réseau pondéré.

4.7. Exemples Concrets et Études de Cas

4.7.1. Petite Simulation Numérique

- 4.7.1.1. 10 entités sub-symboliques + 5 entités logiques, synergie artificielle ou semi-aléatoire.
- 4.7.1.2. Mise en œuvre de la dynamique $\omega_{i,j}(t+1)$. Courbes de convergence, émergence de 2 ou 3 clusters.
- 4.7.1.3. Discussion des paramètres : η , τ , γ (inhibition), etc.

4.7.2. Cas Robotique ou Multi-Agent

- 4.7.2.1. Rappel : flottes de robots, ex. 5 robots simulés, synergie = réussite conjointe de tâches.
- 4.7.2.2. Observations : un cluster se forme entre 3 robots complémentaires, les 2 autres se marginalisent ou se lient plus faiblement.
- 4.7.2.3. Visualisation dynamique (cycles, stabilisation).

4.7.3. Liens Pratiques

- 4.7.3.1. Ce qu'on pourrait coder en Python/C++ pour reproduire cette dynamique en quelques centaines de lignes.
- 4.7.3.2. Aperçu du “SCN Dashboard” : par ex. un outil de monitoring des $\omega_{i,j}$ pour détecter la formation des clusters en temps réel.

4.7. Exemples Concrets et Études de Cas

Les sections précédentes (4.5, 4.6) ont présenté la dynamique d'un **SCN** (Synergistic Connection Network) et les moyens d'observer/extraire les clusters. Pour aller plus loin, il est souvent instructif de mettre en pratique ces notions sur des **cas concrets** ou des **simulations**. Ce chapitre 4.7 illustre ainsi plusieurs **exemples** de configurations (sous forme de petites études de cas) où l'on voit clairement comment la synergie (artificielle, semi-aléatoire ou adaptée à une tâche) déclenche l'auto-organisation du réseau.

4.7.1. Petite Simulation Numérique

- 4.7.1.1. 10 entités sub-symboliques + 5 entités logiques, synergie artificielle ou semi-aléatoire.
- 4.7.1.2. Mise en œuvre de la dynamique $\omega_{i,j}(t+1)$. Courbes de convergence, émergence de 2 ou 3 clusters.
- 4.7.1.3. Discussion des paramètres : η , τ , γ (inhibition), etc.

4.7.2. Cas Robotique ou Multi-Agent

- 4.7.2.1. Rappel : flottes de robots, ex. 5 robots simulés, synergie = réussite conjointe de tâches.
- 4.7.2.2. Observations : un cluster se forme entre 3 robots complémentaires, les 2 autres se marginalisent ou se lient plus faiblement.
- 4.7.2.3. Visualisation dynamique (cycles, stabilisation).

4.7.3. Liens Pratiques

- 4.7.3.1. Ce qu'on pourrait coder en Python/C++ pour reproduire cette dynamique en quelques centaines de lignes.
- 4.7.3.2. Aperçu du “SCN Dashboard” : par ex. un outil de monitoring des $\omega_{i,j}$ pour détecter la formation des clusters en temps réel.

4.7.1.1. 10 Entités Sub-Symboliques + 5 Entités Logiques, Synergie Artificielle ou Semi-Aléatoire

Il peut être instructif, pour illustrer les mécanismes d'un **Deep Synergy Learning (DSL)** à taille **modeste**, de constituer un petit jeu de données mélangeant plusieurs entités **sub-symboliques** et quelques entités **symboliques**. L'objectif est de mettre en évidence la **cohérence** qui peut émerger entre groupes de vecteurs (sous-cluster A ou B) et blocs logiques (règles, axiomes) au terme de la dynamique de mise à jour $\omega_{i,j}(t)$.

A. Configuration des Entités

L'expérimentation suppose la présence de dix entités **sub-symboliques** $\{\mathcal{E}_1, \dots, \mathcal{E}_{10}\}$, chacune décrite par un **embedding** en dimension $d = 5$. La génération de ces vecteurs peut s'effectuer selon un **tirage** aléatoire contrôlé, en faisant apparaître deux **sous-groupes** (A et B) plus proches intérieurement (typiquement, leurs centres de gravité se distinguent). L'on obtient ainsi une répartition semi-artificielle, de sorte que la *similarité* sub–sub prenne des valeurs significativement plus élevées (par exemple, autour de 0.7) dans chacun des deux sous-groupes, tandis qu'elle tombe plutôt autour de 0.2 pour des paires inter-groupes.

On introduit en complément cinq entités **logiques** $\{\mathcal{L}_1, \dots, \mathcal{L}_5\}$. Chacune incarne un **ensemble de règles** ou un **bloc de prédictats**, susceptibles d'être plus ou moins compatibles avec les caractéristiques de A ou B. On peut, par exemple, faire en sorte que $\{\mathcal{L}_1, \mathcal{L}_2\}$ correspondent à des axiomes particulièrement alignés sur le sous-cluster A, alors que $\{\mathcal{L}_3, \mathcal{L}_4\}$ trouvent une plus grande cohérence avec B, tandis que \mathcal{L}_5 demeure relativement **neutre**.

B. Génération de la Synergie

Le calcul de la synergie $S(i, j)$ peut se structurer selon une logique **hybride** (chapitre 3), c'est-à-dire :

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^5$ représentent des embeddings sub-symboliques et $\{R_i, R_j\}$ les blocs logiques associés. Pour deux entités sub-symboliques, on se limite à la **similarité sub-sub** (par exemple un cosinus normalisé ou un noyau gaussien), alors que pour deux entités **symboliques**, un **score binaire** (0 ou 1) ou trinitaire (0, 0.5, 1) quantifie la **compatibilité** des règles. Enfin, pour un couple sub-sym, l'on convient d'une formule adaptée (voir chap. 3.5) ou d'un paramètre β modérant l'influence d'un vecteur sub-symbolique sur un bloc logique.

La distribution semi-aléatoire est paramétrée de telle sorte que, dans chaque sous-groupe sub-symbolique A ou B, on obtienne des similarités sub-sub moyennes de 0.6–0.8, tandis que les paires inter-groupes se maintiennent à des valeurs plus faibles (0.1–0.3). Les entités logiques $\mathcal{L}_1, \mathcal{L}_2$ pourront ainsi être fortement cohérentes avec A ($S_{\text{sym}}(\mathcal{L}_1, \mathcal{E}_i) \approx 1$ si $\mathcal{E}_i \in A$), et moyennement ou faiblement cohérentes avec B, etc.

C. Lecture Attendue du SCN

Lorsque la **dynamique** de pondérations $\omega_{i,j}(t)$ se déroule (au sens de la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$), il est prévisible de voir émerger au moins deux **communautés** majeures : la première associant les entités de **sous-cluster A** (car elles sont proches sub-symboliquement) et les entités logiques compatibles $\{\mathcal{L}_1, \mathcal{L}_2\}$, la seconde regroupant les entités de **sous-cluster B** liées à $\{\mathcal{L}_3, \mathcal{L}_4\}$. La cinquième entité logique \mathcal{L}_5 , dotée d'un profil plus neutre, pourra :

221. Rejoindre l'un des deux groupes si sa compatibilité symbolique penche assez nettement dans ce sens,
222. Rester relativement isolée dans un micro-bloc, ou
223. Établir un pont faiblement pondéré entre les deux macros-clusters.

L'observation de la matrice $\{\omega_{i,j}(t)\}$ devrait mettre en évidence deux **blocs** de fortes pondérations internes, reflétant les deux sous-groupes sub-sym coalisés, et laisser à part une zone de liaisons plus ténues avec \mathcal{L}_5 .

D. Reproductibilité et Variations

Cette configuration, étant essentiellement **semi-artificielle**, s'apprête bien à des expérimentations comparatives. En changeant la graine aléatoire qui distribue les embeddings sub-symboliques et en modifiant légèrement la logique de $\mathcal{L}_1, \dots, \mathcal{L}_5$, on peut évaluer :

- La robustesse de la formation des mêmes blocs A– $\{\mathcal{L}_1, \mathcal{L}_2\}$ et B– $\{\mathcal{L}_3, \mathcal{L}_4\}$,
- L'existence de configurations multiples (multi-stabilité), dans lesquelles \mathcal{L}_5 intègre tel ou tel bloc en fonction des initialisations $\{\omega_{i,j}(0)\}$.

E. Extension et Généralisation

Ce **mini-prototype** avec dix entités sub-symboliques et cinq entités logiques peut aisément s'agrandir pour mieux faire ressortir le **phénomène**. En portant le nombre de sub-symboliques à 30 ou 50, réparties en deux ou trois sous-groupes artificiels, on intensifie la dimension “clustering sub-sub”. Simultanément, on peut augmenter le nombre d'entités logiques en leur associant diverses compatibilités $\{R_i\}$ afin d'obtenir un scénario plus riche, tout en restant dans une échelle

accessible au *monitoring* (tracé de heatmaps, courbes, etc.). Les étapes subséquentes (sections 4.7.1.2, 4.7.1.3) détailleront comment **visualiser** l'évolution des $\omega_{i,j}(t)$ et **régler** des paramètres tels que η et τ pour mettre en évidence la trajectoire menant à la formation de **clusters** logiquement et statistiquement cohérents.

Conclusion (4.7.1.1)

La configuration consistant en **10 entités sub-symboliques** (réparties en deux mini-groupes) et **5 entités logiques** (cohérentes avec l'un ou l'autre sous-groupe) constitue un **exemple minimal** permettant d'**illustrer** la synergie sub–sub, sym–sym et sub–sym. En **comparant** la matrice $\{\omega_{i,j}(t)\}$ avant et après la convergence, on s'attend à voir au moins deux grands **clusters** stabilisés, chacun fédérant un bloc sub-symbolique et les blocs logiques compatibles. Cette **simulation** démontre ainsi la manière dont un **SCN** parvient à un **regroupement** cohésif, appuyé sur une double logique sub-symbolique et symbolique, et fournit un terrain de jeu pour étudier plus avant la dynamique, les paramétrages, et la possibilité de comportements multi-stables.

4.7.1.2. Mise en Œuvre de la Dynamique $\omega_{i,j}(t + 1)$. Courbes de Convergence, Émergence de 2 ou 3 Clusters

La configuration décrite en (4.7.1.1) – réunissant dix entités **sub-symboliques** et cinq entités **logiques** avec une **synergie** semi-aléatoire – constitue un cadre propice pour illustrer la **dynamique** interne d'un **Synergistic Connection Network (SCN)**. Après avoir fixé la structure des entités (embeddings, règles) et leur **synergie** (sub–sub, sym–sym, sub–sym), la prochaine étape consiste à **exécuter** l'algorithme de mise à jour $\omega_{i,j}(t + 1)$. L'observation de la **convergence** (ou non-convergence) et la **visualisation** des pondérations au fil des itérations révèlent la formation potentielle de deux ou trois **clusters** stabilisés.

A. Choix de la Règle de Mise à Jour

Il s'agit ici de mettre en œuvre la **formulation additive**, telle que :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage**, τ la **décroissance** et $S(i, j)$ la **synergie** présumée fixe pour cet exemple (les valeurs issues du modèle sub-symbolique, symbolique ou mixte). Le système démarre avec $\omega_{i,j}(0) \approx 0$ (ou un bruit faible dans $[0, 0.05]$). En option, une **inhibition compétitive** peut s'ajouter, du type

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

afin de limiter la prolifération des liens, bien que l'on puisse initialement ne pas l'inclure (ou la fixer à un faible $\gamma \approx 0.01$).

B. Boucle de Simulation

La procédure la plus classique consiste à répéter l'**itération** suivante pour un nombre fixé T_{\max} de pas de temps (par exemple 500). À chaque itération t , on évalue pour toutes les paires (i, j) :

$$\begin{aligned} \omega_{i,j}(t+1) &= \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] \\ &- \gamma \sum_{k \neq j} \omega_{i,k}(t) \quad (\text{si l'inhibition latérale est activée}), \end{aligned}$$

suivi éventuellement d'un **clipping** $\max\{0, \cdot\}$ si la convention exige que $\omega_{i,j} \geq 0$. Une **mise à jour synchrone** (calcul des $\omega_{i,j}(t+1)$ à partir de toutes les $\omega_{i,j}(t)$) est préférée ici pour la simplicité. Entre deux itérations, on peut **enregistrer** la matrice $\{\omega_{i,j}(t)\}$ ou ses indicateurs sommaires (moyennes, variances).

C. Courbes de Convergence et Émergence de Blocs

En début de simulation, la plupart des pondérations restent très faibles. Puis, à mesure que l'on répète la mise à jour, les **paires** (i,j) pour lesquelles la synergie $S(i,j)$ est élevée s'accroissent nettement, tandis que d'autres stagnent ou régressent. Après plusieurs dizaines ou centaines d'itérations, deux ou trois ensembles **cohérents** ont tendance à se distinguer. Si l'on a prévu :

- Deux **sous-groupes** sub-symboliques (A et B) ;
- Un lot de **règles** $\{\mathcal{L}_1, \dots, \mathcal{L}_5\}$, dont certaines s'alignent sur A et d'autres sur B ;

alors, on observe qu'un groupe (A + logiques $\mathcal{L}_1, \mathcal{L}_2$) voit ses liens internes monter au-dessus d'un certain plateau (par exemple 0.5–0.7), formant ainsi un **cluster** bien défini, tandis que l'autre groupe (B + logiques $\mathcal{L}_3, \mathcal{L}_4$) évolue vers un bloc distinct d'intensités similaires, et la cinquième entité logique \mathcal{L}_5 peut se retrouver en marge ou intégrée faiblement à l'un des deux.

On peut **documenter** la convergence via plusieurs moyens :

Courbes $\omega_{i,j}(t)$: on choisit quelques paires (i,j) caractéristiques (intra-groupe, inter-groupe) et on trace $\omega_{i,j}(t)$ vs. t . Les liaisons intra-groupe montrent souvent une augmentation plus ou moins rapide jusqu'à un palier stable, tandis que les liaisons inter-groupe demeurent faibles ou oscillent en deçà d'un seuil.

Heatmap : on extrait régulièrement la matrice $\{\omega_{i,j}(t)\}$ et on la présente sous forme de *carte colorée*. Les "zones" correspondantes aux clusters apparaissent clairement, occupant des blocs plus saturés.

Algorithmes de détection de communautés (4.6.2.2) : si l'on souhaite une lecture plus automatisée, on peut appliquer, à chaque instant notable (ou à la fin de la simulation), un critère de modularité ou un algorithme Louvain sur la matrice $\{\omega_{i,j}\}$. En pratique, cela confirmera l'émergence de deux ou trois groupes majeurs.

D. Deux ou Trois Clusters au Final

En théorie, l'expérience montre que l'on obtient souvent **deux** clusters correspondant aux regroupements A et B, chacun couplé aux entités logiques qui partagent leur synergie. Toutefois, des ajustements subtils de la synergie ou des paramètres (η, τ, γ) peuvent introduire :

Un troisième bloc, si une portion de A ou B se détache sous l'influence de certaines logiques, ou si \mathcal{L}_5 forme un embryon de communauté autonome.

Un regroupement en un seul super-bloc si la différence sub-sym est insuffisante pour maintenir une réelle scission (par exemple τ trop faible, η trop élevé).

Dans une configuration “parfaite” (sous-cluster A resserré, B bien séparé, et logiques polarisées), la dynamique converge la plupart du temps vers **deux** agglomérations centrales, et éventuellement un “satellite” composé d’entités moins typées.

E. Conclusion

L’exécution concrète de la **règle de mise à jour** $\omega_{i,j}(t+1)$ dans l’exemple décrivant dix entités sub-symboliques + cinq logiques met en relief la **croissance** des liens intra-cluster et la **décroissance** ou la stagnation des liens inter-cluster, aboutissant en pratique à **2 ou 3 clusters** stables. Ces résultats se traduisent mathématiquement par la montée en plateau de $\omega_{i,j}$ pour les paires à **forte** synergie, la fixation d’un noyau **sub-sym**, et la marginalisation de certains nœuds restés en faible affinité. Les méthodes de *visualisation* et de *community detection* (sections 4.6.1 et 4.6.2) confirment l’**émergence** de blocs cohérents. Cela illustre précisément la vocation d’un **SCN** : faire apparaître de manière auto-organisée (et localement régulée) des **groupements** reflétant tant la dimension sub-symbolique (similarité vectorielle) que l’éventuelle cohérence symbolique (règles).

4.7.1.3. Discussion des Paramètres : η , τ , γ (Inhibition), etc.

Le **mini-exemple** présenté aux sections 4.7.1.1 et 4.7.1.2 a montré comment un **SCN** (Synergistic Connection Network) relativement modeste – rassemblant dix entités sub-symboliques et cinq entités logiques – se développe au travers d’une **dynamique** de mise à jour $\omega_{i,j}(t+1)$. L’expérience a simultanément mis en relief l’importance des **paramètres** η , τ et γ . Ces coefficients déterminent la **vitesse** à laquelle les pondérations $\omega_{i,j}$ se réajustent, la **hauteur** qu’elles peuvent atteindre avant saturation et l’**interaction compétitive** qui peut encourager chaque nœud à privilégier certains liens au détriment d’autres. Cet ensemble de réglages exerce une influence directe sur la **structure** finale : nombre de **clusters**, clarté de la partition, potentiel de bifurcation ou d’oscillation.

A. Rôle de η (Taux d’Apprentissage)

La **formulation additive** classique suppose :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η contrôle l’ampleur de la correction appliquée à la pondération $\omega_{i,j}(t)$ à chaque itération. La manière dont η influe sur la **vitesse** et la **stabilité** de la convergence est cruciale :

Un η **trop faible** peut rallonger considérablement la **phase transitoire** : la croissance ou la décroissance de $\omega_{i,j}$ demeurant minime à chaque pas, plusieurs centaines ou milliers d’itérations peuvent être nécessaires pour voir émerger des blocs cohérents.

Un η trop élevé peut au contraire causer des **oscillations** ou une divergence si la boucle de rétroaction devient excessive. La matrice $\omega_{i,j}(t)$ se met alors à “sauter” autour de la valeur d'équilibre sans s'y stabiliser, ou elle s'aventure vers des amplitudes anormales.

Un compromis pratique consiste à essayer des valeurs de $\eta \in [0.01, 0.1]$. Dans le **scénario** illustré (4.7.1.2), un $\eta = 0.05$ s'avère un point de départ correct permettant d'atteindre un **plateau** en quelques centaines d'itérations sans oscillations majeures.

B. Rôle de τ (Décroissance)

La mise à jour comporte un terme $\tau \omega_{i,j}(t)$ que l'on **soustrait** partiellement à la pondération. Cette “force de **décroissance**” empêche $\omega_{i,j}$ de croître indéfiniment en cas de synergie $S(i,j)$ positive. Dans un cadre stationnaire simple, on retrouve un **point fixe** $\omega_{i,j}^* = S(i,j)/\tau$. Ainsi, plus τ est grand, plus ce point d'équilibre demeure **bas**, et inversement.

Lorsque $\tau \approx 0$, le mécanisme se rapproche d'une croissance potentiellement illimitée, risquant des saturations hors de portée ou des oscillations.

Lorsque τ est élevé, la possibilité de liens forts exige que $S(i,j)$ soit vraiment substantiel. Autrement, $\omega_{i,j}^* \approx 0$ pour la plupart des paires.

Dans l'exemple, fixer $\tau = 1$ et $\eta = 0.05$ donne un produit $\eta \tau = 0.05$. Dans un schéma additif linéaire, cette configuration garantit généralement une **convergence** stable pour un large éventail de synergies. Pour des schémas plus sophistiqués (inhibition, non-linéarités), la stabilité peut nécessiter des conditions plus élaborées.

C. Rôle de γ (Inhibition Compétitive)

L'**inhibition latérale** (section 4.2.2.2) introduit un terme supplémentaire $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ dans la mise à jour $\omega_{i,j}(t+1)$. L'effet de cette composante est de **forcer** chaque nœud \mathcal{E}_i à limiter le nombre ou la force totale de ses liaisons en compétition. Lorsque γ est nul, chaque nœud peut, en théorie, entretenir de multiples liens “moyennement élevés”. Une inhibition plus **forte** ($\gamma \approx 0.05$) incite chaque nœud à se focaliser sur moins de liens, ou à “opter” pour un cluster plus net. Cela réduit les configurations mixtes et favorise l'apparition de **communautés** mutuellement exclusives, au risque de scinder des regroupements potentiellement plus larges si γ atteint un niveau trop élevé.

D. Autres Paramètres Pratiques

Outre η , τ et γ , d'autres choix peuvent impacter la dynamique :

Clipping $\omega_{i,j} \geq 0$. Il est fréquent d'imposer $\max\{0, \cdot\}$ pour éviter que certaines pondérations deviennent négatives, ce qui peut heurter l'interprétation de “lien fort”.

Sparsification. On peut décider de conserver seulement les k liaisons les plus fortes par nœud, ou celles dépassant un certain seuil dynamique (voir 4.4.2.2), pour favoriser l'émergence de blocs distincts.

Bruitage ou “recuit” (4.4.3.2). Injecter un léger bruit aléatoire dans la mise à jour $\omega_{i,j}$ aide parfois à échapper à des minima locaux ou à surmonter des phénomènes de multi-stabilité non désirés.

E. Synthèse des Réglages dans l’Exemple

Dans la simulation illustrée en 4.7.1.2, des valeurs modérées $\eta = 0.05$, $\tau = 1.0$ et $\gamma \approx 0$ (ou $\gamma = 0.01$) produisent typiquement la formation de deux à trois **clusters** en moins de 300 itérations, s’accompagnant d’une stabilisation visible dans les courbes $\omega_{i,j}(t)$. En revanche, si η est trop fort (0.2 ou 0.3), le réseau peut osciller ou hésiter plus longtemps avant de trouver un arrangement stable. Si τ est trop petit (0.1 ou 0.2), beaucoup de liaisons gonflent rapidement et les clusters manquent de “relief” ; si τ est trop grand (> 2), la plupart des pondérations restent faibles et l’on ne parvient pas à dégager des blocs cohérents, sauf si $S(i,j)$ est très prononcé. L’ajout d’une inhibition γ plus substantielle (0.05 ou 0.1) va forcer le réseau à “choisir” de manière plus drastique, potentiellement conduisant à une partition plus scindée, avec des blocs plus purs.

Conclusion

Les **paramètres** η (taux d’apprentissage), τ (décroissance) et γ (inhibition) se révèlent fondamentaux pour **guider** l’auto-organisation dans un **DSL**. Leur dosage détermine la **vitesse** de convergence, la **précision** des clusters, l’apparition ou non d’oscillations, ou encore le degré d’exclusivité dans les liens (γ favorisant un “winner-takes-most”). Dans un exemple simple (dix entités sub-symboliques, cinq entités logiques), la bonne combinaison de η, τ, γ permet d’aboutir en quelques centaines d’itérations à une structure claire, typiquement deux (ou trois) blocs majeurs, alignés avec la répartition sous-jacente. L’analyse visuelle des courbes, de la matrice ω et des partitions obtenues offre un retour précieux pour affiner ces paramètres en conséquence.

4.7.2. Cas Robotique ou Multi-Agent

En plus des scénarios de **simulation numérique** purement abstraits (4.7.1), un **SCN** (Synergistic Connection Network) peut s’appliquer à des contextes plus “réels” ou plus “pragmatiques”, comme la coordination de **robots** ou d’**agents multiples**. Dans ce type de cadre, la **synergie** $S(i,j)$ n’est plus simplement une similarité vectorielle ou une compatibilité logique ; elle peut représenter la **complémentarité** de tâches ou la **cohérence** d’actions entre robots. On souhaite alors que le **DSL** (Deep Synergy Learning) favorise la formation de **sous-groupes** de robots qui collaborent efficacement, tout en laissant d’autres robots moins impliqués si leur contribution ne s’imbrique pas bien.

4.7.2.1. Rappel : Flottes de Robots, Ex. 5 Robots Simulés, Synergie = Réussite Conjointe de Tâches

Les principes du **Deep Synergy Learning (DSL)**, décrits dans les sections précédentes, peuvent s’appliquer de manière naturelle à des scénarios **multi-agents**, en particulier lorsqu’on considère plusieurs **robots** ou entités dotées de capacités complémentaires. Pour illustrer cette mise en pratique, on imagine une flotte réduite de cinq robots $\{r_1, \dots, r_5\}$, chacun disposant de capteurs,

effecteurs ou aptitudes particulières, et l'on définit la synergie $S(r_i, r_j)$ sur la base de leur **réussite conjointe** dans des tâches passées ou de leur **compatibilité** fonctionnelle. L'**auto-organisation** d'un **SCN** édifié autour de ces robots peut alors faire émerger un partitionnement clair : un ensemble manipulatoire, un ensemble d'exploration, ou toute autre combinaison reflétant la meilleure configuration de collaboration selon les missions à accomplir.

A. Motivation Générale

Dans une configuration à cinq robots, il existe souvent un **partage** des rôles : certains se spécialisent dans la **manipulation** (bras robotisés), d'autres dans l'**inspection** (caméras, scanners, capacité de déplacement rapide). Les missions collectives imposent parfois une coordination de robots complémentaires, par exemple un robot manipulateur ayant besoin de l'assistance d'un robot d'inspection ou d'un robot de transport. La **synergie** $S(r_i, r_j)$ quantifie alors la probabilité (ou la qualité mesurée) que les robots r_i et r_j réalisent efficacement une tâche ensemble. Pour alimenter cette synergie :

On peut s'appuyer sur la **complémentarité** des modules (capteurs et actionneurs) : la présence d'un bras manipulateur et d'un détecteur de défauts peut s'additionner pour certaines missions.

On peut se baser sur un **historique** de performances : si deux robots ont maintes fois coopéré avec succès, $\text{Hist}(r_i, r_j)$ s'en retrouve valorisée.

On peut inclure la **distance** entre robots si la proximité spatiale influe la communication ou la facilité de travail : $S(r_i, r_j)$ décroît alors quand $\text{dist}(r_i, r_j)$ augmente.

Dans tous les cas, l'idée est de traduire la complémentarité et la réussite en un **score** S au sens du DSL, qui alimentera la mise à jour de $\omega_{i,j}(t)$.

B. Exemple Concret de Configuration

Supposons cinq robots $\{r_1, \dots, r_5\}$ dont trois $\{r_1, r_2, r_3\}$ sont dotés de bras manipulateurs. Les deux autres, $\{r_4, r_5\}$, privilégient l'inspection et la reconnaissance mobile, éventuellement plus rapides mais dépourvus de bras de préhension. Une **matrice** de synergie peut être construite : $S(r_i, r_j) \approx 0.7$ pour deux robots manipulateurs collaborant à une tâche d'assemblage, $S(r_i, r_j) \approx 0.4$ pour un manipulateur joint à un explorateur (un niveau moyen de coopération), et $S(r_i, r_j) \approx 0.1$ ou 0.2 pour deux robots explorateurs qui n'ont pas grand-chose à s'apporter dans une mission de montage.

En pratique, on peut enrichir la formulation :

$$S(r_i, r_j) = \alpha \text{Comp}(r_i, r_j) + \beta \text{Hist}(r_i, r_j) + \gamma \text{Dist}(r_i, r_j),$$

avec Dist prenant la forme d'une fonction décroissante de la distance géographique (si la proximité influe). Par souci de simplicité, on peut rester dans un cas stationnaire où $\{S(r_i, r_j)\}$ ne varie pas au cours du temps, pour observer la formation d'un SCN.

C. Pourquoi un SCN ?

Dans une solution **centralisée**, il serait ais  de programmer explicitement : “Les robots r_1, r_2, r_3 travailleront ensemble pour les missions d’assemblage, r_4, r_5 couvriront l’inspection”. Cependant, l’approche DSL propose une **auto-organisation** : chaque liaison $\omega_{i,j}(t+1)$ est mise   jour localement suivant la synergie $S(r_i, r_j)$ et un terme de d croissance ou d’inhibition. Cette mise   jour r p t e conduira les robots partageant une forte synergie   renforcer leurs liens, alors que des paires moins avantageuses demeureront ou se rapprocheront de z ro. Au final, un **cluster** regroupe naturellement les trois manipulateurs, tandis que les deux explorateurs ou inspecteurs conservent des pond rations r duites avec ceux-ci, ou forment un autre sous-bloc.

D. Exemple Num rique

Si l’on initialise $\omega_{i,j}(0) \approx 0$, puis qu’on applique, par exemple, la **r gle additive** d j   voqu e  :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(r_i, r_j) - \tau \omega_{i,j}(t)],$$

avec $\eta = 0.05$ et $\tau = 1.0$, alors les paires $\{(r_1, r_2), (r_1, r_3), (r_2, r_3)\}$ b n fici ront d’un **score** $S \approx 0.7$, ce qui leur permet de cro tre rapidement. Les paires (r_1, r_4) , (r_2, r_5) , etc., avec $S \approx 0.2$, demeureront plus modestes ou retomberont pr s de z ro. Apr s 100   200 it rations, on constate l’apparition d’un bloc $\{r_1, r_2, r_3\}$ relativement isol  du bloc $\{r_4, r_5\}$. Sur le plan **graphique**, la heatmap $\{\omega_{i,j}\}$ exhibe deux carr s plus satur s (ou un seul pour $\{r_4, r_5\}$, si ces deux explorateurs se connectent l’un   l’autre avec un certain niveau de synergie).

E. Analyse et Extensions

Un tel dispositif peut naturellement  voluer. Si l’on introduit un **changement** de mission, rendant la collaboration entre un manipulateur et un explorateur tout   fait pertinente, la synergie $S(r_i, r_j)$ peut progresser pour ce couple, et $\omega_{i,j}$ s’ leve au point de faire fusionner des sous-blocs ant rieurement disjoints. C’est l  que r side la force d’un SCN : l’**adaptation** en continu, sans qu’un planificateur centralis  impose le regroupement. On peut aussi envisager d’int grer un **terme d’inhibition lat rale** γ pour forcer chaque robot   ne cultiver que peu de liens forts, ou un *clipping* pour  viter la saturation hors de propos.

Ce **exemple**   5 robots d montre la facilit  avec laquelle un **DSL** capture une **auto-organisation** multi-agent : la **synergie** d finit la probabilit  ou la qualit  de r ussite conjointe, et la mise   jour $\omega_{i,j}$ concr tise la consolidation ou l’abandon progressif d’une liaison.   l’ chelle plus large, on peut imaginer des dizaines ou centaines de robots, r partis entre diverses missions. Le mod le DSL s’ tend alors de la m me fa on, exploitant la dimension partiellement locale et r active de la mise   jour pour assigner de facto les robots   des **clusters** de t ches.

Conclusion

La notion de **flotte de robots** constitue un **sc nario** privil gi  pour l’application d’un **SCN** : la **synergie** entre deux robots (r_i, r_j) refl te leur capacit    coop rer avec succ s, soit par compatibilit  d’effecteurs/capteurs, soit par l’historique de missions communes. En d roulant la dynamique de $\omega_{i,j}(t+1)$, on obtient naturellement la constitution d’un **cluster** de robots manipulateurs et un **cluster** de robots explorateurs, ou toute autre configuration correspondant aux

spécificités de leur collaboration. Cette démarche illustre la **puissance** du DSL, qui, sans supervision directe, génère un réseau de pondérations traduisant la **cohérence** (ou l'absence de cohérence) entre agents.

4.7.2.2. Observations : Un Cluster se Forme entre 3 Robots Complémentaires, les 2 Autres se Marginalisent ou se Lient Plus Faiblement

Dans le cadre d'une flotte de cinq robots, comme illustré en section 4.7.2.1, la dynamique des pondérations $\omega_{i,j}(t)$ appliquée par le Synergistic Connection Network (SCN) permet de mettre en évidence comment, sous l'hypothèse d'une synergie inégale entre certains robots, un **cluster principal** se forme spontanément. Ce cluster se compose typiquement de trois robots qui, en raison de leurs capacités complémentaires, développent des liens forts et cohérents, tandis que les deux autres robots présentent des synergies faibles avec ce groupe et avec eux-mêmes, conduisant ainsi à leur marginalisation ou à des connexions moins robustes. Nous allons maintenant développer en détail les différents aspects de cette observation en plusieurs parties.

A. Mise en Place et Déroulé de la Simulation

Le déploiement de la dynamique du SCN s'effectue en initialisant les pondérations $\omega_{i,j}(0)$ à des valeurs faibles, souvent proches de zéro, ce qui permet d'introduire un bruit initial qui simule des conditions réelles. La mise à jour des poids se fait selon la formule suivante :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(r_i, r_j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

où :

- η représente le **taux d'apprentissage** qui détermine la vitesse d'évolution des pondérations,
- $S(r_i, r_j)$ est la **synergie** entre les robots r_i et r_j , fonction qui varie de 0 à 1 en fonction de la complémentarité de leurs capacités,
- τ est le **coefficent de décroissance** qui pénalise les liens trop forts pour éviter une croissance illimitée,
- γ est le paramètre d'**inhibition** qui sert à limiter la diffusion excessive des pondérations sur plusieurs liens, encourageant ainsi une répartition plus sélective des connexions.

Dans la simulation, les paramètres sont choisis de sorte que $\eta \approx 0.05$, $\tau \approx 1.0$ et $\gamma \approx 0.02$. L'algorithme s'exécute pour environ 200 à 300 itérations, ou jusqu'à ce que les modifications des pondérations deviennent négligeables, indiquant une stabilisation du réseau.

La synergie $S(r_i, r_j)$ est définie de manière à refléter la qualité de la collaboration entre les robots. Par exemple, pour le sous-groupe de trois robots complémentaires, on attribue des valeurs élevées, typiquement comprises entre 0.7 et 0.8, alors que pour les paires impliquant les deux robots restants, les scores peuvent être de l'ordre de 0.2 à 0.3, voire moins. Cette différence dans les

valeurs de synergie conditionne la dynamique : les liens entre les robots complémentaires se renforcent, tandis que les autres restent faibles ou nulle.

B. Le Cluster de Trois Robots Complémentaires

Au cours des itérations, les pondérations reliant les robots du sous-groupe complémentaire — disons r_1 , r_2 et r_3 — augmentent progressivement. Par exemple, les liens $\omega_{1,2}(t)$ et $\omega_{1,3}(t)$ vont converger vers des valeurs proches de 0.6 à 0.7, ce qui correspond, en équilibre stationnaire, à la relation

$$\omega_{i,j}^* = \frac{S(r_i, r_j)}{\tau}.$$

Avec $\tau = 1$, si $S(1,2) = 0.8$ et $S(1,3) = 0.7$, on obtient respectivement $\omega_{1,2}^* \approx 0.8$ et $\omega_{1,3}^* \approx 0.7$. Ce phénomène de renforcement des liens parmi ces trois robots traduit la formation d'un **cluster principal**. L'effet d'inhibition, via le terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$, contribue à concentrer l'énergie de chaque robot sur les connexions les plus profitables. En conséquence, ces robots renforcent préférentiellement leurs liens entre eux, se détachant ainsi du reste de la flotte.

C. Les Deux Autres Robots : Marginalisation ou Liens Faibles

Les robots r_4 et r_5 ne bénéficient pas des mêmes synergies. Deux scénarios principaux peuvent se présenter :

- Dans un premier cas, si la synergie entre r_4 et r_5 est très faible (par exemple, $S(r_4, r_5) \approx 0.1$ ou 0.2) et que leurs liens avec le groupe principal sont également faibles (autour de 0.3), alors les pondérations associées à ces robots restent faibles. Graphiquement, les valeurs de $\omega_{4,j}(t)$ et $\omega_{5,j}(t)$ stagnent près de zéro ou oscillent à de faibles niveaux, indiquant que ces entités ne participent pas activement à la formation du cluster principal.
- Dans un second cas, si r_4 et r_5 présentent une synergie intermédiaire entre eux (par exemple, $S(r_4, r_5) \approx 0.5$), ils peuvent former un petit sous-cluster distinct, avec des pondérations convergeant vers une valeur modérée (par exemple, 0.4 à 0.45). Toutefois, leurs liens avec le trio principal restent inférieurs, ce qui les marginalise vis-à-vis du cluster dominant.

Ces situations illustrent comment, selon la répartition des synergies, le SCN peut organiser hiérarchiquement les interactions entre robots, favorisant les collaborations les plus pertinentes et isolant ou réduisant l'influence des entités moins complémentaires.

D. Stabilisation de la Configuration et Visualisation

Après un nombre suffisant d'itérations (typiquement 200 à 300), la dynamique converge vers une configuration stable où la matrice de pondérations $\{\omega_{i,j}\}$ présente un bloc de fortes connexions entre r_1 , r_2 et r_3 , et des liens significativement plus faibles pour les interactions impliquant r_4 et r_5 . La visualisation sous forme de **heatmap** permet de constater cette structuration : le bloc 3×3 correspondant au trio manipulateur affiche des valeurs proches de 0.6 à 0.8, tandis que les liens inter-clusters ou les connexions entre r_4 et r_5 apparaissent en contraste, souvent en dessous de 0.3. Des méthodes d'extraction de communautés, telles que l'indice de Silhouette ou l'algorithme de

Louvain, confirment que le cluster principal se distingue nettement, tandis que les deux autres robots restent soit isolés, soit regroupés dans un sous-ensemble à faible densité.

E. Conclusion

Les observations issues de cette simulation illustrent clairement que, dans un SCN appliqu       une flotte de robots, les entit   ayant une forte **compl  mentarit  ** se regroupent naturellement pour former un **cluster principal**. Dans notre cas, les robots r_1 , r_2 et r_3 d閙onstreront des liens solides gr  ce     des synergies    lev  es (typiquement entre 0.7 et 0.8), ce qui se traduit par des pond  rations $\omega_{i,j}$    lev  es. En revanche, les robots r_4 et r_5 pr  sentent des synergies faibles avec le groupe principal et entre eux, et leurs pond  rations restent faibles ou convergent vers des valeurs mod  r  es. Le param  tre d'inhibition γ joue un r  le essentiel en encourageant chaque robot     concentrer ses liens sur les partenaires les plus profitables, contribuant ainsi     la formation d'un cluster distinct. Ce r  sultat d閙onstre la capacit   du SCN     r  aliser une **auto-organisation** efficace, r  v  lant des clusters qui refl  tent la logique de collaboration inh  rente aux tâches robotiques, tout en marginalisant les entit   dont la synergie ne justifie pas un renforcement important des liens. Cette observation, issue de simulations num  riques, constitue un exemple concret de l'efficacit   du DSL dans des environnements multi-robots, et offre une base pour ajuster les param  tres du syst  me afin d'optimiser la coh  rence et la robustesse des interactions collaboratives.

4.7.2.3. Visualisation Dynamique (Cycles, Stabilisation)

Les **exp  riences** men  es dans le cadre robotique (voir 4.7.2.1 et 4.7.2.2) mettent en    vidence la capacit   d'un **Synergistic Connection Network** (SCN)     faire    merger, au sein d'une flotte de cinq robots, un **cluster** principal regroupant trois manipulateurs dot  s d'une synergie    lev  e, tandis que les deux autres, moins pertinents     la m  me t  che, se marginalisent ou conservent des pond  rations interm  diaires. Afin de **comprendre** plus finement la dynamique menant     cet    tat, il est pr  cieux d'adopter des **techniques de visualisation** qui r  v  lent la progression pas     pas des pond  rations $\omega_{i,j}(t)$, la survenue    ventuelle de **cycles** ou oscillations, ainsi que le moment pr  cis o  u se consolide la structure en sous-groupes.

A. Repr  sentation sous forme de "Graph Dynamique"

Pour repr  senter l'   volution d'un **SCN**, on peut choisir un **graph** pond  r   dont les n  uds sont les robots $\{r_1, r_2, r_3, r_4, r_5\}$ et les ar  tes sont les pond  rations $\omega_{i,j}(t)$. Le principe est de produire une **animation** en temps discret :     chaque it  ration ou palier t , on met     jour :

L'   paisseur (ou la couleur) de chaque ar  te en fonction de $\omega_{i,j}(t)$. Les liaisons fortes apparaissent plus visibles (ar  tes plus larges, couleur satur  e), les liaisons faibles restant minces ou quasi invisibles.

Le positionnement des n  uds dans le plan (si l'on recourt     un layout type *force-directed*), ce qui permet de voir les n  uds fortement connect  s se rapprocher progressivement. Certains utilisateurs conservent toutefois un positionnement fixe pour faciliter la comparaison dans le temps.

Cette représentation dynamique clarifie la formation du **cluster** des trois robots manipulateurs : au départ, toutes les arêtes sont fines ($\omega_{i,j} \approx 0$), puis celles reliant $\{r_1, r_2, r_3\}$ s'épaissent nettement, tandis que les connexions $\omega_{4,k}$ et $\omega_{5,k}$ demeurent maigres. Une fois l'itération 100 ou 150 atteinte, on constate un **noyau** visuel constitué de $\{r_1, r_2, r_3\}$, et le graphe se stabilise autour de ce bloc principal.

Cycles ou pseudo-oscillations.

Dans le cas où la dynamique de mise à jour (paramètres η, τ, γ) permet des **oscillations**, on percevrait un “va-et-vient” dans l’épaisseur de certaines arêtes, l’union de robots se renforçant brièvement puis chutant au profit d’un autre regroupement. Cet état se traduirait visuellement par une “valse” des nœuds dans le layout *force-directed*, ou par une alternance de couleurs pour des liens rivaux. Dans la configuration simple où la synergie des trois manipulateurs dépasse nettement celle des deux explorateurs, un tel phénomène survient peu, à moins que η ou γ ne soient mal calibrés.

B. Heatmap et Courbes de Pondérations

En parallèle d’un *graph* dynamique, on peut stocker et afficher la matrice $\{\omega_{i,j}(t)\}$ sous forme de **heatmap** (section 4.6.1.2). Du fait que la flotte compte seulement cinq robots, la matrice 5×5 est aisément lisible. Dès que les liens $\{\omega_{i,j}\}$ intra-cluster ($r_1 \leftrightarrow r_2$, $r_2 \leftrightarrow r_3$, etc.) grimpent, les cellules correspondantes du carré deviennent plus “chaudes” (couleurs saturées), alors que les autres restent plus froides (teintes pâles). On suit ainsi la montée de la diagonale 3×3 correspondant au noyau manipulateur, et la stagnation de $\omega_{4,j}$ et $\omega_{5,j}$.

Un autre niveau de détail s’obtient en **traçant** les courbes $\omega_{i,j}(t)$ pour quelques paires repères. Pour la paire (r_1, r_2) ou (r_2, r_3) , une courbe monotone ou faiblement oscillatoire croît de 0 vers un plateau proche de 0.6–0.7, tandis que pour la paire (r_4, r_1) , elle plafonne à 0.2 ou 0.3. Le moment où l’on atteint 90 % du plateau renseigne précisément sur l’itération où la **structure** se scelle. Si aucune oscillation n’apparaît, on parle de stabilisation ; si, au contraire, on voit un comportement fluctuant, c’est signe d’un **régime cyclique** ou multi-stable.

C. Intérêt de la Visualisation pour le Diagnostic

Cette **visualisation** dynamique permet de déterminer la **temporalité** de la formation du cluster manipulateur : on aperçoit à quelle itération les liaisons $\omega_{1,2}$ et $\omega_{2,3}$ commencent à surpasser un seuil implicite (ex. 0.5), scellant la communauté $\{r_1, r_2, r_3\}$. Les deux robots restants $\{r_4, r_5\}$ conservent des pondérations plus modestes, se trouvant donc marginalisés en fin de simulation.

En complément, la visualisation dévoile si la **dynamique** traverse une **phase transitoire** agitée (fortes oscillations puis stabilisation tardive) ou se fixe rapidement en un **cluster** ferme. À partir de là, il devient possible d’ajuster η , τ , γ afin d’obtenir un compromis satisfaisant entre **rapidité** de convergence, **contrôle** des oscillations et **clarté** de la séparation des robots.

D. Conclusion

Le recours à un **graph dynamique** (avec épaisseurs d’arêtes) ou à une **heatmap** itérative rend tangibles les phénomènes d’**auto-organisation** dans un **SCN** appliqué à une flotte robotique. Dans l’exemple, on suit l’évolution des pondérations $\{\omega_{i,j}(t)\}$ et on voit émerger un **cluster** de trois

robots manipulateurs solidement reliés, tandis que les deux autres s'écartent par manque de synergie. Selon la distribution initiale de $\omega_{i,j}$ et la différence entre les synergies, on constate un chemin plus ou moins direct vers la stabilisation. Cette approche visuelle est un **complément** essentiel aux seules mesures numériques : elle révèle la nature temporelle des liens, **comment** et **quand** s'établit la structure, et **déetecte** aisément la présence éventuelle d'oscillations ou de compétitions prolongées.

4.7.3. Liens Pratiques

Après avoir illustré deux types de cas (4.7.1 : simulation semi-artificielle ; 4.7.2 : application robotique), il est naturel de se demander **comment** mettre en œuvre concrètement un **SCN** (Synergistic Connection Network) pour **reproduire** ces expériences ou en concevoir de nouvelles. La présente section (4.7.3) aborde quelques **pistes pratiques**, notamment :

Le code Python/C++ minimal pour gérer la dynamique $\omega_{i,j}(t + 1)$ (4.7.3.1),

Un outil de monitoring (“SCN Dashboard”) qui offre un **suivi** en temps réel de la formation des clusters (4.7.3.2).

4.7.3.1. Ce qu'on Pourrait Coder en Python/C++ pour Reproduire cette Dynamique en Quelques Centaines de Lignes

Il est souvent instructif de passer d'un **exemple théorique** à une **implémentation pratique**, même minimale, afin de rendre tangibles les principes de mise à jour $\omega_{i,j}(t + 1)$ d'un **Synergistic Connection Network (SCN)**. L'objectif** d'un tel prototype est de **montrer** comment, dans un code concis (Python ou C++), on peut :

224. Définir l'ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$.
225. Construire la **matrice** de synergie $\mathbf{S} = (S(i,j))$.
226. Gérer la **matrice** $\boldsymbol{\omega}(t)$ représentant les pondérations.
227. Boucler sur un nombre d'itérations T_{\max} afin d'**exécuter** la dynamique de mise à jour en s'appuyant sur les formules du **DSL**.

De cette façon, il devient possible, en quelques centaines de lignes, de **reproduire** les scénarios décrits précédemment : que ce soit le cas **mixte** (entités sub-symboliques, entités logiques) de la section 4.7.1, ou l'exemple **robotique** de la section 4.7.2, ou tout autre schéma dans lequel on veut explorer la formation de clusters par **auto-organisation**.

A. Structure Globale du Code

Le **cœur** de l'implémentation consiste à :

228. **Définir le nombre** d'entités, noté n .
229. **Allouer et remplir** la matrice $\mathbf{S} \in \mathbb{R}^{n \times n}$, qui fixe la **synergie** $S(i, j)$.
230. **Initialiser** la matrice $\boldsymbol{\omega}(0) \approx 0$ (ou un léger bruit), éventuellement en imposant $\omega_{i,i} = 0$ s'il n'existe pas de liaison réflexive.
231. Choisir les **paramètres** : η (taux d'apprentissage), τ (décroissance), γ (inhibition compétitive éventuelle) et le **nombre** d'itérations T_{\max} .
232. Entrer dans la boucle principale sur $t \in \{0, \dots, T_{\max} - 1\}$, où l'on met à jour $\omega_{i,j}(t + 1)$ à partir de $\omega_{i,j}(t)$ et de $S(i, j)$.
233. (Optionnel) Stocker régulièrement des instantanés de la matrice $\boldsymbol{\omega}(t)$ pour les besoins de visualisation ou d'analyse.

Cette démarche se transpose aussi bien en **Python** qu'en **C++**. Dans Python, on utilise souvent **numpy** pour manipuler les matrices ; en C++, on recourra à des tableaux 2D ou à des conteneurs appropriés.

B. Exemple de Pseudo-Code (Version Additive)

Le pseudo-code ci-dessous, essentiellement en Python, illustre la **formulation additive** vue au chapitre 4.2.1 :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t)$$

(avec un éventuel γ pour l'inhibition compétitive). Le code est exprimé de manière lisible, sans prétendre à une optimisation poussée :

```
import numpy as np

# 1) Définition des paramètres et données
n = 15          # nombre d'entités
eta = 0.05      # taux d'apprentissage
tau = 1.0       # décroissance
gamma = 0.0      # si on veut ajouter l'inhibition compétitive
T_max = 300     # nombre d'itérations
```

```
# Matrice de synergie S (n x n), initialisée à zéro
S = np.zeros((n, n))
# => Ici, on doit remplir S[i,j] selon le scénario :
#   - sub-symbolique / logique mixte,
#   - ou robots, etc.
```

```
# Matrice des pondérations w (n x n)
w = np.zeros((n, n))
```

```

# => on peut ajouter un petit bruit :
# w = np.random.uniform(0.0, 0.05, (n, n))
# for i in range(n):
#   w[i,i] = 0.0

# 2) Boucle de mise à jour
for t in range(T_max):
    w_next = np.copy(w)
    for i in range(n):
        for j in range(n):
            if i != j:
                # Terme principal
                delta = eta * (S[i,j] - tau*w[i,j])
                # Inhibition compétitive
                if gamma > 1e-12:
                    # somme de w[i,k] pour k != j
                    inhibition_term = gamma * (np.sum(w[i,:]) - w[i,j])
                    delta -= inhibition_term
                w_next[i,j] = w[i,j] + delta

                # Option : clipping pour éviter w < 0
                if w_next[i,j] < 0.0:
                    w_next[i,j] = 0.0

    w = w_next

# (Option) Stocker ou imprimer un résumé de la matrice w(t)
# if t % 10 == 0:
#   print(f"Iteration {t}: some stats, e.g. mean={w.mean():.3f}, max={w.max():.3f}")

```

3) A la fin, w contient la matrice des pondérations finales

Ce **squelette** en Python, d'une centaine de lignes ou moins, fournit déjà le **cœur** d'un SCN dynamique. Pour un **scénario** plus complexe, on étoffe la partie **construction** de la matrice **S** (par exemple, pour des robots ou des entités logiques, on calcule les scores sub–sub, sym–sym, sub–sym). De même, on peut rendre la décroissance ou l'inhibition plus sophistiquées.

C. Extensions et Version C++

Pour reproduire cette simulation en C++, on définit des **tableaux** ou **vecteurs** 2D (par exemple `std::vector<std::vector<double>>`) pour **S** et **w(t)**. Le schéma principal reste identique : une double boucle `for i in range(n): for j in range(n): ...` actualise $\omega_{i,j}$. On peut stocker la matrice à la fin de la simulation ou à intervalles réguliers.

Dans les deux langages, un **point crucial** est la manière dont la synergie **S** se construit, selon l'hétérogénéité des entités (partie sub-symbolique, symbolique, etc.). Le code de mise à jour reste

lui-même identique, témoignant de la simplicité structurelle d'un SCN une fois que la **fonction** $S(i,j)$ est donnée.

D. Paramétrisation et Scénarios Multiples

Dans un **mini-programme** de ce type, on peut aisément :

Laisser η , τ , γ être passés en **argument** (ou chargés depuis un fichier de config) afin d'expérimenter la sensibilité de la convergence.

Générer **S** selon des distributions semi-aléatoires : par exemple, disjoindre des **sous-groupes** d'entités sub-symboliques avec un niveau de similarité 0.7–0.8 en interne et 0.2–0.3 à l'externe, rattacher à cela quelques entités logiques 0–1, etc.

Introduire un **mode évolutif** où **S** se modifie au fil du temps, ce qui exigerait un recalculation de $S(i,j)$ avant chaque itération.

Ainsi, quelques douzaines ou centaines de lignes de code suffisent à couvrir tous les **scénarios** des exemples (4.7.1) et (4.7.2), qu'il s'agisse de **robots** ou d'entités logiques/sub-symboliques mixtes.

E. Observation et Post-Traitements

Un code minimal comme ci-dessus se borne à mettre à jour ω . Dans la pratique, on procède à des **observations** pour :

Sauvegarder la matrice $\omega(t)$ toutes les 10 ou 20 itérations.

Produire une **heatmap** ou un **graph** (utilisant, par exemple, NetworkX en Python) pour illustrer l'évolution visuelle (4.6.1).

Mesurer la **modularité** ou d'autres indices de partition (4.6.2) afin de vérifier la formation de clusters.

Ces étapes ne dépassent pas quelques lignes supplémentaires, tout en révélant la **dynamique** interne et la structure émergente.

Conclusion

En quelques centaines de lignes Python (ou C++), il est possible de **coder** la mise en œuvre fondamentale d'un **SCN** : on définit **S** (synergie), on initialise $\omega \approx 0$, puis on exécute la boucle de mise à jour $\omega_{i,j}(t+1)$. Malgré la brièveté du code, on parvient à reproduire tous les mécanismes décrits au chapitre 4.2 et dans les exemples (4.7.1, 4.7.2). Il suffit de paramétrier η, τ, γ et de spécifier **S** en fonction du contexte (sub-symbolique, logique, robotique, etc.). En sortie, la matrice ω obtenue rend compte de l'**auto-organisation** en clusters, démontrant la force et la **simplicité** conceptuelle d'un DSL une fois traduite en code.

B. Conclusion (4.7.3.1)

Le développement d'un **mini-code** (en Python ou en C++) pour mettre en place la **dynamique** d'un **Synergistic Connection Network** (SCN) ne requiert qu'un nombre restreint d'instructions, généralement de l'ordre de quelques dizaines ou centaines de lignes. La logique est en effet assez directe : on se dote d'une **matrice S**, qui contient les synergies $S(i, j)$ entre toutes les entités, puis on déclare une **matrice $\omega(0)$** (initialisée à zéro ou à un faible bruit) et une **boucle** d'itérations qui applique la règle de mise à jour décrite au chapitre 4.2 (additive, multiplicative, ou une variante avec inhibition). Il suffit enfin d'observer (ou d'enregistrer) l'évolution de $\omega_{i,j}(t)$ à chaque pas, afin de visualiser la montée ou la stagnation des liens et, in fine, la formation de **clusters**.

L'écriture du code en **Python** permet un prototypage rapide et une intégration aisée avec des librairies de visualisation (matplotlib, seaborn) ou de community detection (networkx). En **C++**, on gagne en performance si le nombre d'entités n devient important, par exemple au-delà de quelques milliers, tout en conservant le même principe de mise à jour. Dans les deux cas, on peut établir divers **scénarios** en variant la façon dont la synergie **S** est calculée : un mélange sub-symbolique/symbolique (tel que présenté en 4.7.1), une configuration multi-robots (4.7.2), ou toute autre instance s'appuyant sur des scores de complémentarité et d'historique.

Pourvu que le code structure correctement l'initialisation des $\omega_{i,j}$ et la boucle itérative, on peut alors, sans effort notable, y ajouter des **outils** pour la **collecte** ou l'**analyse** des résultats : export des matrices $\omega(t)$ à intervalles réguliers, tracés de courbes, heatmaps itératives, algorithmes de détection de communautés pour identifier les blocs émergents. Cette démarche rend possibles des **expérimentations** plus vastes ou plus complexes, tout en conservant la simplicité fondamentale du SCN. Ainsi, le cœur même de la **dynamique** ne nécessite qu'un petit nombre d'instructions, démontrant la **puissance** et la **généralité** d'un DSL, et la facilité avec laquelle on peut reproduire les scénarios présentés dans ce chapitre 4.7 en quelques centaines de lignes de code.

4.7.3.2. Aperçu du “SCN Dashboard” : Un Outil de Monitoring des $\omega_{i,j}$ pour Déetecter la Formation des Clusters en Temps Réel

Dans un cadre de **Deep Synergy Learning (DSL)**, l'implémentation d'un **Synergistic Connection Network (SCN)** nécessite non seulement l'élaboration d'une dynamique de mise à jour des pondérations $\omega_{i,j}$, mais également un outil permettant de **suivre** en temps réel l'évolution de ces poids afin de détecter la formation des clusters et d'identifier les points critiques de réorganisation du réseau. Cet outil, que l'on désigne par le terme “**SCN Dashboard**”, offre une interface de monitoring et de visualisation qui traduit en temps réel la dynamique d'auto-organisation du système. Dans cette section, nous décrivons en détail les principes, l'architecture et les modalités opérationnelles de ce dashboard, en nous appuyant sur des formules mathématiques et des mécanismes algorithmiques concrets.

A. Principes Généraux du SCN Dashboard

Le **SCN Dashboard** a pour objectif de fournir une **vue synthétique** de l'évolution de la matrice de pondérations $\omega(t) = \{\omega_{i,j}(t)\}$ au cours des itérations. Pour ce faire, le dashboard récupère régulièrement des instantanés de la matrice et les présente sous forme de visualisations interactives. Parmi ces visualisations, on retrouve notamment une **heatmap** de la matrice **ω** , des **courbes**

d'**évolution** pour certaines liaisons représentatives, ainsi que des indicateurs quantitatifs tels que la **moyenne** et la **variance** des poids, ou encore des indices de **modularité** issus d'algorithmes de détection de communautés.

Matériellement, pour chaque itération t de la dynamique définie par

$$\begin{aligned} & \omega_{i,j}(t+1) \\ &= \omega_{i,j}(t) \\ &+ \eta [S(i,j) - \tau \omega_{i,j}(t)] \quad (\text{éventuellement complétée par des termes d'inhibition ou de clipping}), \end{aligned}$$

le dashboard capte les valeurs de $\omega_{i,j}(t)$ et calcule des métriques résumées, telles que

$$\begin{aligned} \bar{\omega}(t) &= \frac{1}{n(n-1)} \sum_{i \neq j} \omega_{i,j}(t) \\ \sigma_{\omega}(t) &= \sqrt{\frac{1}{n(n-1)} \sum_{i \neq j} (\omega_{i,j}(t) - \bar{\omega}(t))^2}. \end{aligned}$$

Ces scalaires, ainsi que d'autres indicateurs (par exemple, le nombre de liens dépassant un seuil ω_{\min}), sont affichés en temps réel pour permettre à l'utilisateur d'évaluer la **stabilité** du système.

B. Architecture Technique du Dashboard

Pour construire un **SCN Dashboard** opérationnel, plusieurs architectures techniques sont envisageables. La mise en œuvre peut se faire dans un environnement distribué ou en local, selon le volume des données et la nécessité d'une actualisation en temps réel. Deux approches principales sont :

Serveur intégré avec interface web :

Le système de simulation du SCN s'exécute sur un serveur (par exemple, en Python ou Node.js) et stocke périodiquement les instantanés de la matrice $\omega(t)$ dans une base de données ou un fichier de log. Une interface web, développée en JavaScript (avec des bibliothèques telles que D3.js ou Plotly), interroge régulièrement ce serveur pour mettre à jour les visualisations. Cette approche permet une **interaction** en temps réel, où l'utilisateur peut ajuster dynamiquement les paramètres (par exemple, les valeurs de η , τ ou γ) via des **contrôles** graphiques (sliders, menus déroulants).

Processus séparé et visualisation post-hoc :

Dans un second schéma, la simulation du SCN est exécutée en tant que processus distinct, qui enregistre périodiquement les matrices $\omega(t)$ dans un fichier. Un outil de visualisation, également développé en Python (par exemple, en utilisant Matplotlib ou Seaborn), lit ces fichiers à intervalles réguliers et met à jour les graphiques. Cette solution est plus adaptée à des analyses postérieures, mais peut être complétée par des mécanismes d'actualisation en temps réel pour des applications nécessitant une surveillance continue.

Dans les deux cas, le dashboard intègre des **mécanismes de filtrage** pour réduire la charge visuelle. Par exemple, lorsque le nombre d'entités n est très élevé, le dashboard peut n'afficher que les liens dont les pondérations dépassent un certain seuil ou utiliser des techniques de **clustering** pour regrouper visuellement les entités.

C. Visualisations et Indicateurs

Le dashboard se doit de présenter plusieurs types de visualisations pour rendre la dynamique du SCN intelligible :

- **Heatmaps** : La matrice $\omega(t)$ est représentée sous forme d'une heatmap, où chaque cellule (i,j) est colorée en fonction de la valeur de $\omega_{i,j}(t)$. Cette représentation permet de visualiser instantanément la formation de clusters (zones de couleur intense) et de détecter des zones de faibles connexions.
- **Graphes dynamiques** : Une représentation sous forme de graphe, avec des nœuds représentant les entités et des arêtes pondérées par $\omega_{i,j}(t)$, offre une perspective plus intuitive de la topologie émergente. Les algorithmes de layout de type « force-directed » repositionnent les nœuds en fonction de la force des liens, de sorte que les clusters apparaissent comme des groupes de nœuds rapprochés.
- **Courbes temporelles** : Pour certaines paires d'entités sélectionnées, le dashboard affiche l'évolution de $\omega_{i,j}(t)$ en fonction du temps, permettant d'identifier des régimes de convergence, d'oscillations ou d'instabilité.
- **Indicateurs scalaires** : Des statistiques globales telles que la moyenne, la variance et des indices de modularité (issus d'algorithmes de community detection) sont calculées et affichées. Ces indicateurs permettent de quantifier la qualité de la partition en clusters et de surveiller la stabilité du système.

L'utilisateur peut également disposer d'**outils interactifs** pour modifier les paramètres du système en direct. Par exemple, des sliders peuvent permettre de varier η , τ ou γ et d'observer immédiatement l'impact de ces modifications sur la convergence ou la formation de clusters.

D. Mise en Pratique et Exemples d'Utilisation

Pour illustrer le fonctionnement du dashboard, considérons l'exemple d'une flotte de robots dont les pondérations évoluent selon la règle du SCN. Le dashboard permet alors de suivre en temps réel la matrice $\omega(t)$ et de détecter, par exemple, la formation d'un cluster principal de robots complémentaires. Au fur et à mesure que la simulation progresse, la heatmap révèle l'émergence d'un bloc de valeurs élevées correspondant aux liens internes du cluster. Parallèlement, un graphe dynamique montre que les nœuds représentant ces robots se rapprochent physiquement, tandis que les robots moins complémentaires se déplacent en périphérie. De plus, des courbes temporelles affichées pour certains liens illustrent la convergence exponentielle vers des points fixes, tandis que d'autres liens montrent des oscillations indiquant des transitions entre différents régimes d'interaction.

Un tel dashboard ne se contente pas de visualiser la dynamique, il sert également de **outil de diagnostic** et de **pilotage**. Par exemple, si les indices de modularité diminuent brusquement ou si

la variance des pondérations augmente, cela peut alerter l'opérateur sur une possible instabilité ou sur la nécessité de réajuster les paramètres du SCN (par le biais d'une inhibition plus forte ou d'un ajustement du taux d'apprentissage). Ce retour d'information en temps réel est particulièrement précieux dans des applications critiques, telles que la coordination multi-robots en environnement dynamique, où l'adhérence à une configuration stable et cohérente est essentielle pour la réussite de la mission.

E. Conclusion

Le **SCN Dashboard** se présente comme un outil de monitoring interactif permettant de visualiser et de contrôler la dynamique des pondérations $\omega_{i,j}$ dans un Deep Synergy Learning. En exploitant des visualisations telles que des heatmaps, des graphes dynamiques et des courbes temporelles, le dashboard offre un aperçu détaillé de la formation des clusters en temps réel. Il permet non seulement de détecter rapidement la consolidation des liens entre entités fortement synergiques, mais aussi d'identifier des signaux précurseurs d'instabilités ou d'oscillations, facilitant ainsi l'ajustement des paramètres du système. En résumé, ce tableau de bord constitue un pont essentiel entre la théorie du DSL et son application pratique, offrant à la fois un outil de recherche pédagogique et une interface de supervision opérationnelle dans des environnements multi-agents ou robotiques.

4.8. Stabilisation Avancée : Couplages avec Chapitre 5 (Architecture SCN)

Les sections antérieures du chapitre 4 ont mis en lumière la **dynamique** d'un SCN (Synergistic Connection Network) : les mises à jour $\omega_{i,j}$, la formation de clusters, la gestion de paramètres comme η , τ , γ . Toutefois, pour qu'un **DSL** (Deep Synergy Learning) fonctionne de manière **pérenne et scalable**, il est aussi crucial de **concevoir l'architecture logicielle** appropriée, détaillée au **Chapitre 5**. Cette architecture doit non seulement soutenir la **dynamique auto-organisée** mais également permettre :

- Un **déploiement** clair des différents modules (calcul de $S(i,j)$, mise à jour de $\omega_{i,j}$, inhibition globale, etc.),
- Une **gestion** ordonnée des ressources (threads, verrous, données),
- Une **intégration** flexible (possibilité de changer de schéma de mise à jour ou d'inhibition).

4.8.1. Rôle de l'Architecture

4.8.1.1. Comment l'Organisation Logicielle (Chap. 5) Facilite le Déploiement de la Dynamique Auto-Organisée

La mise en œuvre efficace d'un **Synergistic Connection Network (SCN)** repose sur une **infrastructure logicielle** soigneusement conçue qui permet de séparer les responsabilités au sein du système et de gérer la dynamique d'auto-organisation de manière modulable et évolutive. L'organisation logicielle, telle que décrite au Chapitre 5, joue un rôle déterminant dans le déploiement de la dynamique de mise à jour des pondérations $\omega_{i,j}(t)$ et dans l'intégration des divers modules de calcul de la **synergie** $S(i,j)$, de l'**inhibition** et du **contrôle des paramètres**.

A. Modularisation et Séparation des Rôles

L'un des principes fondamentaux de l'**organisation logicielle** appliquée à un SCN est la **modularisation**. Plutôt que d'imbriquer l'ensemble des calculs dans un unique bloc de code, on divise le système en modules indépendants, chacun ayant un rôle spécifique. Par exemple, le **Module de Calcul de Synergie** est chargé d'extraire et de combiner les informations issues de diverses sources (embeddings sub-symboliques, règles symboliques, etc.) afin de produire un score $S(i,j)$. Ce module peut utiliser diverses méthodes, telles que la **similarité cosinus** ou des noyaux RBF, pour obtenir une mesure quantitative de la proximité entre deux entités. De même, le **Module de Mise à Jour des Pondérations** applique la règle d'auto-organisation, généralement de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage** et τ le **coefficent de décroissance**. Enfin, un **Module d'Inhibition/Régulation** peut être ajouté pour introduire des mécanismes de sparsification ou de limitation (par exemple, via un terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$) qui garantissent que la structure du réseau reste compacte et que les liens moins pertinents ne perturbent pas la dynamique globale.

Cette séparation permet d'isoler les modifications : un changement dans la méthode de calcul de $S(i,j)$ (par exemple, passer d'un modèle sub-symbolique à une approche hybride) n'impacte pas directement la logique de mise à jour des pondérations, facilitant ainsi la maintenance et l'évolution du code.

B. Flexibilité dans le Choix des Schémas

L'**organisation logicielle** favorise également la flexibilité dans le choix des schémas de mise à jour. Par exemple, la règle de mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

peut être aisément remplacée par une variante multiplicative si le contexte l'exige, comme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \exp(\eta[S(i,j) - \tau \omega_{i,j}(t)]).$$

L'architecture logicielle modulaire permet de modifier uniquement le **Module de Mise à Jour des Pondérations** sans avoir à réécrire la totalité du système. Par ailleurs, les paramètres globaux η , τ et γ peuvent être centralisés dans un **Module de Gestion des Paramètres**, qui fournit des interfaces permettant d'ajuster ces valeurs en fonction des besoins de l'application ou des observations en temps réel.

C. Gestion Cohérente des Itérations et du Flux de Données

Un autre aspect clé de l'organisation logicielle est la gestion structurée de la boucle itérative qui orchestre le SCN. Une **boucle centrale** ou **engine** est mise en place pour :

- Appeler le **Module de Calcul de Synergie** pour recalculer $S(i,j)$ (si nécessaire, notamment dans des systèmes non stationnaires),
- Appliquer la règle de mise à jour aux pondérations $\omega_{i,j}$ à chaque itération,
- Activer des mécanismes de **monitoring** et de **visualisation** (par exemple, en enregistrant périodiquement la matrice $\omega(t)$ pour en générer des heatmaps ou des graphes dynamiques).

Ce processus est rendu encore plus robuste par l'introduction d'**observateurs** (observers) qui analysent la convergence, par exemple en mesurant la norme $\|\omega(t+1) - \omega(t)\|$ et en déclenchant des ajustements si la convergence se révèle trop lente ou si des oscillations apparaissent. De plus, la modularité de la boucle permet d'intégrer des mécanismes de contrôle comme le recuit stochastique pour échapper aux minima locaux peu satisfaisants.

D. Stabilisation et Contrôle de la Dynamique

L'**organisation logicielle** favorise une gestion fine de la dynamique d'auto-organisation grâce à des modules spécifiques qui surveillent et ajustent le comportement du SCN. Par exemple, un **Module de Stabilisation** peut intervenir en temps réel pour ajuster les paramètres η et τ en fonction des indicateurs de convergence et des oscillations détectées. Ce module peut également appliquer des techniques de **clipping** et de **sparsification** pour empêcher une prolifération excessive de liens, garantissant ainsi la robustesse et la lisibilité de la matrice ω .

L'approche permet de tester différents scénarios : dans un système multi-entités, la même infrastructure logicielle peut être utilisée pour différents types d'applications (robotique collaborative, agents conversationnels, systèmes de recommandation, etc.), en modifiant uniquement le module de calcul de synergie $S(i,j)$ et en adaptant les paramètres de la boucle d'itération. Cette capacité à réutiliser et à adapter le **moteur d'auto-organisation** est l'un des avantages majeurs de l'organisation logicielle présentée au Chapitre 5.

E. Intégration Multi-Scénario et Maintenance

Enfin, l'architecture logicielle permet une **intégration multi-scénario** efficace. Qu'il s'agisse de systèmes sub-symboliques, symboliques ou hybrides, le même cadre d'auto-organisation peut être déployé avec des modules spécifiques pour le calcul de la synergie et pour la régulation des pondérations. Cette uniformisation facilite non seulement le développement initial mais aussi la maintenance à long terme, puisque chaque composant évolue de manière indépendante. Par ailleurs, les interfaces de monitoring et de contrôle offertes par le SCN Dashboard (voir section 4.7.3.2) permettent aux opérateurs de visualiser en temps réel l'évolution du réseau et d'ajuster les paramètres via des interfaces ergonomiques, rendant ainsi l'ensemble du système adaptable à des environnements en constante évolution.

Conclusion

En résumé, l'**organisation logicielle** présentée au Chapitre 5 facilite le déploiement de la **dynamique auto-organisée** dans un SCN en offrant une **modularisation** claire qui sépare le calcul de la synergie, la mise à jour des pondérations et les mécanismes de régulation. Cette approche permet non seulement d'optimiser la maintenance et l'évolution du code, mais aussi d'adapter le système à divers scénarios applicatifs grâce à des modules interchangeables et à des interfaces de monitoring en temps réel. Les paramètres globaux tels que η , τ et γ sont gérés de manière centralisée, facilitant ainsi leur ajustement en fonction des besoins du système et garantissant une **stabilisation dynamique** efficace. Cette infrastructure logicielle constitue le socle sur lequel repose l'implémentation robuste, évolutive et maintenable d'un DSL, capable de s'adapter aux exigences complexes des environnements multi-agents et robotiques modernes.

4.8.1.2. Notion de Modules Distincts : “Calcul de Synergie”, “Update ω ”, “Inhibition Global ou Local”, etc.

Dans la mise en œuvre d'un **Synergistic Connection Network (SCN)**, il apparaît primordial de structurer le code en **modules distincts** afin de garantir la **clarté**, la **flexibilité** et la **maintenabilité** de l'architecture. Cette approche modulaire, qui sera développée plus en détail au Chapitre 5, permet de séparer les responsabilités liées au **calcul de la synergie**, à la **mise à jour des pondérations** $\omega_{i,j}$, et aux mécanismes de **régulation** tels que l'inhibition (globale ou locale). En procédant ainsi, le système se construit comme un ensemble de briques fonctionnelles interopérables, chacune étant aisément remplaçable ou ajustable selon le contexte d'application. Nous détaillons ci-après les principaux modules qui constituent ce cadre opérationnel.

A. Module “Calcul de Synergie”

Le **module de Calcul de Synergie** est chargé de déterminer le score $S(i, j)$ qui mesure la qualité de la relation entre deux entités \mathcal{E}_i et \mathcal{E}_j . Ce score peut être obtenu à partir de représentations sub-symboliques (telles que des **embeddings** issus de CNN, Transformers, etc.), de critères symboliques (basés sur des règles logiques, axiomes ou ontologies), ou d'une combinaison hybride de ces deux approches. Mathématiquement, on peut écrire la fonction de synergie de manière générique comme

$$S(i, j) = f(\mathbf{r}(i), \mathbf{r}(j)),$$

où $\mathbf{r}(i)$ représente la **représentation** de l'entité i (qui peut être un vecteur, un ensemble de règles, ou une concaténation de ces deux types d'informations) et f est une fonction mesurant la similarité ou la compatibilité. Par exemple, dans un contexte purement sub-symbolique, on utilisera typiquement la **similarité cosinus** :

$$S_{\text{sub}}(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

tandis que dans un contexte symbolique, $S_{\text{sym}}(i, j)$ pourra être définie en fonction de la concordance entre deux ensembles de règles R_i et R_j . Dans un système hybride, on opère une fusion pondérée telle que

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(i, j) + (1 - \alpha) S_{\text{sym}}(i, j),$$

où $\alpha \in [0, 1]$ détermine l'importance relative des deux composantes. La modularité de ce calcul permet de modifier ou de substituer le module de synergie sans perturber la structure de la mise à jour des pondérations, rendant ainsi le système adaptable à divers types de données et de contextes.

B. Module “Update ω ”

Le **module de mise à jour des pondérations** est le cœur de la dynamique d'auto-organisation du SCN. Il applique les règles d'évolution des liens qui relient les entités entre elles. La règle de mise à jour additive classique s'exprime par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage** et τ représente un terme de **décroissance** qui pénalise une croissance excessive des liens. Dans certains cas, pour obtenir une dynamique plus exponentielle, on peut opter pour une variante multiplicative :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \exp(\eta[S(i, j) - \tau \omega_{i,j}(t)]).$$

Ce module est conçu pour être indépendant du module de calcul de la synergie ; il reçoit en entrée la matrice $\mathbf{S} = \{S(i, j)\}$ et les paramètres η et τ , et renvoie la nouvelle matrice de pondérations. Par cette séparation, il devient trivial de changer le schéma de mise à jour (passer d'un modèle additif à un modèle multiplicatif, ou encore y intégrer des mécanismes de clipping ou d'inhibition) sans affecter le calcul du score de synergie.

C. Module “Inhibition Global ou Local”

Dans des systèmes complexes, il est souvent nécessaire d'introduire une **inhibition** afin d'empêcher qu'une entité ne se lie de manière excessive à un trop grand nombre de partenaires. Cette inhibition peut être implémentée de deux manières principales :

- **Inhibition locale** : Chaque entité \mathcal{E}_i se voit imposer une contrainte sur la somme des pondérations associées à ses liens. Par exemple, on peut imposer que

$$\sum_{j \neq i} \omega_{i,j} \leq \Omega_{\max},$$

où Ω_{\max} est un seuil maximal. Cela force l'entité à concentrer ses liens sur les partenaires les plus synergiques.

- **Inhibition globale** : On ajoute directement un terme d'inhibition dans la règle de mise à jour, tel que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

où γ est un coefficient d'inhibition qui réduit la pondération si l'entité est déjà fortement connectée à plusieurs partenaires.

Ce module peut être conçu comme une fonction à part entière, qui s'applique en post-traitement de la mise à jour des $\omega_{i,j}$. En isolant ce mécanisme, on peut facilement expérimenter avec différentes stratégies d'inhibition (par exemple, en modifiant le seuil Ω_{\max} ou en ajustant le coefficient γ) sans impacter la logique de calcul de la synergie ou la mise à jour des pondérations.

D. Synergie d'Ensemble et Ordonnancement

L'architecture logicielle d'un SCN s'organise de manière séquentielle en un pipeline d'itération. À chaque itération t , le **moteur** de l'auto-organisation procède de la manière suivante :

Le **Module de Calcul de Synergie** est appelé pour déterminer les scores $S(i,j)$ pour toutes les paires d'entités.

Le **Module de Mise à Jour** applique la règle de mise à jour sur la matrice $\omega(t)$ pour générer $\omega(t+1)$.

Le **Module d'Inhibition** intervient ensuite pour appliquer des ajustements supplémentaires, tels que la régulation de la somme des liens ou le clipping.

Des **observateurs** ou **logs** sont activés pour suivre l'évolution de la matrice $\omega(t)$ et calculer des indicateurs de convergence, par exemple la norme de la différence $\|\omega(t+1) - \omega(t)\|$ ou des indices de modularité issus d'algorithmes de détection de communautés.

Ce séquençage permet d'intégrer aisément des mécanismes de **contrôle adaptatif**. Par exemple, un **Module de Paramètres** central peut ajuster dynamiquement η et τ en fonction des indicateurs de stabilité, ou activer un **recuit stochastique** pour sortir des minima locaux peu intéressants.

E. Avantages de la Modularisation

La division du système en modules distincts présente plusieurs **avantages** majeurs :

- **Clarté et maintenabilité** : Chaque module est développé et testé indépendamment, ce qui facilite la compréhension globale du système et simplifie les modifications ou mises à jour.
- **Flexibilité** : Il devient trivial de substituer ou d'améliorer un module particulier sans affecter les autres parties du SCN. Par exemple, l'algorithme de calcul de synergie peut être remplacé par une version hybride (fusionnant sub-symbolique et symbolique) tout en conservant la même interface avec le module de mise à jour des ω .
- **Extensibilité** : Dans des scénarios multi-entités ou multi-domaines, le même cadre de mise à jour peut être réutilisé en adaptant uniquement le module de calcul de synergie ou les paramètres de régulation, ce qui permet une intégration homogène de divers types de données et de logiques.
- **Contrôle et adaptation** : La structure modulaire permet l'implémentation de mécanismes de surveillance et de contrôle (inhibition, clipping, recuit) qui peuvent être activés ou désactivés indépendamment, offrant ainsi une gestion fine de la dynamique d'auto-organisation.

Conclusion

L'approche modulaire, telle que présentée dans cette section, est essentielle pour traduire les principes théoriques du SCN en une solution logicielle opérationnelle. En séparant distinctement le **calcul de la synergie**, la **mise à jour des pondérations** et les mécanismes d'**inhibition** ou de **régulation**, le système gagne en **clarté**, en **flexibilité** et en **extensibilité**. Ce découpage permet non seulement de maintenir une architecture propre et évolutive, mais aussi d'expérimenter différents schémas de mise à jour et de contrôler la dynamique du réseau par des ajustements paramétriques précis. En intégrant ces modules dans une boucle d'itération orchestrée par un moteur central, le SCN peut être déployé efficacement pour diverses applications, allant de la robotique collaborative à la gestion de systèmes multi-agents, en garantissant une auto-organisation robuste et adaptable aux exigences spécifiques de chaque domaine.

4.8.2. Gestions des Ressources et Threads

Dans un **SCN** (Synergistic Connection Network) à grande échelle, la boucle de mise à jour $\omega_{i,j}(t+1)$ peut rapidement devenir un **goulet d'étranglement** si elle s'exécute de façon **strictement séquentielle**. On peut souhaiter **paralléliser** certaines opérations ou recourir à des **threads** multiples, surtout si n (le nombre d'entités) se compte en milliers ou plus. Toutefois, cette parallélisation doit être orchestrée avec soin pour éviter les **incohérences** dans le calcul de $\Delta\omega_{i,j}$.

4.8.2.1. Mise à Jour Parallèle ou Asynchrone : Chaque Entité Calcule Localement $\Delta\omega_{i,j}$

Dans les systèmes de **Deep Synergy Learning (DSL)**, la mise à jour des pondérations $\omega_{i,j}(t)$ entre entités se réalise traditionnellement par le biais d'une double boucle itérative sur toutes les paires (i,j) . Cependant, lorsque le nombre d'entités n devient grand, le coût de calcul, de l'ordre de $\mathcal{O}(n^2)$ opérations par itération, peut rapidement devenir prohibitif. Pour pallier cette limitation, il est naturel d'envisager une **mise à jour parallèle ou asynchrone** des pondérations, dans laquelle chaque entité calcule localement sa contribution $\Delta\omega_{i,j}$ aux mises à jour, permettant ainsi de répartir la charge de calcul entre plusieurs threads ou processeurs. Nous présentons ici de manière détaillée les mécanismes, les avantages et les risques associés à cette approche.

A. Principe de la Mise à Jour Parallèle

L'idée fondamentale consiste à **diviser** le calcul de la matrice des pondérations $\mathbf{\omega}(t)$ en sous-tâches indépendantes, qui peuvent être exécutées simultanément sur différents cœurs ou processeurs. Soit n le nombre d'entités, la mise à jour standard s'exprime par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ représente la synergie entre les entités i et j , η est le taux d'apprentissage et τ le coefficient de décroissance. Pour paralléliser ce calcul, plusieurs approches sont possibles. Par exemple, l'on peut découper la matrice $\mathbf{\omega}$ en **sous-blocs** ou en attribuant à chaque entité i la responsabilité de mettre à jour tous les liens sortants $\omega_{i,j}$ pour $j = 1, \dots, n$. Formellement, chaque entité réalise le calcul :

$$\Delta\omega_{i,j} = \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

puis met à jour localement

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta\omega_{i,j}.$$

L'utilisation d'un **double-buffer** est souvent recommandée pour éviter les conflits d'accès en écriture. On définit ainsi une matrice $\mathbf{w}^{(t+1)}$ dans laquelle chaque thread écrit ses mises à jour, alors que la matrice $\mathbf{w}^{(t)}$ est lue en lecture seule pendant toute l'itération.

B. Synchronous vs. Asynchronous Update

Dans le **mode synchrone**, toutes les entités ou tous les threads réalisent leur calcul de $\Delta\omega_{i,j}$ en se basant sur la même version de la matrice $\omega(t)$. La procédure se décompose en deux phases distinctes :

- **Phase de lecture** : Chaque thread lit la matrice $\omega(t)$ pour calculer les variations locales $\Delta\omega_{i,j}$ pour les indices qui lui sont assignés.
- **Phase de mise à jour** : Une fois que toutes les variations ont été calculées, un **barrier** ou **synchronisation** est appliquée, puis la nouvelle matrice $\omega(t+1)$ est construite en copiant les résultats du buffer de sortie.

Mathématiquement, on définit deux matrices $\mathbf{w}^{(t)}$ et $\mathbf{w}^{(t+1)}$ telles que, pour tous i et j ,

$$\mathbf{w}_{i,j}^{(t+1)} = \mathbf{w}_{i,j}^{(t)} + \eta [S(i,j) - \tau \mathbf{w}_{i,j}^{(t)}].$$

Ce mode garantit une **cohérence** absolue puisque chaque entité travaille sur une version identique de $\omega(t)$. Le principal inconvénient réside dans la nécessité d'attendre que tous les threads aient terminé leur calcul, ce qui peut introduire un délai de synchronisation dans des environnements massivement parallèles.

En **mode asynchrone**, chaque entité met à jour ses propres liens dès que son calcul de $\Delta\omega_{i,j}$ est terminé, sans attendre la fin de l'itération pour l'ensemble du réseau. Cela peut se formaliser par l'équation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où la lecture et l'écriture de $\omega_{i,j}$ se font de manière **concurrente** et **lock-free**. Dans ce contexte, il arrive que différentes entités lisent des versions légèrement différentes de la matrice ω (certaines mises à jour étant déjà effectuées tandis que d'autres non). Ce mode peut accélérer le temps de calcul global, surtout dans des architectures distribuées, mais il soulève des problèmes de **cohérence** et de **stabilité**. Par exemple, une mise à jour asynchrone peut être analysée comme une itération de type Gauss-Seidel, où l'**ordre** d'exécution influe sur la convergence. Des techniques telles que le **verrouillage léger** ou l'emploi de structures atomiques peuvent être nécessaires pour minimiser les conflits, sans pour autant imposer une synchronisation complète.

C. Avantages et Risques du Calcul Parallèle/Asynchrone

L'**avantage principal** de la mise à jour parallèle ou asynchrone réside dans la **réduction du temps de calcul** global. En divisant la charge de mise à jour sur plusieurs threads, le coût de $\mathcal{O}(n^2)$ opérations peut être réduit de manière quasi linéaire avec le nombre de processeurs disponibles, permettant ainsi d'aborder des systèmes à grande échelle. Par ailleurs, un mode asynchrone, s'il est correctement contrôlé, permet une **adaptation continue** et une mise à jour plus fluide de ω , même dans des environnements distribués.

Cependant, ces gains de performance ne vont pas sans risques. En mode asynchrone, l'absence d'une synchronisation stricte peut conduire à des **incohérences** où certaines mises à jour sont appliquées sur des données déjà modifiées, générant des écarts imprévus et potentiellement des oscillations dans les valeurs des pondérations. La garantie théorique de convergence devient alors plus complexe, nécessitant l'emploi d'algorithmes de contrôle tels que les techniques basées sur le **Gossip** ou des schémas de verrouillage léger pour assurer que la dynamique globale demeure stable.

D. Stratégies pour la Mise en Œuvre Asynchrone

Afin d'assurer la **stabilité** tout en bénéficiant des avantages du parallélisme, plusieurs stratégies peuvent être mises en œuvre :

Double Buffering :

Chaque itération lit une copie immuable de la matrice $\mathbf{w}^{(t)}$ pour effectuer les calculs et écrit les résultats dans une nouvelle matrice $\mathbf{w}^{(t+1)}$. Cette approche garantit que toutes les mises

à jour de la phase t sont basées sur des données cohérentes, éliminant ainsi les conflits d'accès simultanés.

Utilisation de Verrous Atomiques ou Lock-Free :

Dans des environnements multi-threadés, l'emploi de verrous atomiques permet de s'assurer que les mises à jour sur un élément $\omega_{i,j}$ ne sont pas interrompues ou écrasées par d'autres threads. Bien que cette approche puisse introduire un certain coût en termes de performance, elle offre une **garantie** de cohérence. Des algorithmes lock-free sophistiqués, inspirés des techniques de la programmation concurrente, peuvent également être envisagés pour minimiser les surcharges tout en maintenant la stabilité.

Décomposition en Blocs :

En divisant la matrice \mathbf{w} en sous-blocs disjoints, chaque thread peut être affecté à un bloc spécifique, réduisant ainsi les risques de conflits. Cette approche est particulièrement efficace si l'on peut isoler des sous-ensembles d'entités ayant peu d'interaction avec d'autres sous-ensembles.

Ajustement Dynamique des Paramètres :

Les paramètres η et τ jouent un rôle crucial dans la vitesse de convergence et la stabilité. Un **Module de Gestion des Paramètres** peut surveiller en temps réel la variation des pondérations et ajuster η de manière à atténuer les oscillations lorsque des mises à jour asynchrones induisent des incohérences.

E. Conclusion

La mise à jour parallèle ou asynchrone dans un SCN représente une approche essentielle pour rendre le déploiement d'un **DSL** scalable et efficace, en particulier lorsque le nombre d'entités n est élevé. En permettant à chaque entité ou à chaque bloc d'entités de calculer localement $\Delta\omega_{i,j}$ et de mettre à jour les pondérations de manière simultanée, on réduit significativement le temps de calcul global. Toutefois, cette approche soulève des défis importants en termes de **cohérence** et de **stabilité** de la dynamique, nécessitant des stratégies telles que le double buffering, l'utilisation de verrous atomiques, la décomposition en blocs et l'ajustement dynamique des paramètres. L'objectif ultime est de garantir que, malgré la nature asynchrone des mises à jour, le système converge vers une configuration stable et fiable, tout en profitant des avantages du parallélisme pour traiter de très grands réseaux d'entités en temps réel. Cette méthode opérationnelle est le pilier qui permet au DSL de s'adapter aux environnements complexes et distribués, assurant ainsi une auto-organisation robuste et scalable.

4.8.2.2. Conflits Possibles, Besoin de Cohérence Globale (Verrous, ou Algorithmes Lock-Free ?)

Dans un environnement de mise à jour parallèle ou asynchrone d'un **Synergistic Connection Network (SCN)**, la **cohérence** globale des pondérations $\omega_{i,j}$ est mise à rude épreuve par la concurrence des opérations de lecture et d'écriture. Dès lors que plusieurs threads ou processus

interviennent simultanément pour calculer les incrémentations $\Delta\omega_{i,j}$ et mettre à jour les valeurs correspondantes, il apparaît impératif d'assurer une synchronisation adéquate afin de préserver l'intégrité de la dynamique globale. Dans cette section, nous analysons les conflits qui peuvent surgir, les approches traditionnelles basées sur des verrous (mutex) et les alternatives lock-free, et nous discutons de leur impact sur la stabilité et la convergence du SCN.

A. Nature des Conflits en Mise à Jour Concurente

Le SCN repose sur la règle de mise à jour additive donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ représente le score de synergie entre les entités i et j . Dans un environnement parallèle, la matrice ω est partagée entre plusieurs threads, et plusieurs d'entre eux peuvent tenter de lire ou d'écrire simultanément la même valeur $\omega_{i,j}(t)$. Par exemple, si un thread lit $\omega_{i,j}(t)$ pour calculer $\Delta\omega_{i,j}$ pendant qu'un autre thread a déjà effectué une mise à jour sur la même case, alors la valeur utilisée pour le calcul peut être obsolète ou partiellement mise à jour. Ce phénomène de **races conditions** (conditions de concurrence) peut conduire à des incohérences, notamment lorsque la somme d'une inhibition globale dépend de la lecture simultanée de plusieurs $\omega_{i,k}(t)$. En d'autres termes, des mises à jour concurrentes non coordonnées peuvent aboutir à une matrice qui ne reflète plus fidèlement la dynamique théorique du système, générant ainsi des déphasages, des oscillations imprévues ou même la divergence de la procédure de convergence.

B. Stratégies de Synchronisation Basées sur les Verrous

La méthode la plus intuitive pour prévenir ces conflits est d'utiliser des **verrous** (*mutex*), lesquels garantissent que lorsqu'un thread est en train d'accéder ou de modifier une partie de la matrice ω , aucun autre thread n'y a accès. Deux approches se distinguent :

- **Verrous Globaux** : On peut appliquer un verrou global sur l'ensemble de la matrice ω pendant la phase de mise à jour. Autrement dit, dès qu'un thread commence à mettre à jour une valeur $\omega_{i,j}$, il acquiert un verrou qui empêche toute autre modification sur ω jusqu'à la fin de l'itération. Cette approche garantit une cohérence parfaite, mais elle réduit fortement le parallélisme, car les autres threads doivent attendre la libération du verrou global.
- **Verrous Fins** : Une approche plus raffinée consiste à appliquer des verrous sur des sous-ensembles de la matrice, par exemple sur chaque ligne ou sur des blocs spécifiques. Ainsi, si un thread met à jour la ligne i de la matrice, il acquiert un verrou sur cette ligne, tandis que d'autres threads peuvent mettre à jour des lignes différentes simultanément. Cette granularité fine permet une meilleure **concurrence** tout en assurant une cohérence locale, même si la gestion de multiples verrous peut devenir complexe dans un système à très haute dimension.

La mise en œuvre des verrous s'appuie souvent sur des primitives fournies par les bibliothèques de concurrence (par exemple, le module *threading* en Python ou les classes *std::mutex* en C++). Bien que cette approche soit efficace pour assurer la cohérence, elle peut introduire un surcoût en termes de **latence** et de **contention**, en particulier lorsque le nombre d'entités est grand.

C. Approches Lock-Free et Algorithmes Gossip

Pour surmonter les limitations imposées par les verrous, des approches **lock-free** ont été développées. Ces méthodes reposent sur l'utilisation d'opérations atomiques, telles que le **compare-and-swap (CAS)**, qui permettent de mettre à jour les valeurs de $\omega_{i,j}$ sans verrou explicite. L'idée est de garantir que chaque opération de mise à jour soit effectuée de manière indivisible, de sorte qu'un thread vérifie que la valeur de $\omega_{i,j}$ n'a pas changé depuis sa lecture avant d'y écrire la nouvelle valeur. Formulée de manière schématique, une mise à jour lock-free s'effectue ainsi :

if $\omega_{i,j} = \text{old_value}$ then $\omega_{i,j} \leftarrow \text{new_value}$ (atomiquement).

Cette approche permet une **concurrence** très élevée, particulièrement dans des environnements distribués ou sur des architectures multicœurs. Toutefois, la garantie de **convergence** en mode lock-free nécessite des conditions strictes, notamment un paramétrage très faible de η et τ pour limiter les déphasages. Par ailleurs, la complexité de l'analyse mathématique augmente, car la dynamique asynchrone introduit des retards variables et des mises à jour non uniformes, qui peuvent, dans certains cas, conduire à des oscillations accrues.

Une autre méthode est l'utilisation d'algorithmes **Gossip-based**, dans lesquels chaque nœud partage ses mises à jour de manière probabiliste et locale avec ses voisins. Cette approche permet de propager l'information de manière distribuée, avec des garanties de convergence sous certaines conditions sur le réseau de communication. Cependant, les algorithmes de type Gossip introduisent également des retards et des incohérences temporaires qui doivent être absorbés par la dynamique globale du SCN.

D. Impact sur la Convergence et la Stabilité Globale

Le choix entre des verrous, des approches lock-free ou des algorithmes Gossip influence directement la **convergence** du SCN. En mode synchrone, la garantie de convergence est aisée à démontrer grâce à l'hypothèse que toutes les mises à jour se font sur une version immuable $\omega(t)$. Le double-buffering, qui consiste à utiliser une matrice $\omega^{(t)}$ en lecture seule et une autre $\omega^{(t+1)}$ pour les écritures, est souvent utilisé pour préserver la **cohérence** tout en permettant une parallelisation. En mode asynchrone, la lecture de valeurs légèrement décalées peut induire des fluctuations dans les mises à jour. Pour assurer la stabilité, il est alors nécessaire d'ajuster le taux d'apprentissage η pour qu'il soit suffisamment faible, garantissant que même en présence de retards, l'opération de mise à jour reste contractante dans la norme choisie (i.e., $|1 - \eta \tau| < 1$).

L'utilisation de techniques de contrôle, telles que l'injection de bruit contrôlé (similaire à un recuit stochastique) ou l'application périodique de mécanismes de réindexation, peut aider à compenser les déphasages induits par une mise à jour asynchrone. Ainsi, l'architecture lock-free, bien que plus rapide en termes de débit de mise à jour, requiert une analyse plus fine des conditions de convergence, et peut parfois se traduire par des comportements oscillatoires ou des divergences temporaires qu'il faudra ensuite amortir par des correctifs algorithmiques.

E. Conclusion

La mise à jour parallèle ou asynchrone des pondérations $\omega_{i,j}$ dans un SCN offre un gain substantiel en termes de performance lorsque le nombre d'entités est élevé. Cependant, cette accélération

s'accompagne de défis majeurs en matière de cohérence et de stabilité globale. L'emploi de verrous (globaux ou fins) permet d'assurer une cohérence stricte, mais au prix d'une réduction de la parallélisation. À l'inverse, les techniques lock-free et les algorithmes Gossip, tout en maximisant la rapidité d'exécution, introduisent des incohérences temporaires qui nécessitent un paramétrage prudent (réduction de η , ajustement de τ) pour garantir la convergence du système vers des attracteurs stables. Le choix final entre ces stratégies dépend largement de la taille du réseau, du degré de distribution souhaité et des contraintes de performance spécifiques à l'application. En pratique, une approche hybride, souvent basée sur le double-buffering dans un mode synchrone, s'avère être un compromis efficace, car elle assure une cohérence globale tout en tirant parti du parallélisme offert par les architectures modernes. Cette solution opérationnelle constitue le socle sur lequel repose le déploiement robuste et scalable d'un SCN dans des environnements multi-threadés ou distribués.

4.8.3. Transition

Les sections précédentes (4.8.1, 4.8.2) ont souligné l'importance de l'**architecture logicielle** (modularisation, gestion des ressources, parallélisme) pour soutenir la dynamique d'un **SCN** (Synergistic Connection Network). Il apparaît clairement qu'au-delà de la simple mise à jour $\omega_{i,j}$, il faut un cadre organisationnel robuste : comment instancier les modules de synergie, comment gérer l'inhibition, comment orchestrer la boucle de calcul en mode synchrone ou asynchrone, etc.

4.8.3.1. Ce qu'on Pourrait Coder en Python/C++ pour Reproduire cette Dynamique en Quelques Centaines de Lignes

Dans cette section, nous présentons une approche concrète pour implémenter la dynamique d'un **Synergistic Connection Network (SCN)** dans un langage de programmation tel que Python ou C++. L'objectif est de traduire la théorie développée dans les chapitres précédents en un code minimaliste mais complet, permettant de simuler la mise à jour des pondérations $\omega_{i,j}(t)$ et d'observer l'émergence de structures telles que des clusters, en quelques centaines de lignes. Cette démarche met en œuvre la règle de mise à jour additive ainsi que des mécanismes complémentaires comme l'inhibition et le clipping, tout en assurant une modularisation pour une meilleure maintenance.

A. Définition des Paramètres et Initialisation

Le système comporte n entités, chacune identifiée par un indice $i \in \{1, 2, \dots, n\}$. La dynamique du SCN est régie par la règle de mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

où :

- η représente le **taux d'apprentissage**,
- τ est le **facteur de décroissance** qui module la pénalisation d'un lien trop élevé,

- γ est le **coefficent d'inhibition** destiné à limiter la prolifération des connexions non sélectives,
- $S(i,j)$ désigne le score de **synergie** entre les entités i et j .

Le premier module du code consiste à définir ces paramètres et à initialiser la matrice de pondérations $\omega(0)$. Par convention, on initialise $\omega_{i,j}(0)$ à des valeurs faibles (par exemple, un bruit aléatoire autour de zéro) pour simuler l'état initial d'un système non organisé.

En Python, on utilisera la bibliothèque NumPy pour manipuler efficacement les matrices. Par exemple :

```
import numpy as np
```

```
# Nombre d'entités
```

```
n = 15
```

```
# Paramètres globaux
```

```
eta = 0.05    # Taux d'apprentissage
tau = 1.0     # Facteur de décroissance
gamma = 0.02   # Coefficient d'inhibition
T_max = 300    # Nombre d'itérations
```

```
# Initialisation de la matrice des pondérations, avec un léger bruit aléatoire
```

```
np.random.seed(42)
omega = np.random.uniform(0.0, 0.05, (n, n))
np.fill_diagonal(omega, 0.0)
```

```
# Initialisation de la matrice de synergie S (à définir selon le contexte)
```

```
S = np.zeros((n, n))
# Par exemple, pour simplifier, on peut définir S(i,j) statique selon des critères prédéfinis
# Ici, S est symétrique et les valeurs sont entre 0 et 1.
for i in range(n):
    for j in range(n):
        if i != j:
            S[i, j] = np.random.uniform(0.1, 0.8)
```

B. Implémentation de la Boucle de Mise à Jour

La dynamique d'auto-organisation est réalisée via une boucle itérative. À chaque itération t , la mise à jour des pondérations se fait de manière synchrone sur l'ensemble des paires (i,j) . Afin d'éviter les conflits de lecture-écriture, on utilise le **double-buffering** : la matrice $\omega(t)$ est lue en lecture seule, tandis que les nouvelles valeurs sont écrites dans un buffer temporaire ω_{next} .

La mise à jour additive est alors formulée comme suit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Le pseudo-code Python ci-dessous illustre ce processus :

```
# Liste pour enregistrer l'évolution de la moyenne des pondérations (pour monitoring)
omega_means = []

# Boucle principale de mise à jour
for t in range(T_max):
    omega_next = np.copy(omega)
    for i in range(n):
        for j in range(n):
            if i != j:
                # Calcul de la variation (delta) de omega[i,j]
                delta = eta * (S[i, j] - tau * omega[i, j])
                # Application de l'inhibition : somme des pondérations de la ligne i sauf pour j
                inhibition = gamma * (np.sum(omega[i, :]) - omega[i, j])
                # Mise à jour additive
                omega_next[i, j] = omega[i, j] + delta - inhibition
                # Clipping pour garantir que omega reste positif
                omega_next[i, j] = max(0.0, omega_next[i, j])
        omega = omega_next
    # Enregistrement d'une statistique pour monitoring
    omega_means.append(np.mean(omega))

# À la fin de la boucle, omega contient la configuration finale du SCN.
```

Ce code, bien que succinct, encapsule le cœur de la **dynamique auto-organisée** du SCN. On observe que la mise à jour est effectuée de manière synchrone via un double-buffering, assurant la cohérence de la lecture de $\omega(t)$ pendant toute l'itération.

C. Visualisation et Analyse des Résultats

Pour évaluer la **convergence** et la **formation des clusters**, il est utile de visualiser l'évolution de la matrice ω et des indicateurs tels que la moyenne ou la variance des pondérations. En Python, on peut utiliser des bibliothèques telles que Matplotlib pour tracer des graphiques en temps réel ou après simulation :

```
import matplotlib.pyplot as plt

# Tracé de la moyenne des pondérations au cours des itérations
plt.figure(figsize=(8, 5))
plt.plot(omega_means, label='Moyenne de $\omega(t)$')
plt.xlabel('Itérations (t)')
plt.ylabel('Valeur moyenne de $\omega$')
plt.title('Convergence de la matrice de pondérations')
plt.legend()
plt.grid(True)
plt.show()
```

De plus, pour une analyse plus fine, on peut générer une **heatmap** de la matrice ω à la fin de la simulation pour identifier visuellement les clusters :

```
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.heatmap(omega, cmap='viridis', annot=True, fmt=".2f")
plt.title('Heatmap finale de la matrice $\omega$')
plt.xlabel('Entités')
plt.ylabel('Entités')
plt.show()
```

Ces visualisations permettent de confirmer que les pondérations se sont organisées en clusters stables, conformément aux attentes théoriques.

D. Extensions Possibles et Adaptations en C++

La même logique de mise à jour peut être implémentée en C++ en utilisant, par exemple, la bibliothèque STL et des structures de données adaptées (comme `std::vector<std::vector<double>>` pour représenter ω). La boucle de mise à jour sera similaire à celle présentée en Python, avec des boucles `for` imbriquées pour parcourir les indices i et j . Voici un exemple schématique en C++ :

```
#include <vector>
#include <iostream>
#include <algorithm>
#include <cstdlib>

// Définition des paramètres
const int n = 15;
const int T_max = 300;
const double eta = 0.05;
const double tau = 1.0;
const double gamma = 0.02;

// Fonction pour initialiser la matrice avec un bruit faible
std::vector<std::vector<double>> initializeMatrix(int n) {
    std::vector<std::vector<double>> omega(n, std::vector<double>(n, 0.0));
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j) {
                omega[i][j] = ((double) rand() / RAND_MAX) * 0.05;
            }
        }
    }
    return omega;
}
```

```

// La fonction principale de mise à jour
void updateOmega(std::vector<std::vector<double>> &omega, const std::vector<std::vector<double>> &S) {
    int n = omega.size();
    std::vector<std::vector<double>> omega_next = omega;
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j) {
                double sum_inhibition = 0.0;
                for (int k = 0; k < n; ++k) {
                    if (k != j) {
                        sum_inhibition += omega[i][k];
                    }
                }
                double delta = eta * (S[i][j] - tau * omega[i][j]) - gamma * sum_inhibition;
                omega_next[i][j] = std::max(0.0, omega[i][j] + delta);
            }
        }
    }
    omega = omega_next;
}

int main() {
    // Initialisation des matrices omega et S
    std::vector<std::vector<double>> omega = initializeMatrix(n);
    std::vector<std::vector<double>> S(n, std::vector<double>(n, 0.0));
    // Remplissage simple de S, par exemple avec des valeurs aléatoires entre 0.1 et 0.8
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            if (i != j) {
                S[i][j] = 0.1 + ((double) rand() / RAND_MAX) * 0.7;
            }
        }
    }

    // Boucle de mise à jour
    for (int t = 0; t < T_max; ++t) {
        updateOmega(omega, S);
    }

    // Affichage final de la matrice omega
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < n; ++j) {
            std::cout << omega[i][j] << " ";
        }
        std::cout << std::endl;
    }
}

```

```

    return 0;
}

```

Ce code C++ illustre la structure de base du pipeline de mise à jour, de la même manière que la version Python, et démontre que la **logique** sous-jacente est indépendante du langage de programmation. La modularité du code permet ensuite d'y intégrer facilement des mécanismes de **visualisation** ou d'**exportation** des résultats pour une analyse plus poussée.

E. Conclusion

L'implémentation d'un SCN dynamique à l'aide d'un code en Python ou en C++ repose sur un ensemble de modules bien définis : l'initialisation des matrices de synergie et de pondérations, la boucle itérative de mise à jour appliquant la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

et enfin des étapes de régulation telles que le clipping et la sparsification. Cette approche permet de reproduire, en quelques centaines de lignes de code, la dynamique auto-organisée décrite théoriquement dans le cadre d'un **Deep Synergy Learning**. La simplicité conceptuelle de la mise à jour est ainsi transformée en une architecture logicielle modulaire, facilitant la maintenance, l'extension à des systèmes plus vastes et l'intégration dans des frameworks multi-agents ou robotiques. Les exemples de pseudo-code présentés en Python et en C++ servent de point de départ pour des développements plus avancés, dans lesquels l'injection de visualisations, la gestion dynamique des paramètres et la persistance des données viendront enrichir l'ensemble et permettre des expérimentations en temps réel ou sur de grands volumes de données.

Voici une proposition détaillée et approfondie, rédigée dans un style académique et structuré, qui intègre de nombreuses formules mathématiques et explications approfondies.

4.8.3.2. Les Concepts de ce Chapitre 4 (Mise à Jour, Inhibition, Clusters Émergents) s'Inséreront dans une Architecture Logicielle Solide

L'ensemble des notions développées dans le chapitre 4 – notamment la dynamique de mise à jour des pondérations $\omega_{i,j}(t+1)$, les mécanismes d'inhibition (locaux ou globaux) et l'émergence des clusters – doit être opérationnalisé dans une infrastructure logicielle modulaire pour permettre un déploiement à grande échelle et une maintenance aisée. Le passage de la théorie à la pratique exige la conception d'un framework où chaque composant de la dynamique est isolé sous forme de modules interchangeables. Cette approche facilite non seulement l'évolution et l'extension du code, mais également l'intégration de nouvelles méthodes de calcul ou de nouveaux algorithmes de contrôle. Nous détaillons ci-après les axes majeurs de cette intégration.

A. Intégration Modulaire des Composants de la Dynamique

Dans un premier temps, il convient de séparer distinctement les responsabilités fonctionnelles de la dynamique du SCN. On distingue principalement trois modules :

- **Module de Calcul de Synergie** : Ce module est chargé de déterminer la fonction $S(i, j)$, qui représente la qualité ou l'intensité de la synergie entre deux entités \mathcal{E}_i et \mathcal{E}_j . La fonction $S(i, j)$ peut être définie à partir de critères sub-symboliques (par exemple, la similarité cosinus ou une distance gaussienne entre embeddings) ou symboliques (compatibilité de règles ou axiomes). On peut formuler, pour une approche hybride, la synergie comme

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où $\alpha \in [0, 1]$ module l'importance relative des composantes. Ce module est conçu pour être interfaçable avec différents types de données et peut être mis à jour indépendamment, par exemple en recalculant périodiquement les embeddings ou en ajustant les règles symboliques.

- **Module de Mise à Jour des Pondérations ω** : Ce module encapsule la règle d'auto-organisation appliquée à la matrice ω . Qu'il s'agisse d'un schéma additif classique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

ou d'une version multiplicative (par exemple,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \exp(\eta [S(i, j) - \tau \omega_{i,j}(t)]),$$

ce module reçoit en entrée la matrice S issue du module précédent, ainsi que les paramètres η et τ , et produit la nouvelle matrice $\omega(t+1)$. L'architecture modulaire permet de remplacer aisément la règle de mise à jour en modifiant uniquement le code de ce module, sans impacter le calcul de la synergie.

- **Module de Régulation (Inhibition, Sparsification, Clipping)** : Pour éviter une croissance non contrôlée des pondérations et pour favoriser la formation de clusters cohérents, il est souvent nécessaire d'introduire des mécanismes d'inhibition et de clipping. Par exemple, l'ajout d'un terme inhibiteur tel que

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t)$$

dans la mise à jour permet de pénaliser la distribution excessive de liens par une entité. Ce module peut être implémenté soit comme une étape intégrée dans le module de mise à jour, soit comme une étape postérieure appliquant un filtrage sur la matrice ω . Le recours à une méthode de clipping, où l'on borne $\omega_{i,j}(t)$ à une valeur maximale ω_{\max} , peut également être intégré ici pour éviter des valeurs aberrantes. Cette modularisation garantit qu'en cas de besoin, le comportement de régulation pourra être ajusté sans modifier les autres composants du système.

Ces modules interagissent au sein d'un **pipeline** orchestré par un **engine** central, dont le rôle est de gérer la séquence d'exécution, la synchronisation (via par exemple un double-buffering pour garantir la cohérence) et la persistance des données entre les itérations. Le code principal de ce

pipeline appelle successivement le module de calcul de synergie, puis le module de mise à jour des pondérations, et enfin le module de régulation, avant de stocker le résultat ou de déclencher une phase de visualisation.

B. Gestion Dynamique et Paramétrage

Une des forces de l'architecture modulaire est la capacité à gérer dynamiquement les paramètres η , τ et γ . Un **Param Manager** est ainsi mis en place, centralisant la configuration de ces valeurs. Ce gestionnaire peut, par exemple, surveiller la convergence en calculant à chaque itération des indicateurs tels que la norme $\|\omega(t+1) - \omega(t)\|$ ou la variation de l'énergie potentielle $J(\omega)$. Si une oscillation excessive est détectée, le Param Manager peut alors automatiquement réduire η (ou augmenter τ) pour stabiliser la dynamique. La flexibilité apportée par ce module permet non seulement d'ajuster les paramètres de manière adaptative en fonction des données entrantes, mais également de tester divers schémas de mise à jour sans réécriture globale du code.

C. Persistance et Interface d'Intégration

La persistance des données dans un SCN est cruciale, surtout lorsqu'on déploie le système sur des volumes de données conséquents ou sur des plateformes distribuées. Une fois que la boucle de mise à jour est exécutée, la matrice ω doit être sauvegardée pour permettre une reprise ultérieure ou une analyse post-hoc. Ce processus s'effectue via des modules de sauvegarde qui peuvent écrire la matrice dans un format binaire optimisé (par exemple, en utilisant le format HDF5) ou en format sparse si la majorité des liens tend à être nulle. Par ailleurs, une API d'intégration permet de brancher le SCN au sein d'un framework plus large – par exemple, pour interfaçer avec des simulateurs robotiques ou des systèmes de monitoring en temps réel. Ce type d'interface se traduit par des endpoints REST ou par l'injection de flux de données dans le SCN, facilitant ainsi son déploiement dans des environnements industriels.

D. Orchestration de la Dynamique par un Engine Central

Le cœur de l'implémentation repose sur une boucle maîtresse ou un **scheduler** qui orchestre les appels successifs aux modules. Ce scheduler, implémenté par exemple en Python via des bibliothèques de multi-threading (comme *multiprocessing* ou *concurrent.futures*), effectue les opérations suivantes à chaque itération t :

- Il appelle le module de **calcul de synergie** pour obtenir la matrice $S(t)$ ou pour la mettre à jour en fonction des nouvelles données.
- Il déclenche le module de **mise à jour de ω** en passant les paramètres courants et la matrice $S(t)$, puis recueille la nouvelle matrice $\omega(t+1)$ via un système de double-buffering afin d'assurer une lecture cohérente.
- Il active ensuite le module de **régulation** (inhibition et clipping) pour appliquer des ajustements supplémentaires sur la matrice $\omega(t+1)$.
- Enfin, il enregistre l'état actuel de $\omega(t+1)$ pour permettre des analyses ultérieures (par exemple, via des heatmaps ou des calculs d'indicateurs de modularité).

Cet ensemble d'opérations, en étant modulé et orchestré par un engine central, garantit que la **dynamique globale** du SCN reste stable, tout en permettant une adaptation rapide aux changements de paramètres ou à l'insertion de nouveaux modules.

E. Conclusion

L'architecture logicielle modulaire décrite ici transforme la théorie du SCN en un système robuste, extensible et maintenable. En séparant clairement le **calcul de synergie**, la **mise à jour de ω** et la **régulation** (via inhibition et clipping), le framework permet une substitution rapide de chaque composant en fonction des besoins spécifiques de l'application. La gestion dynamique des paramètres, la persistance des matrices et l'intégration d'un engine central pour orchestrer la boucle d'auto-organisation font de ce système un outil flexible, capable d'être intégré dans des environnements multi-agents, distribués ou industriels. Le **Chapitre 5** détaillera davantage ces aspects, en fournissant des spécifications d'API, des exemples de modules en C++ et en Python, ainsi que des études de cas sur l'intégration du SCN dans un framework plus large. Cette modularisation permet ainsi de préserver à la fois la **souplesse** théorique du SCN et la **stabilité** pratique nécessaire pour un déploiement réel, facilitant ainsi l'expérimentation et l'évolution continue du système.

4.9. Conclusion et Transition

Tout au long de ce **chapitre 4**, nous avons examiné en détail comment un **SCN** (Synergistic Connection Network) s'**auto-organise** et fait émerger des **clusters** cohérents via la mise à jour adaptative des pondérations $\omega_{i,j}$. Les divers mécanismes (inhibition, saturation, injection de bruit, parallélisation...) ont montré la **richesse** et la **flexibilité** de cette dynamique. Il est maintenant temps de faire la **synthèse** (4.9.1) et de préparer le passage vers les **chapitres** suivants (4.9.2 et 4.9.3) où l'on abordera l'architecture globale (Chap. 5) et des développements plus avancés (Chaps. 6, 7, 8).

4.9.1. Synthèse

Dans l'ensemble des développements qui précèdent, le **cœur** de la dynamique d'un **Synergistic Connection Network** (SCN) repose sur la mise à jour locale des pondérations $\omega_{i,j}(t)$ selon des règles inspirées du schéma $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$. Une telle relation, parfois agrémentée d'un terme d'**inhibition** ou d'un **recuit**, définit un processus d'**auto-organisation** qui tend à renforcer les liens $\omega_{i,j}$ pour lesquels la **synergie** $S(i,j)$ est jugée élevée, tout en affaiblissant ceux qui se révèlent peu utiles ou inconsistants. Il est apparu, au fil des sections, que cette mécanique engendre la **formation** de **clusters** cohésifs, c'est-à-dire des ensembles d'entités fortement connectées par des pondérations $\omega_{i,j}$ élevées, alors que les connexions intergroupes demeurent faibles ou presque nulles.

La règle additive simple, $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$, se situe au fondement du processus. Dans un régime stationnaire, elle conduit souvent à un équilibre $\omega_{i,j}^* \approx S(i,j)/\tau$ pour chaque paire (i,j) , à condition que les entités n'évoluent pas et que les paramètres restent constants. Les **variantes** multiplicatives ou inhibitrices enrichissent ce tableau, en introduisant soit des effets de proportionnalité (cas multiplicatif) soit des phénomènes de compétition plus marqués (cas d'une inhibition latérale ou globale). Il a également été mis en évidence que la présence de **recuit** ou l'injection de bruit pouvaient aider à franchir des barrières ou à sortir de minima locaux, favorisant un **cluster** plus pertinent à l'échelle globale.

Le **contrôle** des oscillations se révèle un aspect capital pour la stabilité. Un choix cohérent de η (taux d'apprentissage), τ (décroissance) et γ (inhibition) circonscrit la zone d'attraction évitant les emballements, tout en permettant une différenciation nette des liaisons fortes et faibles. Il subsiste néanmoins une possibilité de **multi-stabilité** lorsque plusieurs configurations de clusters satisfont la matrice de synergie **S**. Le système peut alors basculer vers l'un ou l'autre attracteur suivant la condition initiale ou les perturbations reçues au fil des itérations.

À l'issue de cette analyse, il apparaît clairement que la dynamique d'un SCN ne saurait se réduire à une suite d'équations isolées. Dans un contexte pratique ou à grande échelle, elle doit s'inscrire dans une **architecture logicielle** modulaire, où l'on distingue le **module** affecté au **calcul** de la synergie, le **module** chargé de la **mise à jour** ω , le **module** gérant l'**inhibition** (ou la sparsification), et enfin les composantes d'**observation** ou de **clustering** (détection de communautés, visualisation). Cette séparation en blocs distincts, qui sera explicitée plus avant dans le chapitre 5, facilite la **maintenance**, l'**évolution** et la **persistent** des données, tout en permettant

d'orchestrer de manière ordonnée la boucle itérative : mise à jour parallèle, double-buffer, logs, etc.

En conclusion, la **mise à jour** $\omega_{i,j}(t + 1)$, complétée par les mécanismes d'**inhibition**, de **recuit** ou de **sparsification**, constitue le **noyau** d'un **SCN** permettant d'aboutir à des **clusters** stables ou à un régime oscillatoire si les paramètres le favorisent. La prochaine étape, telle que le décrit le **chapitre 5**, est de **concrétiser** cette dynamique dans un cadre logiciel structuré, garantissant la **cohérence** des opérations, la **scalabilité** face à un grand nombre d'entités et l'**interopérabilité** avec d'autres modules ou frameworks. Cette articulation entre la **théorie** (chapitre 4) et l'**architecture** (chapitre 5) rend possible l'**implémentation** efficace d'un SCN, depuis les principes de base jusqu'au déploiement à grande échelle.

4.9.2. Ouverture

Les développements du présent chapitre 4 ont présenté la **dynamique** d'un **Synergistic Connection Network** (SCN), en détaillant les principales règles de **mise à jour** $\omega_{i,j}(t + 1)$ (qu'elles soient additives ou multiplicatives), l'**inhibition** sous diverses formes, les mécanismes de **recuit** ou de **sparsification**, ainsi que l'**émergence** de **clusters** et les questions de **parallélisation**. Ces éléments constituent les fondations d'un système d'**auto-organisation** robuste. Néanmoins, ils ne sont que la première étape vers un **DSL** (*Deep Synergy Learning*) pleinement intégré. Les chapitres ultérieurs, à commencer par le **Chapitre 5**, franchiront un palier supplémentaire en décrivant la manière dont cette logique s'inscrit dans une **architecture** modulaire et scalable, apte à gérer de gros volumes de données, à s'interfacer avec d'autres modules, et à évoluer vers des scénarios avancés.

A. Chapitre 5 : Architecture Logicielle d'un SCN

Le chapitre suivant s'attachera à montrer **comment** transformer la simple "boucle de mise à jour" $\omega_{i,j}(t + 1)$ et ses règles (inhibition, recuit, etc.) en un **environnement logiciel** :

Organisation en modules. Le cœur du système (mise à jour de ω , calcul de synergie **S**, stratégies d'inhibition ou de sparsification) sera découpé en **briques** clairement définies, chacune remplissant un rôle précis.

Persistante de la matrice ω . Il s'agira notamment de gérer le stockage (sur disque, en RAM distribuée, ou sous format sparse) pour des matrices éventuellement très volumineuses, ainsi que de prévoir la reprise du SCN là où il s'était arrêté (enregistrant $\omega(t)$, les paramètres, etc.).

API et intégration. On décrira les interfaces (fonctions, classes, endpoints) permettant de brancher un SCN dans un framework existant (multi-agent, robotique, deep learning, etc.), de recevoir les flux de données exogènes (évolution de la synergie **S**), ou d'exporter les clusters émergents vers d'autres composants.

Cette approche **mature** du SCN assure la **maintenance** du code, la possibilité de tester facilement de nouveaux modules (par exemple un module d'inhibition différent), et une **adaptation** aisée à des cas pratiques (robotique, streaming de données, etc.).

B. Chapitres 6, 7, et 8 : Extensions et Scénarios Avancés

La base théorique et algorithmique mise en place dans le chapitre 4 (dynamique, clusters, multi-stabilité) et concrétisée architecturalement dans le chapitre 5 trouvera des prolongements au sein des chapitres 6, 7 et 8 :

Chapitre 6 : on traitera de l'**apprentissage multi-échelle**, ou de la structure **hiérarchique** d'un SCN, où l'on combine différents niveaux de granularité (micro-clusters, macro-clusters) et où la **fractalité** ou l'organisation emboîtée jouent un rôle.

Chapitre 7 : on explorera les **algorithmes d'optimisation** plus complexes (au-delà de la simple règle additive) et les techniques d'**adaptation dynamique**, qu'il s'agisse d'ajuster automatiquement η ou τ au fil du temps, ou de concevoir des approches globales (recuit multi-nœuds, stratégies d'exploration).

Chapitre 8 : on se tournera vers la **fusion multimodale**, c'est-à-dire des SCN manipulant des entités décrites par plusieurs modalités (texte, image, son, capteurs divers). Cette extension mettra en évidence la **flexibilité** du DSL pour unifier différents canaux de synergie au sein d'une même architecture.

Conclusion

Les **idées clés** de ce chapitre 4 (règle de mise à jour, inhibition, stabilisation, clusters, parallélisation) ne demeurent pas à l'état de simples algorithmes théoriques. Elles se prolongent immédiatement en :

Chapitre 5, qui décrit **comment** disposer les **modules** (mise à jour ω , calcul **S**, persistance, APIs) dans une **infrastructure modulaire et extensible**,

Chapitres 6, 7, 8, qui approfondissent des **dimensions** plus avancées : multi-échelle, optimisation et adaptation dynamique, ou fusion multimodale.

Ainsi, la **dynamique** d'un SCN (avec ses phénomènes de **clusters** émergents et ses mécanismes de contrôle) s'enrichit dans les chapitres ultérieurs d'une **architecture** logicielle robuste et de **scénarios** pratiques plus complexes, faisant de l'auto-organisation synergique un outil **générique** et **puissant** pour l'**apprentissage** et la **coordination** distribuée.

4.9.3. Place du Chapitre 4

Ce chapitre a présenté la **dynamique** propre à un **Synergistic Connection Network** (SCN), en montrant de quelle manière la **synergie** $S(i,j)$, introduite au chapitre 2 et liée aux représentations d'entités décrites au chapitre 3, conduit à la **mise à jour** adaptative de $\omega_{i,j}(t)$. Ce quatrième volet joue ainsi un **rôle de charnière** entre les **fondements** conceptuels (chapitres 2 et 3) et les **perspectives** d'implémentation et d'extension (chapitres 5 à 8).

A. Passage des Concepts à la Dynamique Opérationnelle

Les deux premiers chapitres ont posé les **fondations** d'un SCN : la notion d'entités, de synergie $S(i,j)$, et de co-occurrences ou de similarités logiques ou sub-symboliques. Sans la **dynamique**

décrise ici, ces idées se limiteraient à une théorie ou une cartographie d'affinités statiques. Le chapitre 4 a explicité comment, itération après itération, les pondérations $\omega_{i,j}$ s'**auto-adaptent** en fonction de $\omega_{i,j}(t)$ et de $S(i,j)$. Les notions de **clusters** émergents, de **multi-stabilité**, de **rôle** des paramètres η, τ, γ et de **convergence** (ou oscillations) illustrent la mise en mouvement concrète des entités, faisant passer le SCN d'une théorie de la synergie à un **réseau vivant**.

Cette transition se manifeste par la **formule** de mise à jour de base

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

éventuellement agrémentée d'un terme d'**inhibition** compétitive, d'un **recuit**, ou d'autres variantes non linéaires. Ces mécanismes répondent directement à la volonté de concrétiser la synergie en des liaisons qui se renforcent ou s'affaiblissent au fil du temps, selon leur utilité mutuelle.

B. Mise en Relation avec le Chapitre 5

Le chapitre 5 abordera la **conception** d'une **architecture** logicielle modulaire, destinée à rendre cette dynamique exploitable à grande échelle ou dans des systèmes complexes. Les principes détaillés ici — tels que l'inhibition locale ou globale, les règles additives ou multiplicatives, les stratégies de **sparcification**, la détection de clusters (section 4.6), la parallélisation ou l'asynchronisme (section 4.8) — s'intègrent dans un “**engine**” de mise à jour, que le chapitre 5 décrira selon des modules de calcul de **S**, de mise à jour ω , de persistance ou de connexion avec des frameworks externes.

L'**insertion** de ces éléments dans une architecture modulaire permet de **séparer** clairement la fonction “calculer la synergie” (chap. 3) de la fonction “faire évoluer la matrice ω ” (chap. 4). Le chapitre 5 montrera que cette discipline, impliquant parfois un double-buffer ou un mécanisme synchrone, évite bien des conflits ou pertes de cohérence. C'est grâce à de telles dispositions qu'il devient possible de **maintenir** ou de **faire évoluer** un SCN au fil du temps, éventuellement dans un contexte distribué ou multimodal.

C. Reliance aux Chapitres Avancés (6, 7, 8)

Les phénomènes d'**auto-organisation** décrits en ce chapitre sont aussi la **base** sur laquelle s'appuient les travaux ultérieurs :

- Le **multi-niveau** ou **multi-échelle** (chapitre 6), où l'on construit un SCN à plusieurs paliers hiérarchiques ;
- Les **algorithmes** d'optimisation et d'adaptation (chapitre 7), poussant plus loin la gestion de la multi-stabilité ou l'autoconfiguration de η, τ ;
- La **fusion multimodale** (chapitre 8), montrant que la mise à jour $\omega_{i,j}$ peut agréger simultanément plusieurs types de synergie (similitudes de vecteurs visuels, textes, sons), prolongeant la simple règle additive de base vers des combinaisons plus riches.

Ces développements avancés réutilisent la **dynamique** présentée dans le chapitre 4, parfois en l'agrémentant de mécanismes supplémentaires ou de variantes de la formule de mise à jour.

Conclusion

Le **chapitre 4** assure la **charnière** essentielle entre les **concept**s (chapitres 2 et 3) et la **réalisation** logicielle (chapitre 5) ainsi que les **extensions** (chapitres 6 à 8). Il a permis de voir comment la **synergie** se transforme, via des équations locales de mise à jour, en un **processus** d'auto-organisation conduisant à la **naissance de clusters**. Les règles d'**inhibition**, de **recuit** ou de **sparcification** assurent un contrôle précis du comportement de la dynamique, permettant d'éviter les effets négatifs (divergence, oscillations incontrôlées) et de stabiliser le système. Les résultats en attestent : de petits **clusters** émergent dans des cas simples, et des communautés plus complexes apparaissent dans des situations multimodales ou multi-agent.

Le chapitre suivant explicitera de manière détaillée la **structure modulaire** d'un SCN, ouvrant la voie à la mise en œuvre d'un environnement professionnel, évolutif et apte à intégrer des flux de données massifs ou permanents. Cette transition du **concret** algorithmique (chapitre 4) à l'**ingénierie** (chapitre 5) parachève la logique du **Deep Synergy Learning**, associant auto-organisation théorique et déploiement logiciel robuste.

Chapitre 5 : Le Synergistic Connection Network (SCN) : Architecture Générale

Chapitre 5 : Le Synergistic Connection Network (SCN) : Architecture Générale	648
5.1. Introduction et Enjeux.....	650
5.1.1. Contexte du Chapitre	650
5.1.2. Objectifs Principaux.....	652
5.2. Vision d'Ensemble de l'Architecture SCN.....	657
5.2. Vision d'Ensemble de l'Architecture SCN.....	658
5.2.1. Notion de “Noyau” vs. “Modules Adjoints”	658
5.2.1.2. Modules	661
5.2.2. Cycle de Vie du SCN	666
5.2.3. Exemples d'Organisation	671
5.3. Structures de Données pour la Matrice ($\backslash\omega$)	683
5.3. Structures de Données pour la Matrice ω.....	684
5.3.1. Dense vs. Sparse	684
5.3.2. Indexation et Accès Rapide	686
5.3.3. Synchronisation en Cas de Distribution	690
5.4.1. Séparation du Calcul de Synergie	693
5.4.2. Méthodes d'Implémentation	695
5.4.3. Gestion du Coût	701
5.5. Module de Mise à Jour ω et Inhibition/Contrôle.....	705
5.5.1. Rappels des Règles DSL.....	705
5.5.3. Saturation	720
5.5.4. Recuit Simulé ou Bruit	726
5.6. Interfaces Entrée-Sortie et Gestion en Temps Réel.....	739
5.6. Interfaces Entrée-Sortie et Gestion en Temps Réel.....	740
5.6.1. Ajouter / Retirer une Entité	740
5.6.2. Mise à Jour Périodique vs. Événementielle	742
5.6.3. Extraction de Clusters / Sous-Réseaux	745
5.7. Distribution, Scalabilité et Architecture Modulaire.....	751
5.7. Distribution, Scalabilité et Architecture Modulaire.....	752

5.7.1. SCN Distribué sur Plusieurs Nœuds	752
5.7.2. Mise en Place d'un “meta-SCN”.....	758
5.7.3. Ingénierie Logicielle	764
5.8. Sécurité, Fiabilité et Vérifications.....	773
5.8. Sécurité, Fiabilité et Vérifications.....	774
5.8.1. Vulnérabilités au Bruit ou à l'Attaque	774
5.8.2. Robustesse Face aux Pannes.....	779
5.8.3. Contrôle d'Intégrité	783
5.9. Exemples d'Implémentations et Études de Cas	789
5.9. Exemples d'Implémentations et Études de Cas	790
5.9.1. SCN Multimodal	790
5.10. Conclusion et Ouverture	822
5.10. Conclusion et Ouverture	823
5.10.1. Bilan du Chapitre	823
5.10.2. Lien vers Chapitres 6, 7, 8	824
5.10.3. Conclusion Générale.....	825

5.1. Introduction et Enjeux

5.1.1. Contexte du Chapitre

5.1.1.1. Rappeler le Parcours : Après Avoir Défini (Ch. 3) la Représentation des Entités et (Ch. 4) la Dynamique d'Auto-Organisation, on Aborde Maintenant la “Structuration” d'un SCN au Niveau “Architecture Globale”

Le Deep Synergy Learning (*DSL*) progresse selon un fil conducteur qui, jusqu'à présent, a posé les **fondations conceptuelles** (chapitres 2 et 3) et la **dynamique** de l'auto-organisation (chapitre 4). Ces acquis, s'ils permettent de comprendre *comment* un **Synergistic Connection Network (SCN)** calcule la **synergie** et met à jour ses pondérations $\omega_{i,j}$, doivent à présent être prolongés par une **réflexion** plus large sur la **structuration logicielle**. En effet, il ne suffit plus de connaître la formule de mise à jour additive ou la manière d'introduire une inhibition compétitive ; il faut également clarifier la **façon** dont on organise les divers modules et flux de données pour gérer efficacement un SCN de grande taille ou relié à d'autres systèmes.

Le **chapitre 3** a introduit la **représentation** des **entités** $\{\mathcal{E}_i\}$ et a montré comment traduire, dans un espace sub-symbolique (embeddings, vecteurs, signaux) ou symbolique (règles, ontologies), la notion de **contenu** d'une entité. Le **chapitre 4** s'est concentré sur la **dynamique** : il a détaillé la **mise à jour** $\omega_{i,j}(t+1)$ et le rôle crucial des **paramètres** η, τ ou γ (inhibition), révélant comment s'organisent peu à peu des **clusters** cohérents. Cette logique d'**auto-organisation**, essentielle à la philosophie du *DSL*, est désormais bien comprise : la **synergie** $S(i,j)$ détermine la vitesse à laquelle la pondération $\omega_{i,j}$ grandit ou diminue, conduisant à la formation de blocs d'entités fortement connectées.

Toutefois, disposer d'équations ou d'algorithmes de mise à jour ne garantit pas, à lui seul, la **maintenabilité** et la **scalabilité** du système. Lorsque l'on souhaite intégrer un SCN dans une application ou un cadre de recherche complexe (pouvant impliquer un grand nombre d'entités, des flux continus de données, ou une interconnexion avec d'autres briques logicielles), il faut régler plusieurs questions d'**architecture**. Le **chapitre 5** s'attachera donc à expliquer **comment** aller au-delà de la simple dynamique *algorithme* pour parvenir à un **environnement modulaire**, où les blocs de calcul (mise à jour ω , calcul de synergie, module d'inhibition, etc.) sont clairement découplés. Ce faisant, on pourra gérer la **persistence** des données (stockage de ω sur disque ou en mémoire distribuée), la **synchronisation** (threads multiples, double-buffer, etc.) et l'**interface** avec des frameworks externes (traitement multimodal, robotique, systèmes distribués).

Ainsi, ce **chapitre 5** étendra la démarche amorcée dans les chapitres précédents, en montrant **comment** on fait vivre un **SCN** dans un *code* réaliste. On y retrouvera le **calcul** de **S**, la **dynamique** de ω , et on soulignera la nécessité de séparer en **modules** indépendants tout ce qui relève de la synergie (entités, scoring), de la mise à jour ω (logique de l'auto-organisation), et des mécanismes transverses (inhibition, recuit, gestion des ressources). Par ailleurs, la question de l'**observabilité** (possibilité de visualiser l'évolution de la matrice ω et l'apparition de clusters) ou celle de la **concurrence** (threading, verrous) seront détaillées, puisqu'elles conditionnent la faisabilité d'un *SCN* à grande échelle.

Ce **parcours** illustre la progression naturelle du *DSL*. Après avoir étudié **quoi** calculer (chap. 3, la représentation des entités) et **comment** l'actualiser (chap. 4, la dynamique ω), on doit maintenant se pencher sur **comment** organiser *logiciellement* ces éléments, de sorte qu'ils demeurent robustes, évolutifs et faciles à interfacer. C'est précisément la finalité du **chapitre 5**, qui vient *après* la dynamique d'auto-organisation et qui va *ouvrir* sur les chapitres suivants consacrés aux **extensions** (multi-échelle, multimodalité, etc.).

5.1.1.2. Expliquer la Différence entre la “Dynamique” (Vue Algorithm/Math) et l’Architecture (Vue Ingénierie, Mise en Œuvre Concète)

La distinction fondamentale entre la dynamique, telle qu'elle est présentée au niveau mathématique et algorithmique, et l'architecture, qui représente la concrétisation logicielle et l'ingénierie d'un système, constitue un axe majeur pour la conception d'un **Synergistic Connection Network (SCN)** opérationnel. Au niveau de la **dynamique**, l'accent est

mis sur les équations de mise à jour des pondérations $\omega_{i,j}(t+1)$ et sur l'analyse théorique de leur comportement. Par exemple, nous avons vu que la règle additive classique s'exprime par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta (S(i,j) - \tau \omega_{i,j}(t)),$$

ou que, dans une variante multiplicative, la mise à jour peut être formulée comme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \exp(\eta [S(i,j) - \tau \omega_{i,j}(t)]).$$

Ces formules, ainsi que les paramètres associés (η pour le taux d'apprentissage, τ pour la décroissance, et éventuellement γ pour des mécanismes d'inhibition), définissent l'**essence** du comportement du SCN : ils précisent « quoi » calculer à chaque itération, comment la pondération évolue en fonction de la synergie $S(i,j)$ et comment les liens se renforcent ou se décroissent. Cette vue mathématique permet d'étudier la convergence, la formation de clusters, et d'identifier des phénomènes tels que les oscillations ou la multi-stabilité. Elle se traduit le plus souvent par un pseudo-code qui, dans sa forme la plus simple, itère sur les entités pour actualiser $\omega_{i,j}$ selon la formule ci-dessus, fournissant ainsi un cadre théorique précis pour la dynamique d'auto-organisation.

En revanche, la **vue architecturelle** se situe à un niveau d'implémentation concrète. Ici, il s'agit de concevoir une **infrastructure logicielle** capable d'exécuter la dynamique décrite tout en assurant la **maintenance**, l'**extensibilité** et la **scalabilité** du système dans un environnement réel. Cette infrastructure doit spécifier « comment » on organise les différents **modules** du SCN, c'est-à-dire la séparation claire entre le module de calcul de synergie, le module de mise à jour des pondérations et le module d'inhibition ou de recuit, ainsi que la manière de stocker la matrice ω dans des structures de données adaptées (par exemple, sous forme dense ou sous forme sparse). En outre, la gestion de la **parallelisation** et de la **synchronisation** entre les threads ou processus qui effectuent la mise à jour de ω devient cruciale quand le nombre d'entités n est élevé, car la complexité temporelle de la mise à jour est de l'ordre de $\mathcal{O}(n^2)$.

L'architecture se doit également d'offrir une interface robuste pour interagir avec d'autres modules du système, par exemple pour recevoir les modifications de la synergie $S(i,j)$ lorsqu'elle évolue au fil du temps (par exemple, en cas de mise à jour des embeddings ou de modifications des règles logiques), ou pour transmettre la matrice ω à des modules d'extraction de clusters ou d'optimisation. Des choix techniques tels que le double-buffering (pour séparer lecture et écriture des données), la gestion asynchrone des mises à jour et l'utilisation de verrous (ou de mécanismes lock-free) pour garantir la cohérence globale sont autant de considérations qui relèvent de la vue architecturelle.

Ainsi, la **dynamique** représente le “**quoi**” : elle détermine, via des formules mathématiques précises, comment les pondérations $\omega_{i,j}$ doivent évoluer en fonction des synergies et des paramètres de stabilisation. C'est une approche purement algorithmique, qui se focalise sur l'analyse théorique, les conditions de convergence et les comportements attendus (par exemple, la convergence vers $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$ dans un cas stationnaire). La **vue architecturelle**, quant à elle, répond au “**comment**” : elle concerne la structuration du code, la gestion des ressources, la modularisation, la persistance des données et l'intégration du SCN dans un système d'envergure. Cette approche permet de passer de la théorie abstraite à une application concrète, dans laquelle la dynamique mathématique est encapsulée dans un environnement logiciel modulaire, scalable et interopérable.

En résumé, la différence réside dans le fait que la **dynamique** se concentre sur la formulation et l'analyse des règles de mise à jour (par exemple, la formule additive ou multiplicative, les paramètres η , τ , et l'étude de la convergence ou des oscillations), tandis que l'**architecture** s'intéresse à la manière d'implémenter ces règles de façon efficace et fiable dans un système logiciel réel. Cette dernière inclut la séparation des responsabilités en modules indépendants, la gestion de la mémoire (stockage de la matrice ω), la parallelisation des calculs, et l'interface avec des systèmes externes. Le Chapitre 5 développera en profondeur ces aspects d'ingénierie pour fournir un “contenant” logiciel robuste qui encapsule toute la dynamique théorique exposée au Chapitre 4. Cela permet non seulement de garantir la stabilité et la performance du SCN, mais aussi d'assurer sa maintenabilité et son adaptabilité dans des environnements complexes et évolutifs.

5.1.2. Objectifs Principaux

Au sein de ce **Chapitre 5**, nous ne restons plus seulement au niveau des **équations** ou des **concepts** (décrits en chapitres 2, 3, 4), mais nous nous focalisons sur la **mise en œuvre** concrète d'un **SCN** (Synergistic Connection Network). Nous allons préciser comment **concevoir** et **organiser** l'ensemble des modules et des données nécessaires à son fonctionnement, en gardant toujours à l'esprit des objectifs tels que la **clarté**, la **robustesse**, la **scalabilité** et **l'extensibilité**. Les grandes thématiques couvertes dans cette section (5.1.2) incluent :

5.1.2.1. Discuter Comment Organiser le SCN en Modules (Calcul de Synergie, Mise à Jour ω , etc.)

La conception d'un **Synergistic Connection Network (SCN)** opérationnel ne se limite pas à l'implémentation de la dynamique mathématique de mise à jour des pondérations $\omega_{i,j}(t+1)$. Pour que le système puisse être déployé à grande échelle, intégré dans des applications réelles et maintenu dans le temps, il est impératif de l'organiser en **modules distincts**. Cette approche modulaire, qui relève du principe de « separation of concerns », permet de dissocier la logique algorithme/mathématique – qui décrit « ce qu'il faut calculer » – de l'architecture logicielle – qui répond à la question « comment le code est structuré et géré dans un environnement d'exécution complexe ». Nous détaillerons ci-après les différents modules ainsi que les avantages qu'apporte cette organisation.

A. Raison d'être de la Modularisation

La dynamique d'un SCN repose sur des formules telles que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left(S(i,j) - \tau \omega_{i,j}(t) \right)$$

ou sa variante multiplicative, qui définissent l'évolution de la matrice ω en fonction de la **synergie** $S(i,j)$ entre les entités \mathcal{E}_i et \mathcal{E}_j . Ces équations, ainsi que leurs paramètres η , τ et éventuellement γ (pour l'inhibition), décrivent le comportement dynamique du système. Cependant, pour rendre ce comportement exploitable en pratique, il faut structurer le code de manière à isoler les différentes responsabilités. La modularisation permet notamment de séparer le **calcul de synergie** – qui peut reposer sur des données hétérogènes telles que des embeddings ou des règles logiques – de la **mise à jour** effective des pondérations ω . Cette séparation offre une meilleure lisibilité du code, facilite le débogage et permet de remplacer ou d'améliorer chaque composant sans avoir à repenser l'ensemble du système. En outre, elle rend possible une configuration dynamique des paramètres et une gestion efficace des ressources, deux aspects cruciaux quand le nombre d'entités n devient important.

B. Composition des Modules et Communication

Dans une implémentation modulaire d'un SCN, plusieurs blocs fonctionnels sont identifiés, chacun jouant un rôle spécifique :

- **Module « Calcul de Synergie »** : Ce module est responsable de la détermination de la fonction $S(i,j)$ qui quantifie la proximité ou la complémentarité entre les entités \mathcal{E}_i et \mathcal{E}_j . Pour des données sub-symboliques, il peut s'agir d'une similarité vectorielle (par exemple, une similarité cosinus ou une fonction de noyau gaussien) et, dans un contexte hybride, ce module peut combiner des scores issus d'embeddings et des scores de compatibilité logique. La fonction peut être exprimée, par exemple, sous la forme

$$S_{\text{hybrid}}(i,j) = \alpha S_{\text{sub}}(\mathbf{x}_i, \mathbf{x}_j) + (1 - \alpha) S_{\text{sym}}(R_i, R_j),$$

où α ajuste l'importance relative des composantes sub-symbolique et symbolique. Ce module reçoit en entrée les données des entités (embeddings, règles, etc.) et renvoie un score numérique exploitable par les autres modules.

- **Module « Mise à Jour de ω »** : Ce bloc implémente la dynamique propre aux pondérations. Il applique la règle de mise à jour, qu'elle soit additive ou multiplicative, et intègre les paramètres de contrôle tels que le taux d'apprentissage η et le facteur de décroissance τ . Par exemple, en mode additif, la mise à jour est donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left(S(i,j) - \tau \omega_{i,j}(t) \right).$$

L'objectif de ce module est de transformer, à chaque itération, la matrice $\omega(t)$ en une nouvelle configuration $\omega(t+1)$, en respectant les règles de la dynamique auto-organisée. En encapsulant cette fonction dans un module dédié, on peut facilement expérimenter différentes variantes (par exemple, intégrer des mécanismes de recuit ou de clipping) sans affecter le calcul de $S(i,j)$.

- **Module « Inhibition » :** Afin d'éviter une croissance excessive ou non sélective des pondérations, ce module applique des politiques d'inhibition, telles que l'inhibition compétitive locale ou globale. La règle peut prendre la forme d'un terme additionnel, par exemple

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

qui pénalise l'activation simultanée de nombreux liens par une même entité. La modularisation de cette fonction permet de l'activer ou de la désactiver aisément, voire de la remplacer par une stratégie de sparsification, en fonction des exigences de l'application.

- **Module « Observer et Extraction de Clusters » :** Ce module a pour rôle de surveiller la matrice ω au fil des itérations, d'extraire des indicateurs de performance (tels que la modularité ou l'indice de silhouette) et d'identifier les clusters émergents. Il peut s'agir d'un outil de visualisation en temps réel, similaire à TensorBoard, qui permet de suivre l'évolution de la dynamique et d'ajuster les paramètres en conséquence.

La communication entre ces modules s'effectue via des interfaces bien définies : le module de mise à jour reçoit le score $S(i,j)$ calculé par le module de synergie, et transmet la nouvelle matrice ω aux modules d'inhibition et d'observation. Cette séparation garantit que l'évolution d'un module n'impacte pas directement les autres, permettant ainsi une maintenance aisée et une évolutivité du système.

C. Flexibilité et Extensibilité de l'Architecture

L'un des grands avantages d'une architecture modulaire est sa **flexibilité**. En isolant chaque composant, il est possible de modifier ou de remplacer une partie du système sans réécrire l'ensemble du code. Par exemple, si l'on souhaite passer d'une formule additive à une mise à jour multiplicative, il suffit d'ajuster le module de mise à jour de ω tout en conservant intact le module de calcul de synergie. De même, l'activation d'un module de recuit stochastique ou l'introduction de techniques de sparsification peut se faire par simple configuration, via des plugins ou des paramètres dans le gestionnaire de la dynamique. Cette approche modulaire rend également le système **scalable** : dans le cas où le nombre d'entités n devient très grand, la matrice ω peut être gérée à l'aide de structures de données adaptées (par exemple, des matrices creuses ou des bases de données distribuées), et le traitement peut être parallélisé de manière efficace grâce à un découpage des tâches entre plusieurs threads ou machines. Enfin, l'interopérabilité est facilitée puisque chaque module communique via des API standardisées, ce qui permet d'intégrer le SCN dans des environnements hétérogènes (systèmes multimodaux, simulateurs de robotique, etc.).

D. Exemple d'Organisation Modulaire

Pour illustrer, considérons un exemple hypothétique où le SCN est organisé en quatre modules principaux, orchestrés par un moteur central :

- **SCNCore** : Ce composant central détient la matrice ω et gère le cycle d'itération global. Il initie la boucle de mise à jour et coordonne les appels aux différents modules.
- **SynergyModule** : Responsable du calcul de $S(i,j)$, il reçoit les données d'entrée (embeddings, règles) et produit le score de synergie. Ce module peut être configuré pour utiliser différentes fonctions de similarité selon le contexte.
- **UpdateModule** : Ce module applique la règle de mise à jour de ω . Par exemple, il peut utiliser la formule additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left(S(i,j) - \tau \omega_{i,j}(t) \right),$$

ou une variante multiplicative, selon les paramètres fournis. Il reçoit le score $S(i,j)$ du SynergyModule et intègre les valeurs η et τ pour produire la nouvelle matrice.

- **InhibitionModule** : En tant que composant optionnel, il s'occupe d'appliquer des stratégies d'inhibition ou de sparsification, par exemple en limitant la somme des pondérations par ligne ou en imposant un clipping sur les valeurs extrêmes.
- **ObserverModule** : Ce module effectue des analyses de la matrice ω (par exemple, calcul de la modularité, détection de clusters) et fournit des visualisations ou des rapports pour surveiller la dynamique du SCN.

Chaque module communique avec SCNCORE via des interfaces bien définies. Par exemple, le SynergyModule peut être appelé avec une fonction « computeSynergy($E[i]$, $E[j]$) » qui retourne $S(i,j)$, et le UpdateModule peut disposer d'une fonction « updateOmega(ω , S , η , τ) » qui produit $\omega(t+1)$. Ce découpage facilite la maintenance et permet de tester indépendamment chaque composant.

E. Conclusion

L'organisation modulaire d'un SCN, telle que décrite dans ce point, permet de dissocier clairement la **dynamique mathématique** – qui s'exprime par des règles de mise à jour de ω en fonction de $S(i,j)$ – de l'**architecture logicielle** qui matérialise ces calculs dans un environnement réel. En isolant les modules de **calcul de synergie**, de **mise à jour de ω** et d'**inhibition**, ainsi que les outils d'**observation** et de **persistence**, on obtient un système **flexible, scalable et interopérable**. Cette structure permet non seulement de modifier aisément des aspects internes du SCN (comme le passage d'une règle additive à une règle multiplicative), mais aussi d'intégrer le SCN dans des environnements complexes, de gérer de grands volumes de données et de garantir une maintenance efficace. Le chapitre 5 approfondira ces concepts en décrivant l'architecture globale, les interfaces, et les stratégies de parallélisation et de persistance, assurant ainsi que la dynamique théorique du SCN se traduise par une infrastructure logicielle robuste et adaptable à divers cas d'usage.

5.1.2.2. Aborder la Persistance des Données, la Scalabilité, la Gestion d'Entités Hétérogènes dans un Seul Framework

Dans le contexte d'un **Synergistic Connection Network** (SCN), une fois que les modules essentiels de la dynamique – notamment le **calcul de la synergie** $S(i,j)$ et la **mise à jour** des pondérations $\omega_{i,j}(t+1)$ – ont été définis, il apparaît indispensable de développer une infrastructure logicielle robuste qui intègre des mécanismes de **persistence**, assure la **scalabilité** pour un grand nombre d'entités et permette la gestion homogène d'entités hétérogènes dans un cadre unique. Cette section présente une approche intégrée, en insistant sur la formalisation mathématique des problèmes, sur les choix de structures de données et sur l'architecture logicielle qui facilitent l'implémentation opérationnelle d'un SCN.

A. Persistence des Données

La **persistence** des données constitue un enjeu majeur lorsque la matrice des pondérations, notée $\omega \in \mathbb{R}^{n \times n}$, peut devenir volumineuse, en particulier lorsque le nombre d'entités n croît de manière significative, entraînant une complexité mémoire de l'ordre de $\mathcal{O}(n^2)$. Dans un tel scénario, il est crucial d'enregistrer l'état du SCN à intervalles réguliers afin de permettre la reprise d'une simulation interrompue ou d'effectuer une analyse *a posteriori* de la dynamique. Par exemple, on peut définir une fonction de sauvegarde

$$\text{SaveState} : \omega(t) \rightarrow \text{Fichier (ou base de données)},$$

qui stocke l'état courant dans un format binaire ou au format HDF5. Cette approche permet non seulement de garantir la continuité des simulations en cas d'interruption, mais aussi d'enregistrer l'évolution de la dynamique pour une analyse ultérieure. En outre, lorsque la matrice ω est essentiellement creuse – c'est-à-dire que la majorité des liens $\omega_{i,j}$ restent faibles ou nuls après l'application de techniques de **sparsification** – l'utilisation de structures de données spécialisées telles que les matrices **sparse** contribue à optimiser la persistance et la rapidité d'accès, tout en réduisant l'empreinte mémoire.

B. Scalabilité

La **scalabilité** se réfère à la capacité du SCN à gérer efficacement un grand nombre d'entités, ce qui pose deux défis majeurs : le coût du stockage de la matrice ω et le temps de calcul associé à la mise à jour, qui implique potentiellement de traiter $\mathcal{O}(n^2)$ paires à chaque itération. Pour pallier ces difficultés, il est judicieux de recourir à des méthodes de **sparcification** qui consistent à filtrer les liens insignifiants. Par exemple, en définissant un opérateur de filtre Π tel que

$$\tilde{\omega}_{i,j} = \Pi(\omega_{i,j}) = \begin{cases} \omega_{i,j}, & \text{si } \omega_{i,j} \geq \theta, \\ 0, & \text{sinon,} \end{cases}$$

on peut ainsi réduire le nombre total de liens stockés à $\mathcal{O}(nk)$, avec $k \ll n$. Par ailleurs, la parallélisation du calcul, en utilisant des techniques telles que le double-buffering et la distribution des tâches sur plusieurs threads ou machines (via MPI ou OpenMP), permet de diviser le travail de mise à jour de manière efficace. La stratégie consiste à partitionner la matrice ω en sous-blocs, de sorte que chaque thread traite indépendamment un ensemble de lignes ou de colonnes, réduisant ainsi considérablement le temps de calcul par itération.

C. Gestion d'Entités Hétérogènes dans un Seul Framework

Dans de nombreux cas d'usage, le SCN doit intégrer des entités de nature diverse – par exemple, des **entités sub-symboliques** issues d'embeddings neuronaux, des **entités symboliques** représentées par des ensembles de règles ou d'axiomes, et même des agents physiques comme des robots. Pour assurer une **intégration homogène**, il est essentiel de définir une **interface** abstraite, par exemple une classe **Entity**, qui spécifie les méthodes communes telles que `getRepresentation()` ou `computeSynergy(other)`. Chaque type d'entité (sub-symbolique, symbolique ou hybride) implémente cette interface en fournissant ses propres méthodes de calcul. Par exemple, dans un scénario où la fonction de synergie est définie par

$$S(i, j) = \alpha S_{\text{sub}}(i, j) + (1 - \alpha) S_{\text{sym}}(i, j),$$

les entités sub-symboliques et symboliques peuvent ainsi être traitées uniformément par le module de **Calcul de Synergie**, qui s'appuie sur les méthodes polymorphiques définies dans la classe **Entity**. Cette approche permet non seulement de faciliter l'extension du SCN à de nouveaux types d'entités, mais également de standardiser la manière dont les différentes représentations interagissent au sein du même réseau.

D. Interfaçage et API

Afin de garantir une intégration efficace du SCN dans un environnement logiciel plus vaste, il est impératif de développer des **API** claires et bien définies. Ces API doivent permettre l'accès aux données du SCN (la matrice ω et la fonction de synergie $S(i, j)$), ainsi que la modification dynamique des paramètres essentiels tels que η , τ et γ . Par exemple, une interface du type

`getOmega()` et `setParameters(η, τ, γ)`

permet de communiquer avec d'autres modules ou systèmes, tels que des outils de visualisation, des systèmes de contrôle en temps réel ou des simulateurs de tâches complexes. Cette interopérabilité est cruciale pour que le SCN puisse fonctionner en tant que composant intégré dans des frameworks plus larges, notamment dans des applications de robotique ou d'apprentissage multimodal.

Conclusion

L'**architecture** d'un SCN opérationnel doit être conçue pour répondre simultanément aux exigences de persistance, de scalabilité et d'intégration d'entités hétérogènes. En mettant en œuvre un **module de persistance** robuste, le système peut sauvegarder et restaurer efficacement l'état de la matrice ω , ce qui est indispensable pour la continuité des simulations et l'analyse a posteriori. La scalabilité est assurée grâce à des techniques de **sparcification**, à une parallélisation judicieuse et à l'utilisation de structures de données adaptées, ce qui permet de gérer des réseaux de grande dimension tout en maintenant un temps de calcul raisonnable. Enfin, la gestion d'entités hétérogènes repose sur une abstraction orientée objet et sur des API standardisées qui garantissent une interopérabilité optimale entre les différents types d'entités et les modules de calcul. Ces stratégies, intégrées dans un cadre modulaire et extensible,

posent les bases d'un SCN capable de s'insérer dans des environnements complexes et évolutifs, tel que détaillé au Chapitre 5.

5.1.2.3. Préparer les Chapitres Suivants (Multi-Échelle, Optimisations Avancées, Etc.)

Dans les sections précédentes, il a été montré comment une architecture modulaire favorise la clarté et la flexibilité d'un **Synergistic Connection Network** (SCN). Les deux premiers objectifs, à savoir la séparation des rôles (calcul de synergie, mise à jour ω , inhibition, etc.) et la prise en compte de la persistance, de la scalabilité et de l'hétérogénéité des entités, répondent déjà à la plupart des exigences d'une mise en œuvre robuste. Néanmoins, cet agencement ne représente que la base d'un cadre plus large. Les chapitres suivants (6, 7, 8) détailleront des domaines où l'architecture doit être étendue ou adaptée à des situations plus sophistiquées : il peut s'agir de gérer plusieurs niveaux d'échelles, de recourir à des algorithmes d'optimisation plus élaborés, ou d'intégrer des canaux multimodaux variés.

Le chapitre 6 abordera la question de la **multi-échelle**, c'est-à-dire l'articulation du SCN selon des niveaux hiérarchiques, avec la possibilité que la synergie et la mise à jour ω se déclinent sous différentes granularités. On peut envisager des entités formant des micro-clusters locaux, enchevêtrés dans de plus grands macro-clusters destinés à un aperçu plus global. Cette hiérarchie demande souvent de disposer de structures de données ou de métadonnées permettant de résumer les sous-réseaux à des niveaux supérieurs, et de prévoir des règles distinctes à chaque palier. L'architecture modulaire telle qu'elle se dessine au chapitre 5 devra autoriser l'insertion de "meta-niveaux" ou "cluster-layers" dans la boucle de mise à jour. Les modules définissant la synergie ou la mise à jour ω se voient ainsi complétés par un gestionnaire multi-niveau supervisant l'échange d'informations entre échelles.

Le chapitre 7 se consacrera aux **optimisations avancées** et à l'**adaptation dynamique** des paramètres. L'idée d'adapter automatiquement η , τ ou γ en cours d'exécution, en fonction d'observateurs mesurant la vitesse de convergence ou le taux d'oscillation, nécessitera un composant "feedback" supplémentaire. Des algorithmes inspirés des approches globales (tels que le recuit simulé plus élaboré, ou un gradient global sur la matrice ω) pourront aussi être intégrés au SCN. Une architecture modulaire rend bien plus simple l'activation de routines spécialisées (par exemple, un "ParamManager" réajustant η selon le rythme d'évolution des liens ω), tout en préservant la mise à jour locale. Il pourra également être question d'une commande "top-down" ou d'un feedback direct injectant un contrôle externe sur certaines parties du réseau, pour dépasser la simple auto-organisation "bottom-up".

Enfin, le chapitre 8 portera sur la **fusion multimodale**. Il s'agira de démontrer la souplesse du **Deep Synergy Learning** (DSL) dans des domaines complexes (vision, langage, audio, etc.) où les entités représentent des canaux ou des descripteurs hétérogènes. On appréciera alors l'importance cruciale d'une architecture apte à recevoir, pour chaque paire (i, j) , plusieurs sources de synergie $\mathbf{S}_1(i, j), \mathbf{S}_2(i, j) \dots$, correspondant à différents canaux (visuel, textuel, sonore). Ce cadre multimodal repose sur la cohabitation naturelle de multiples "entités" dans le SCN, ce que le chapitre 5 aura préparé, en s'assurant que la gestion polymorphe des entités et la persistance à grande échelle puissent se décliner de façon transparente, quel que soit le mode de calcul du score $S(i, j)$.

En somme, l'architecture décrite au chapitre 5 constituera le socle permettant de développer chacune de ces expansions. Il sera beaucoup plus aisés d'ajouter, par exemple, un module "multi-scale synergy" pilotant un niveau hiérarchique supplémentaire, ou un module "fusion visuo-textuelle" manipulant diverses modalités, lorsqu'un découpage modulaire (mise à jour ω , calcul de synergie, etc.) est déjà en place. Une mise en cohérence entre les chapitres 6, 7, et 8 permettra de montrer à quel point le **SCN**, initialement concentré sur l'auto-organisation locale des liens, peut s'adapter à des configurations hiérarchiques, des stratégies de contrôle top-down, ou des données multimodales abondantes. L'ensemble fera apparaître le **Deep Synergy Learning** comme un paradigme général et extensible, autorisant nombre d'expérimentations et d'applications pratiques, dès lors qu'on se dote d'une architecture logicielle rigoureuse et ouverte.

5.2. Vision d'Ensemble de l'Architecture SCN

5.2.1. Notion de “Noyau” vs. “Modules Adjoints”

- 5.2.1.1. **Noyau (Core)** : stocke la matrice $\{\omega_{i,j}\}$, exécute le cycle de mise à jour selon la formule du DSL.
- 5.2.1.2. **Modules** :
 - Module Synergie (calcule $S(i,j)$),
 - Module Inhibition/Contrôle,
 - Module d'Interface (ajout/suppression d'entités), etc.

5.2.2. Cycle de Vie du SCN

- 5.2.2.1. **Initialisation** : chargement des entités, création (ou non) d'une matrice ω initiale.
- 5.2.2.2. **Boucle d'itérations** : calcul de synergie, mise à jour ω , éventuelle inhibition ou saturation, extraction de clusters.
- 5.2.2.3. **Sortie / Flux continu** : quand stabilisé, ou en mode perpétuel si apprentissage continu.

5.2.3. Exemples d'Organisation

- 5.2.3.1. **Mono-module** minimal (tout dans un même bloc)
- 5.2.3.2. **Architecture modulaire** (chacun s'occupe d'une fonction)
- 5.2.3.3. **Architecture distribuée** (plusieurs sous-SCN coopérant)

5.2. Vision d'Ensemble de l'Architecture SCN

5.2.1. Notion de “Noyau” vs. “Modules Adjoints”

5.2.1.1. Noyau (Core) : Stocke la Matrice $\{\omega_{i,j}\}$ et Exécute le Cycle de Mise à Jour selon la Formule du DSL

Le **Noyau** représente la pièce maîtresse de l'architecture du *Synergistic Connection Network* (SCN) dans le cadre du **Deep Synergy Learning** (DSL). Il constitue la **colonne vertébrale** du système en assurant à la fois le **stockage** central de la matrice des pondérations $\Omega(t) \in \mathbb{R}^{n \times n}$ et le **pilotage** de la dynamique d'auto-organisation par la mise à jour itérative des poids $\omega_{i,j}(t)$. Cette section présente en détail le rôle du Noyau, sa formalisation mathématique, la gestion des aspects de stockage et de complexité, ainsi que son interaction avec d'autres modules du système.

A. Rôle Fondamental du Noyau

Le **Noyau** se définit avant tout comme le gardien de la matrice $\Omega(t)$, qui regroupe l'ensemble des pondérations $\omega_{i,j}(t)$ établies entre toutes les entités \mathcal{E}_i du réseau. En pratique, il doit garantir un **stockage efficace** de ces valeurs, que ce soit sous un format **dense** ou **sparse** afin de minimiser l'empreinte mémoire, notamment lorsque le nombre d'entités n devient très grand (par exemple, $n = 10^4$ à 10^5). Sur le plan **mathématique**, la matrice $\Omega(t)$ évolue selon une fonction de transition que nous pouvons écrire de manière générale :

$$\omega(t+1) = F(\omega(t), \mathbf{S}, \{\text{inhibition, bruit, etc.}\}),$$

où \mathbf{S} représente la matrice des **synergies** $S(i,j)$ entre les entités, et les autres termes (inhibition, bruit, etc.) viennent moduler l'évolution du système.

Sur le plan **algorithmique**, le Noyau agit comme une brique centrale qui orchestre l'ensemble du **cyclique itératif** de mise à jour. Ce cycle repose sur la formule de mise à jour propre au DSL. Par exemple, dans le cas d'une mise à jour de type **additif**, nous avons :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t) + \dots,$$

où η représente le **taux d'apprentissage**, τ le **facteur de décroissance** qui module le coût de maintenir un lien élevé, et γ un coefficient d'**inhibition** appliqué sur les liens concurrents. Cette formule exprime l'essence de la **descente d'énergie** qui guide l'auto-organisation du réseau.

B. Stockage de la Matrice Ω et Gestion de la Complexité

Le Noyau doit assurer le **stockage** et la gestion de la matrice $\Omega(t)$, qui contient $O(n^2)$ éléments pour n entités. Lorsque n est élevé, le coût mémoire d'un stockage dense peut rapidement devenir prohibitif (pouvant atteindre des dizaines ou centaines de gigaoctets). Pour pallier ce problème, des techniques de **sparcification** telles que l'utilisation de structures de données *sparse* ou la limitation des liaisons à un nombre restreint de voisins (k-NN ou ϵ -radius) peuvent être mises en œuvre. Ainsi, le Noyau doit intégrer des routines d'accès rapide et de mise à jour efficaces, en exploitant par exemple des stratégies de **parallélisation** ou de **double-buffer** pour gérer les itérations.

Sur le plan **algébrique**, la mise à jour s'effectue selon la relation :

$$\omega(t+1) = F(\omega(t), \mathbf{S}, \{\text{termes de régulation}\}),$$

ce qui implique un parcours complet (ou partiel en mode sparse) sur tous les indices $(i,j) \in \{1, \dots, n\}^2$.

C. Cycle Itératif de Mise à Jour

Le fonctionnement du Noyau peut être décomposé en une série d'étapes séquentielles qui assurent la mise à jour correcte de la matrice Ω . Ce **cyclique** se réalise généralement en six étapes clés :

234. **Lecture** : Le Noyau commence par lire la matrice des pondérations $\omega_{i,j}(t)$ depuis un buffer de lecture, souvent désigné par w_{current} .

235. **Acquisition de la Synergie** : Il sollicite un module externe, tel que le **Module Synergie**, afin d'obtenir les valeurs de $S(i, j)$ qui déterminent la proximité ou la complémentarité entre les entités.

236. **Calcul des Incréments** : Pour chaque paire (i, j) , le Noyau calcule l'incrément $\Delta\omega_{i,j}$ à partir de la formule de mise à jour, tenant compte des paramètres d'apprentissage, des termes d'inhibition et de toute composante stochastique.

237. **Écriture** : Les nouvelles valeurs $\omega_{i,j}(t + 1)$ sont écrites dans un buffer d'écriture, appelé w_{next} .

238. **Traitement Postérieur** : Un module optionnel d'inhibition ou de post-traitement peut être appliqué pour ajuster les valeurs calculées, par exemple en imposant des limites ou en appliquant des règles de sparsification.

239. **Basculement** : Enfin, après vérification de la cohérence de l'itération, le Noyau effectue un swap entre w_{current} et w_{next} , marquant la transition de l'itération t à $t + 1$.

Cette séquence garantit que la dynamique du DSL se réalise de manière ordonnée et cohérente, en appliquant à chaque itération la formule mathématique qui définit la descente d'énergie du système.

D. Interfaces et Modularité

Le Noyau se veut **minimaliste** dans ses responsabilités : il gère la matrice Ω et orchestre la boucle de mise à jour, tout en déléguant le calcul détaillé de $S(i, j)$ et des éventuels termes additionnels (comme l'inhibition) à des modules externes. Grâce à cette **modularité**, il est possible de modifier ou de remplacer ces modules indépendamment du cœur de la dynamique. Par exemple, il est envisageable de substituer le **Module Synergie** basé sur la distance euclidienne par un module exploitant la **co-information** ou d'autres critères, sans affecter le mécanisme central de mise à jour de ω . Cette séparation des préoccupations favorise l'**extensibilité** et la **maintenabilité** du système.

E. Conclusion sur le Noyau (Core)

En résumé, le **Noyau** représente la **colonne vertébrale** du SCN dans le cadre du DSL. C'est l'endroit où la **matrice $\Omega(t)$** est **stockée** et **gérée**, et où la **boucle itérative** de mise à jour se réalise en appliquant la formule fondamentale, par exemple :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t) + \dots$$

D'un point de vue **mathématique**, c'est dans ce Noyau que se matérialise la **descente d'énergie** et la convergence vers des points fixes ou des attracteurs du système. Du point de vue **logiciel**, le Noyau constitue une brique modulaire essentielle, qui, grâce à des interfaces claires avec les modules de calcul de synergie, d'inhibition, et éventuellement de gestion d'états internes, assure l'auto-organisation du SCN dans un environnement évolutif et scalable. Ainsi, le **Noyau (Core)** n'est pas seulement responsable de la mise à jour des pondérations, il assure également la cohérence globale du système en orchestrant la séquence d'opérations qui fait converger la dynamique vers une configuration stable, tout en permettant l'intégration et la maintenance de modules complémentaires dans un cadre logiciel robuste et évolutif.

F. Le Noyau gérant également les États, Mémoires et Agents (complement)

Dans l'architecture précédente, on a insisté sur le fait que le **Noyau** (ou *Core*) d'un **Synergistic Connection Network (SCN)** se concentrerait principalement sur le stockage de la matrice ω et l'exécution de la boucle de mise à jour du **Deep Synergy Learning (DSL)**. Toutefois, lorsqu'une **entité** \mathcal{E}_i n'est pas un simple noeud statique, mais un **agent** susceptible de disposer d'un **état** interne (mémoire, variables locales, etc.), cette information doit être gérée d'une manière cohérente avec le **Noyau**. En pratique, il est fréquent que le *Core* stocke non seulement les pondérations $\omega_{i,j}$, mais aussi tout ou partie des **états** de ces agents, afin d'assurer un **cycle** de mise à jour unifié.

Si chaque entité \mathcal{E}_i possède un vecteur d'état $\mathbf{s}_i(t)$ (par exemple, un accumulateur ou même une cellule LSTM locale), on peut étendre la structure du **Noyau** pour mémoriser ces informations. Au lieu de se contenter d'une simple matrice $\omega(t) \in \mathbb{R}^{n \times n}$, le Noyau conserve également :

$$\mathbf{S}_{\text{etats}}(t) = \{\mathbf{s}_1(t), \mathbf{s}_2(t), \dots, \mathbf{s}_n(t)\},$$

chaque $\mathbf{s}_i(t)$ pouvant être un vecteur (ou un objet plus complexe) décrivant la **mémoire** courante de l'agent \mathcal{E}_i . On dispose alors d'une **dynamique** double : la mise à jour de $\omega_{i,j}$ (pondérations inter-agents) et la mise à jour de $\mathbf{s}_i(t)$ (mémoire ou état interne), laquelle peut être notée :

$$\mathbf{s}_i(t+1) = \Phi(\mathbf{s}_i(t), \{\omega_{i,j}(t)\}, \{\mathbf{s}_j(t)\}_{j \neq i}).$$

Le **Noyau** coordonne ainsi l'évolution simultanée de ω et \mathbf{s}_i , selon les règles du DSL enrichi (une équation couplée permettant aux agents de modifier leur état selon la synergie, et aux poids d'être ajustés en fonction des états et synergies).

Lorsqu'une **entité** \mathcal{E}_i représente un **agent** autonome, capable de prendre des décisions ou de gérer un comportement local, on veut généralement lui attribuer un espace d'état $\mathbf{s}_i(t) \in \mathbb{R}^d$ (ou un ensemble de variables plus symboliques). Le **Noyau** peut alors :

240. **Stocker** ces états dans une structure annexe, par exemple un tableau $\text{Etats}[1..n]$ où $\text{Etats}[i]$ pointe vers $\mathbf{s}_i(t)$.

241. **S'assurer** que, lors de chaque itération DSL, on exécute la mise à jour de $\mathbf{s}_i(t)$. Ceci peut être fait après ou avant la mise à jour de $\omega_{i,j}(t)$. Sur le plan formel, on définit un couple :

$$(\Omega(t), \mathbf{S}_{\text{etats}}(t)) \mapsto (\Omega(t+1), \mathbf{S}_{\text{etats}}(t+1)),$$

où la fonction de transition prend en compte à la fois la dynamique DSL sur ω et la dynamique interne sur \mathbf{s}_i .

On peut imaginer une **formule** conjointe :

$$\begin{cases} \mathbf{s}_i(t+1) = \Phi_i(\mathbf{s}_i(t), \{\omega_{i,j}(t)\}, \{\mathbf{s}_j(t)\}), \\ \omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathbf{s}_i(t), \mathbf{s}_j(t)) - \tau \omega_{i,j}(t)] + \dots \end{cases}$$

La **synergie** $S(\mathbf{s}_i, \mathbf{s}_j)$ dépend alors des **états** respectifs, ce qui justifie que le **Noyau** gère ou au moins coordonne l'accès à \mathbf{s}_i . En pratique, on peut loger \mathbf{s}_i dans le **Noyau** ou dans un "Module État" qui communique fortement avec le **Noyau**. L'important est que la **boucle** de mise à jour globale traite à la fois les poids $\omega_{i,j}$ et les états \mathbf{s}_i , maintenant une cohérence entre les deux.

D'un point de vue **architecture**, si l'on ne gérait que ω , le **Noyau** se limite à la "matrice de liens" et sa dynamique. Dès que des entités \mathcal{E}_i possèdent un état interne évolutif, il devient naturel d'étendre le **Noyau** pour inclure la **mémorisation** de ces états, ou au moins l'orchestration de leur mise à jour. Selon le degré de modularité voulu, le Noyau peut :

- Soit conserver physiquement les \mathbf{s}_i
- Soit appeler un "Module État" qui, à chaque itération, calcule $\mathbf{s}_i(t+1)$.

Dans tous les cas, la **synchronicité** de la mise à jour ω et \mathbf{s}_i doit être garantie : on évite que l'agent modifie son état selon un $\omega(t)$ déjà obsolète ou qu'il mette à jour ω avant d'avoir tenu compte du nouvel état. Les algorithmes de double buffer (pour ω) peuvent s'étendre à l'état \mathbf{s}_i s'il faut traiter la mise à jour de façon synchrone.

Conclusion

Il apparaît donc naturel que le **Noyau** n'héberge pas **uniquement** la matrice ω , mais puisse **inclure** (ou gérer étroitement) les **états** des entités/agents, si l'on souhaite un **DSL** où ces entités possèdent une mémoire ou un comportement local. Les formules mathématiques, dans ce cas, se présentent sous forme de couples $(\omega, \mathbf{s}) \mapsto (\omega', \mathbf{s}')$,

assurant la cohérence de la **dynamique** auto-organisée. L'architecture globale reste la même : le **Noyau** — désormais muni à la fois de Ω et de S_{etats} — exécute le **cycle** DSL, tandis que les **Modules** adjacents (calcul de synergie, inhibition, recuit simulé, etc.) fournissent les composantes nécessaires (valeurs ou compléments de formules), tout en déléguant la mise à jour itérative au **Noyau**.

5.2.1.2. Modules

Si le **Noyau (Core)** est la pièce maîtresse qui assure la conservation et la mise à jour de la matrice ω (voir 5.2.1.1), il n'opère pas tout seul. Dans une **architecture** SCN modulaire, nous identifions plusieurs **modules** “adjoints” (ou “satellites”) qui viennent se **connecter** au Noyau pour fournir des fonctionnalités supplémentaires :

- 242. **Module Synergie** (calcul de $S(i, j)$),
- 243. **Module Inhibition/Contrôle**,
- 244. **Module d'Interface** (ajout/suppression d'entités, configuration, etc.),
- 245. (Possiblement) d'autres modules : “Recuit simulé”, “Analyse/Observateur”, “Gestion paramétrique adaptative”, etc.

Chacun de ces **modules** sert un **rôle** précis et interagit mathématiquement ou logiquement avec le **Noyau** pour enrichir le comportement du SCN. Nous détaillons ci-après les trois modules majeurs cités.

A. *Module Synergie (calcule $S(i, j)$)*

Le **Module Synergie** constitue un élément fondamental dans l'architecture d'un **Synergistic Connection Network (SCN)**, car c'est lui qui fournit la **fonction** $S(i, j)$ permettant de mesurer la “distance”, la “similarité” ou plus généralement la **coopération** entre deux entités \mathcal{E}_i et \mathcal{E}_j . Dans le **Deep Synergy Learning (DSL)**, chaque pondération $\omega_{i,j}(t)$ se met à jour en tenant explicitement compte de la valeur $S(i, j)$, ce qui fait du Module Synergie un composant déterminant pour l'évolution du réseau.

Il est fréquent de poser, dans la forme la plus simple, une mise à jour additive du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)] - \text{ (termes d'inhibition éventuels)},$$

où $S(i, j)$ désigne précisément la valeur fournie par ce **Module Synergie**. Sur le plan **mathématique**, on peut voir S comme une matrice $\mathbf{S} \in \mathbb{R}^{n \times n}$, mais sa construction ou son calcul peut être complexe, en fonction de la nature des entités et des traitements nécessaires pour en déduire la synergie.

L'objectif fondamental du Module Synergie est de **fournir**, pour chaque couple (i, j) , une quantité $S(i, j)$ qui sera ensuite exploitée par le Noyau (Core). Le Module Synergie peut être envisagé comme un “serveur” auquel le Noyau fait appel : lors de chaque itération du DSL, le Noyau “demande” la valeur $S(i, j)$ pour mettre à jour $\omega_{i,j}$. Selon la **typologie** des entités, la fonction S peut adopter diverses formules. Dans un cas sub-symbolique, on peut considérer une distance de type gaussien ou exponentiel :

$$S(i, j) = \exp(-\alpha \| \mathbf{x}_i - \mathbf{x}_j \|^2),$$

avec $\alpha > 0$, ou encore une similarité cosinus :

$$S(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\| \mathbf{x}_i \| \| \mathbf{x}_j \|}.$$

Dans un contexte plus symbolique ou hétérogène, le Module Synergie peut interroger une ontologie, combiner des scores de règles ou de co-occurrences, ou même implémenter une “co-information” statistique

$$S(i, j) = I(\mathcal{E}_i; \mathcal{E}_j),$$

et retourner ce score au Noyau. La **généralité** du DSL impose souvent que le Module Synergie soit écrit de manière suffisamment polymorphe ou extensible, afin que l'on puisse brancher différents calculs de $S(i, j)$ en fonction des types d'entités \mathcal{E}_i et \mathcal{E}_j .

D'un point de vue **implémentation**, le Module Synergie peut contenir des structures d'index (par exemple, KD-Tree, LSH) lorsqu'il s'agit de recherches k-NN ou d'autres formes de *voisinage*, mais peut aussi reposer sur un pré-calcul ou sur un appel direct à des fonctions (similitude de cosinus, distance euclidienne). La forme abstraite se conçoit aisément : à chaque itération, on parcourt les paires (i, j) actives ou autorisées, et on invoque une fonction

```
synergyModule.getSynergy( $i, j$ )
```

qui renvoie la valeur $S(i, j)$. Sur le plan **algorithmique**, si l'on n'impose aucune parcimonie (k-NN ou ϵ -radius), on se confronte à un calcul en $O(n^2)$, ce qui peut être prohibitif pour de grands n . Les mêmes techniques de réduction de densité (Section 7.2.3.3) s'appliquent alors : on ne fait appel à `getSynergy` que pour un nombre restreint de paires.

Il convient de noter que le Module Synergie est **indépendant** de la mise à jour de $\omega_{i,j}$. Il ne modifie pas lui-même les pondérations : il se contente de calculer ou de fournir $S(i, j)$. Cette **séparation** clarifie l'architecture du SCN : le **Noyau** exécute la règle DSL, tandis que le Module Synergie "répond" aux requêtes $S(i, j)$. Sur le plan **mathématique**, la valeur de $S(i, j)$ peut dépendre directement des entités $\mathcal{E}_i, \mathcal{E}_j$ (par exemple de leurs vecteurs internes), ou s'appuyer sur des états que l'on stockerait dans un autre module (ex. mémoire interne de l'agent). Il reste que, pour chaque couple (i, j) , le Noyau reçoit une simple valeur scalaire, laquelle est injectée dans la formule de mise à jour $\omega_{i,j}(t + 1)$.

En conclusion, le **Module Synergie** occupe une place centrale dans la boucle DSL, assurant que chaque liaison $\omega_{i,j}$ se mette à jour proportionnellement à la "coopération" ou "affinité" mesurée entre \mathcal{E}_i et \mathcal{E}_j . Que l'on utilise une distance exponentielle, une corrélation, une co-information symbolique ou d'autres métriques, tout passe par ce Module Synergie, rendant la solution logicielle plus modulaire et plus souple : la nature concrète de $S(i, j)$ est encodée hors du Noyau, dans un composant spécialisé dont le rôle strict est de **calculer** la valeur de la synergie pour chaque paire active dans le SCN.

B. Module Inhibition/Contrôle

Le **Module Inhibition/Contrôle** joue un rôle complémentaire dans la **dynamique** d'un **Synergistic Connection Network** (SCN), en ajoutant ou en modulant les termes d'**inhibition** (ou d'autres formes de régulation) pendant la mise à jour des pondérations $\omega_{i,j}$. Sur le plan **mathématique**, il s'agit de façonner la descente ou la dynamique du **Deep Synergy Learning (DSL)** au-delà du simple terme $\eta[S(i, j) - \tau \omega_{i,j}]$, de manière à imposer une "**compétition**" ou un "**budget**" aux liens sortants. La présentation qui suit précise la nature de ces fonctions et leur intégration au sein du SCN, où le **Noyau** s'en remet à un composant externe (ce Module) pour appliquer le contrôle nécessaire.

Dans de nombreuses variantes du DSL (cf. Chap. 4.2.2.2 ou 4.4.2), on considère une **inhibition** compétitive : lorsque $\omega_{i,j}$ tend à s'amplifier, on désire qu'une partie de cette croissance "pénalise" les autres liaisons $\omega_{i,k}$ pour $k \neq j$. L'idée courante est d'ajouter dans la **formule** :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \Delta_{\text{update}}(i, j) + \Delta_{\text{inhibition}}(i, j),$$

où l'inhibition peut s'écrire, par exemple, sous la forme

$$\Delta_{\text{inhibition}}(i, j) = -\gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Le **terme** $\gamma > 0$ règle l'intensité de la compétition : plus γ est grand, plus on pousse l'entité \mathcal{E}_i à **sélectionner** un (ou quelques) liens "gagnants", au détriment des autres. Sur le plan **algébrique**, on formalise alors une contrainte de type

$$\sum_j \omega_{i,j}(t) \leq \Omega_{\max}$$

pour chaque i , dans une perspective saturante ou dans un style "winner-takes-most". Ce **contrôle** s'interprète comme une régulation additionnelle empêchant la croissance illimitée de multiples $\omega_{i,j}$ autour du même nœud \mathcal{E}_i .

Dans un SCN **modulaire**, le **Noyau** (Core) détient la boucle principale de mise à jour (cf. 5.2.1.1), mais ne gère pas directement les détails de l'inhibition (ou du contrôle plus vaste). Il se contente d'appeler un composant externe chaque fois qu'il doit ajouter l'effet de la régulation. On obtient un **module** dédié, qu'on peut désigner “Module Inhibition” ou “Module Contrôle” selon l'étendue de ses fonctions, chargé d'appliquer :

$$\Delta_{\text{inhibition}}(i, j) = \text{inhibitionModule.computeTerm}(i, j, \omega_{\cdot, \cdot}(t)).$$

Ainsi, si l'on souhaite passer d'une inhibition “classique” ($-\gamma \sum_{k \neq j} \omega_{i,k}$) à un schéma plus sophistiqué (budget local, saturation, etc.), on modifie ce **Module** sans altérer la logique interne du **Noyau**. De la même façon, si l'on désire ajouter un terme de “bruit” ou d'autres contrôles (clamping, recuit, activation imposée), on étend simplement ce composant pour retourner la portion $\Delta_{\text{contrôle}}$.

Le terme “Inhibition” peut recouvrir différentes formes de **contrôle** :

Compétition Hebbienne : imposer un frein proportionnel à la somme des poids sortants $\sum_k \omega_{i,k}$.

Budget local : $\sum_j \omega_{i,j} \leq \Omega_{\max}$. Cela se traduit parfois par un “coût” ou un “clip” imposé si la somme dépasse Ω_{\max} . On peut écrire :

$$\Delta_{\text{budget}}(i, j) = -\lambda(\sum_m \omega_{i,m} - \Omega_{\max})_+,$$

où $(x)_+ = \max\{x, 0\}$.

Injection de Bruit / Recuit : pour franchir les minima locaux, on peut inclure un “terme stochastique” $\sigma(t) \xi_{i,j}$ comme décrit en recuit simulé. Alors, le **Module Contrôle** ajoute $\Delta_{\text{noise}}(i, j)$ pour réaliser la partie aléatoire.

Rétroaction top-down : dans certains schémas multi-niveaux, un “macro-niveau” envoie un retour (cf. chap. 6.4.1.2) : “renforce les liens entre \mathcal{E}_i et \mathcal{E}_j car ils appartiennent à un cluster macro”, etc.

Ces divers processus se concentrent dans la même entité conceptuelle : le **Module Inhibition/Contrôle**, un “carré” dans l'architecture reliant le **Noyau** via des appels du style “*inhibition.apply(i,j)*”.

La mise à jour globale s'écrit en un unique bloc :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{update}}(i, j) + \Delta_{\text{inhibition}}(i, j) + \Delta_{\text{noise}}(i, j) + \dots$$

ou, dans une version plus condensée,

$$\omega_{i,j}(t+1) = F_{\text{DSL}}\left(\omega_{i,j}(t), S(i, j)\right) + G_{\text{ctrl}}(\{\omega_{i,k}(t)\}_k, \dots).$$

La **séparation** modulaire signifie simplement que la partie **Inhibition/Contrôle** est **extraite** en un **Module**, afin que le code ou l'algorithme correspondant puisse être aisément remplacé ou ajusté. Cela se justifie dans un DSL **flexible** : on veut expérimenter différents **mécanismes** d'inhibition ou de régulation, sans retoucher la boucle itérative du **Noyau**.

En conclusion, le **Module Inhibition/Contrôle** occupe une place analogue au **Module Synergie** : il ne modifie pas ω de son propre chef (sauf dans la logique du cycle imposé par le Noyau), mais renvoie ou applique la **composante** $\Delta_{\text{inhibition}}(i, j)$ qu'il calcule. Les **règles** (inhibition compétitive, budgets locaux, bruit stochastique, etc.) sont encodées dans la formule $\Delta_{\text{inhibition}}$. Le **Noyau** intègre ensuite ce terme dans la mise à jour, assurant une **auto-organisation** plus riche, pilotée par plusieurs composantes (Synergie, Inhibition, Contrôle).

C. Module d'Interface (ajout/suppression d'entités, etc.)

Dans un **Synergistic Connection Network (SCN)** qui se veut adaptable ou évolutif, on peut être amené à **insérer** de nouvelles entités \mathcal{E}_{n+1} ou au contraire à **retirer** une entité devenant obsolète. Sur le plan **mathématique**, il s'agit de **changer** la dimension de la matrice des pondérations ω , passant d'un espace de taille $n \times n$ à $(n+1) \times (n+1)$ (ou

l'inverse) lorsqu'on introduit (ou supprime) une entité. Sur le plan **logiciel**, le **Noyau** (voir 5.2.1.1) n'a pas à gérer lui-même ces questions de "re-dimensionnement" : le **Module d'Interface** prend en charge ces opérations, assurant une **API** de manipulation externe tout en maintenant la cohérence du SCN.

Lorsqu'un SCN évolue de manière **dynamique**, divers scénarios justifient l'ajout ou la suppression d'entités. D'une part, on peut recevoir un **nouveau** flux d'informations conduisant à la création d'une entité inédite (par exemple, l'apparition d'un nouveau robot-agent dans un essaim, ou de nouvelles données dans un système multimodal). D'autre part, on peut devoir **retirer** une entité qui n'est plus pertinente ou qui s'avère défaillante. Formulons cela plus précisément :

$$\omega^{(n \times n)} \mapsto \omega^{((n+1) \times (n+1))},$$

dans le cas où l'on ajoute \mathcal{E}_{n+1} . Il faut alors initialiser les poids $\omega_{n+1,j}$ et $\omega_{j,n+1}$ pour les anciens j , ce qui revient à insérer une ligne et une colonne dans ω . Le **Module d'Interface** offre donc des fonctions de type

$$\text{addEntity}(\mathcal{E}_{\text{new}}),$$

garantissant l'allocation ou l'expansion du tableau des pondérations, avec une initialisation (par exemple $\omega_{\text{new},j} \approx 0$ ou un petit bruit). Inversement, la **suppression** se modélise comme

$$\omega^{(n \times n)} \mapsto \omega^{((n-1) \times (n-1))},$$

lorsqu'on enlève l'entité \mathcal{E}_q et qu'on retire les lignes et colonnes correspondantes. Le Module Interface propose alors une fonction du type

$$\text{removeEntity}(q),$$

pour "nettoyer" la matrice ω (ou un format sparse) et s'assurer que \mathcal{E}_q n'est plus référencée.

Le **Noyau** (Core) se concentre sur la mise à jour itérative de $\omega(t)$, tandis que le **Module d'Interface** assure le "point d'entrée" permettant aux applications externes ou aux administrateurs d'ajuster la configuration du SCN. Sur le plan **logiciel**, on peut envisager une API ou un service :

246.*addEntity(Entity e)*:
Prépare un nouvel ID pour l'entité, insère une ligne et une colonne dans ω , initialise les pondérations $\omega_{\text{new},j}$ et $\omega_{j,\text{new}}$.

247.*removeEntity(ID eID)*:
Retire l'entité eID, supprime la ligne et la colonne associées.

248.*setParameter(paramName, value)*:
Modifie un paramètre ($\eta, \tau, \gamma, \dots$) en notifiant éventuellement le Noyau ou le Module Inhibition.

Cette modularité évite de *déranger* le code cœur du Noyau à chaque fois qu'on veut gérer de nouveaux agents ou entités. Sur le plan **mathématique**, la **dimension** n du SCN est ainsi rendue dynamique, ce qui peut être crucial dans un cadre de robotique ou de multimodalité évolutive, où le flux d'informations s'accroît ou se renouvelle fréquemment.

Le **Module d'Interface** doit, la plupart du temps, insérer ou retirer les entités *entre* deux itérations de mise à jour (plutôt que *pendant* une itération), afin de maintenir la cohérence. Techniquement, on peut concevoir que le Noyau, avant de lancer un "step" suivant, vérifie si des *opérations* en attente (addEntity, removeEntity) doivent être appliquées, met à jour sa structure ω , puis procède au calcul $\omega(t+1) = \dots$. C'est un **schéma** standard : on garantit la synchronisation en décalant l'"insertion" ou la "suppression" au début ou à la fin d'un cycle.

L'Interface ne se limite pas au *ajout/suppression* d'entités : dans bien des cas, on veut aussi **exposer** :

$$\text{setParameter(paramName,value)},$$

pour par exemple ajuster η (taux d'apprentissage) ou γ (facteur d'inhibition) en temps réel. Mathématiquement, cela revient à “déformer” la descente ou la mise à jour $\omega_{i,j}(t+1)$. Dans une architecture modulaire, le **Module d'Interface** transmet cette consigne au Noyau (ou au Module Inhibition), leur indiquant d'appliquer la nouvelle valeur à la prochaine itération. Ce mécanisme rend le SCN **adaptable** : on peut *augmenter* γ pour renforcer la compétition, ou baisser τ pour laisser les poids plus libres, sans redémarrer tout le système.

Dans des scénarios **massifs** ou temps réel, on peut imaginer un “dashboard” ou un “control center” reliant l'extérieur (l'utilisateur, un orchestrateur, un superviseur IA) au SCN. Cette passerelle d'interface se focalise sur le “**comment** ajouter tel agent ? supprimer tel agent ?” ou “**comment** régler η ?”, en interne. Le **Module Interface** se charge de traduire ces requêtes en modifications effectives de la structure Ω ou des paramètres.

Le **Module d'Interface** se révèle donc crucial pour un **SCN évolutif** ou pilotable. Il prend en charge :

249. L'**insertion** d'une nouvelle entité \mathcal{E}_{n+1} , en allouant les lignes/colonnes additionnelles ou en initialisant $\omega_{\text{new},j}$.

250. La **suppression** d'une entité obsolète, en retirant la ligne/colonne correspondante.

251. L'ajustement dynamique des **paramètres** de l'algorithme (comme η, τ, γ), qu'il transmet au Noyau ou aux autres modules.

Sur le plan **mathématique**, cela traduit la capacité à faire varier la dimension n du SCN et à reconfigurer la descente ou l'inhibition en temps réel, ce qui confère au DSL une plasticité véritablement **multi-scénario**. Le **Noyau**, quant à lui, reste focalisé sur l'exécution de la boucle de mise à jour, tandis que ce Module d'Interface gère la “capacité adaptative” : entrée/sortie d'entités, repositionnement des valeurs de η, τ , etc. Ainsi, l'architecture SCN reste **modulaire** : le **Noyau** ne se trouve pas alourdi par les contingences d'ajout ou de retrait, confiées au composant d'Interface qui orchestre ces changements structurels selon les besoins extérieurs.

D. Autres modules éventuels

En plus de **Synergie**, **Inhibition/Contrôle**, et **Interface**, d'autres **modules** peuvent être envisagés :

- **Module Recuit/Brut** (injection de bruit stochastique, planning de “température”),
- **Module d'Observation** (qui calcule la modularité ou extrait les clusters après chaque itération),
- **Module d'Optimisation paramétrique** (qui essaie diverses valeurs de η, τ, γ en cours de route), etc.

Chaque **extension** s'insère dans l'**architecture** par l'ajout d'un bloc additionnel, plutôt que de modifier le **Noyau** en profondeur.

Conclusion

Les **Modules adjoints** du SCN (au-delà du Noyau) apportent la **richesse** fonctionnelle et la **flexibilité** nécessaires :

Module Synergie : assure le **calcul** de $S(i,j)$ (en se basant sur des vectors, des règles, des données robot, etc.).

Module Inhibition/Contrôle : applique des **règles** complémentaires (inhibition compétitive, saturation, recuit), modulant la mise à jour ω .

Module d'Interface : gère l'**ajout** ou la **retrait** d'entités, la **configuration** de paramètres, et constitue une **API** pour l'extérieur.

Cette **architecture modulaire** sépare clairement les **concerns** : le **Noyau** traite la dynamique ω , tandis que chaque module s'occupe de son **domaine** (synergie, inhibition, interface). Mathématiquement, cela permet d'**incorporer** de nouvelles formules ou mécanismes (inhibition non linéaire, synergie binaire, etc.) sans devoir refondre la boucle centrale. En ingénierie, cela facilite la **maintenance**, les **expérimentations**, et la **scalabilité**.

5.2.2. Cycle de Vie du SCN

Au-delà de la simple **séparation** (Noyau vs. Modules) abordée en (5.2.1), il est crucial de voir **comment** un SCN (Synergistic Connection Network) vit **dans le temps** : de l'**initialisation** (qui crée ou charge la matrice ω , installe les entités) à la **boucle itérative** (où s'appliquent la synergie, la mise à jour, l'inhibition, etc.), jusqu'à l'**arrêt** ou la **poursuite** en flux continu. Cette section (5.2.2) détaille ce **cycle de vie**, en le reliant tant aux **aspects mathématiques** (comment on initialise ω , comment on sait qu'on est stabilisé) qu'aux **aspects ingénierie** (comment on charge/sauvegarde les entités, quels signaux déclenchent l'arrêt, etc.).

5.2.2.1. Initialisation : Chargement des Entités et Crédation (ou non) d'une Matrice ω Initiale

Un **Synergistic Connection Network (SCN)** opère toujours à partir d'un ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ et d'une matrice (ou structure) ω regroupant les liaisons $\omega_{i,j}$. La phase d'**initialisation** vise ainsi à introduire l'ensemble des entités dans le système (lecture, import, création dynamique), à attribuer des identifiants et à choisir la condition initiale pour la matrice ω . D'un point de vue **mathématique**, il s'agit de définir $\omega(0) \in \mathbb{R}^{n \times n}$ (ou une structure équivalente) et de **paramétriser** le SCN (fixer η, τ, γ , etc.) pour que la boucle d'auto-organisation puisse commencer dans de bonnes conditions.

Le **premier** acte de l'initialisation consiste à **rassembler** les entités $\{\mathcal{E}_i\}$. Selon l'application, on peut :

(1) Lire un jeu de données sub-symboliques $(\mathbf{x}_1, \dots, \mathbf{x}_n) \in \mathbb{R}^d$,

c'est-à-dire importer des vecteurs ou embeddings depuis un fichier, une base de données, etc.

(2) Importer des entités logiques (symboliques) (règles, ontologies, etc.),

ou

(3) Connecter un flot d'agents (p. ex. robots) (chaque agent devenant \mathcal{E}_i).

D'un point de vue **algorithmique**, on aboutit à un ensemble $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ d'entités numérotées ou identifiées. Sur le plan **mathématique**, on sait qu'on doit manipuler $\omega_{i,j}$ pour $i, j \in \{1, \dots, n\}$. Sur le plan **logiciel**, il faut attribuer une numérotation ou un ID à chaque entité, éventuellement en lien avec un stockage distribué (si le SCN est réparti sur plusieurs nœuds).

Dans un SCN **local**, il suffit de donner des indices $i = 1, \dots, n$ pour indexer la matrice ω . Si le SCN est **distribué** (voir plus loin, Chap. 5.7), on doit répartir les entités \mathcal{E}_i sur plusieurs machines ou partitions, se dotant d'un mécanisme d'allocation d'ID global (ou d'une table de routage). Cette initialisation inclut donc une "cartographie" des entités, permettant à chaque nœud de connaître les indices qu'il gère. D'un point de vue **mathématique**, la matrice ω peut alors être partitionnée par blocs.

Une **question** cruciale est de savoir comment initialiser $\omega_{i,j}(0)$. Plusieurs politiques sont possibles :

$$\omega_{i,j}(0) = \begin{cases} 0, & (\text{zéro total}), \\ \epsilon_{i,j}, & (\text{petit bruit pour casser la symétrie}), \\ \omega_{i,j}^{(\text{saved})}, & (\text{recharge d'un snapshot précédent}). \end{cases}$$

Zéro total : on fixe $\omega_{i,j}(0) = 0$. C'est la solution la plus simple, mais il peut arriver qu'un SCN dont tous les liens sont à zéro ait besoin de plusieurs itérations pour "briser la symétrie" et commencer à se structurer.

Petit bruit : on affecte à chaque $\omega_{i,j}(0)$ une valeur $\epsilon_{i,j}$ aléatoire dans un intervalle $[-\delta, +\delta]$ ou $[0, \delta]$. Cette option est fréquente pour "réveiller" le réseau et éviter un démarrage trop homogène.

Recharge : si l'on dispose d'un état précédent, on le **charge** (avec persistance). Mathématiquement, il suffit d'initialiser $\omega(t = 0) \leftarrow \omega_{\text{saved}}$. On peut ainsi reprendre une simulation, garder le SCN “en mémoire” entre deux sessions, etc.

Dans l'**implémentation**, on peut décider du **format** : dense vs. sparse. Si l'on vise un SCN complet, on attribue une matrice dense de taille $n \times n$. Pour des réseaux clairsemés, on préfère une structure de liste d'adjacence. Le choix repose sur la taille de n et l'hypothèse de densité. Sur le plan **mathématique**, chaque $\omega_{i,j}$ se trouve dans un espace \mathbb{R} (ou \mathbb{R}^+ , si l'on impose non-négativité).

L'initialisation est aussi le moment de **fixer** ou **charger** les paramètres du DSL :

η (taux d'apprentissage), τ (décroissance), γ (inhibition), ω_{\max} (saturation potentielle), etc.

On choisit par exemple $\eta = 0.1$, $\tau = 0.2$, etc. Si l'on veut un mode additif ou multiplicatif (Chap. 4.2.2.1), on le déclare ici. Sur le plan **mathématique**, c'est un jeu de variables dans la mise à jour $\omega_{i,j}(t + 1) = \dots$. Certains algorithmes autorisent une mise à jour **adaptative** de η ou τ , il faut donc prévoir une structure pour stocker ces variables et un moyen de les changer en cours de route.

Avant de démarrer la **boucle** d'auto-organisation, on exécute souvent un **contrôle** :

252. **Nombre** d'entités n cohérent.

253. **Taille** de la matrice ω adaptée.

254. **Paramètres** (ex. η, τ, γ) chargés.

255. **Modules** (Synergie, Inhibition...) opérationnels, pouvant calculer $S(i, j)$ ou $\Delta_{\text{inhibition}}(i, j)$.

C'est une sorte de “checkpoint” mathématico-logique : si tout est validé, on enclenche l'itération $t = 0 \rightarrow 1$. Sinon, on signale une erreur de configuration.

L'**initialisation** dans un SCN constitue donc la **première** phase du cycle de vie. On :

- **Charge** ou **crée** les entités $\{\mathcal{E}_i\}$,
- **Attribue** des IDs et un format (dense, sparse) pour ω ,
- **Choisit** ou **recharge** la matrice initiale $\omega(0)$,
- **Fixe** les paramètres DSL (η, τ, γ , etc.),
- **Contrôle** la cohérence globale.

À l'issue, on est prêt à entrer dans la **boucle** d'itérations, où la mise à jour $\omega_{i,j}(t + 1)$ (et éventuellement la mise à jour d'états internes) pourra démarrer, faisant évoluer le SCN vers la structure auto-organisée visée.

Conclusion :

La **phase d'initialisation** du SCN revêt une importance mathématique et pratique : c'est là qu'on **fixe** les entités $\{\mathcal{E}_i\}$, qu'on choisit la **matrice** $\omega(0)$ (zéro, bruit, ou recharge), et qu'on **paramètre** la dynamique. D'un point de vue ingénierie, cela implique un module d'**Interface** (pour charger ou créer des entités), un possible **module Persistance** (pour recharger ω), et la configuration du **Noyau** (Core). Sans une initialisation cohérente, le SCN risque de partir en oscillations stériles (si η, τ sont mal calibrés) ou d'échouer à discriminer les entités (si on restait strictement à 0). Mathématiquement, c'est la condition initiale $\omega(0)$ et la “setup” paramétrique qui déterminent la trajectoire $\omega(t)$ subséquente.

5.2.2.2. Boucle d’Itérations : Calcul de Synergie, Mise à Jour ω , Éventuelle Inhibition ou Saturation, Extraction de Clusters

Après la **phase d’initialisation** (voir §5.2.2.1) qui a permis de rassembler un ensemble d’entités $\{\mathcal{E}_i\}_{i=1}^n$ et de définir la matrice $\omega(t = 0)$ (ainsi que les paramètres $\eta, \tau, \gamma, \dots$), le **cœur** de la vie d’un **Synergistic Connection Network** (SCN) se déroule dans une **boucle d’itérations**. Cette boucle actualise, pas après pas, la matrice ω selon une **dynamique** de type **Deep Synergy Learning (DSL)**, en s’appuyant sur les **modules** éventuels (synergie, inhibition, observation). Sur le plan **mathématique**, on cherche à faire converger la suite $\omega(t)$ vers un point fixe (ou un cycle, ou un attracteur plus complexe) qui reflète la formation de **clusters** ou de sous-structures. Sur le plan **implémentation**, on exécute un enchaînement stable : calcul de la **synergie** $S(i, j)$, mise à jour de $\omega_{i,j}$, application d’**inhibition** ou de **contrôles** spécifiques, puis extraction ou observation des clusters résultants.

A. Logique Générale de la Boucle

La **boucle itérative** de mise à jour se formalise par l’entrée et la sortie d’un “pas” de temps. À l’itération t , on dispose d’une **matrice** $\omega(t) \in \mathbb{R}^{n \times n}$ (ou un autre format, par exemple sparse). On connaît aussi l’ensemble des **entités** $\{\mathcal{E}_i\}$ et leurs éventuelles représentations (sub-symboliques, symboliques). Les **paramètres** du DSL (η, τ, γ , etc.) sont fixés, ou adaptables si on a un mécanisme paramétrique. De plus, on a chargé les **modules** (synergie, inhibition, etc.) nécessaires. L’**issue** de chaque pas est la nouvelle matrice $\omega(t + 1)$, potentiellement couplée à des informations d’observation (clusters, modularité, etc.), puis on recommence au pas suivant. On répète cette itération jusqu’à atteindre un **critère** (nombre maximal d’itérations, stabilisation, flux continu).

B. Calcul de la Synergie $S(i, j)$

Le **Module Synergie** joue un rôle crucial dans chaque itération : pour que la mise à jour $\omega_{i,j}(t + 1)$ tienne compte de la **coopération** (ou distance, ou similarité) entre \mathcal{E}_i et \mathcal{E}_j , on doit évaluer $S(i, j)$. Sur le plan **mathématique**, on se dote d’une fonction

$$S: (\mathcal{E}_i, \mathcal{E}_j) \mapsto \mathbb{R}^+,$$

que l’on peut voir comme

$$S(i, j) = \mathcal{F}(\mathbf{x}_i, \mathbf{x}_j),$$

selon la nature des entités (vecteurs sub-symboliques, symboles avec ontologie, etc.). D’un point de vue **implémentation**, plusieurs approches sont possibles. On peut recalculer l’intégralité de $\{S(i, j)\}$ en $O(n^2)$ à chaque itération si n n’est pas trop grand, ou si la synergie dépend de variables qui changent à chaque pas. On peut aussi ne recalculer que les paires (i, j) actives dans un scénario parcimonieux (k-NN, ϵ -radius). Dans certains cas, $S(i, j)$ varie peu dans le temps si les entités \mathcal{E}_i ne changent pas de représentation, et l’on peut donc se contenter d’une mise à jour épisodique ou “à la demande”.

Sur le plan **synchrone**, on peut figer la matrice S au début de chaque itération t , l’utiliser pour mettre à jour ω , puis éventuellement la régénérer à l’itération $t + 1$. Cette **séparation** assure la cohérence : chaque “step” emploie la même table de synergies. Dans le cas asynchrone (plus rare), S peut fluctuer en continu, mais cela complique l’analyse mathématique de convergence.

C. Mise à Jour $\omega(t + 1)$

Une fois $S(i, j)$ disponible (pour chaque paire (i, j) considérée), le **Noyau** applique la **formule** du DSL :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)] + \dots$$

La partie ... peut inclure des termes de recuit simulé (ajout d’un bruit $\sigma(t) \xi_{i,j}$), d’inhibition compétitive, ou de budgets. Sur le plan formel, on peut réécrire :

$$\omega_{i,j}(t + 1) = F_{\text{DSL}}(\omega_{i,j}(t), S(i, j)) + G_{\text{ctrl}}(\{\omega_{i,k}(t)\}_{k \neq j}),$$

où F_{DSL} recouvre la “descente locale” $\eta [S(i,j) - \tau \omega_{i,j}(t)]$, et G_{ctrl} est un terme de **contrôle** ou d'**inhibition**. L’application synchrone se met aisément en œuvre via un “double-buffer” : on crée une nouvelle matrice $\omega^{\text{next}}(t+1)$ en lisant $\omega^{\text{current}}(t)$. D’un point de vue **complexité**, si l’on considère chaque paire (i,j) de $\{1, \dots, n\}^2$, on a un coût $O(n^2)$, qu’il est parfois nécessaire de réduire par parcimonie ou parallélisation.

D. Éventuelle Inhibition ou Saturation

De nombreux schémas de DSL prévoient une **inhibition** (chap. 4.2.2.2, 4.4.2) imposant une compétition entre les liens sortants de \mathcal{E}_i . Sur le plan **mathématique**, on ajoute un terme

$$\Delta_{\text{inhibition}}(i,j) = -\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

faisant décroître $\omega_{i,j}(t+1)$ proportionnellement à la somme des autres $\omega_{i,k}$. Il s’agit d’empêcher qu’un nœud \mathcal{E}_i entretienne simultanément trop de liaisons fortes. On peut également imposer une **saturation** ω_{\max} , c’est-à-dire un clipping des valeurs pour éviter que $\omega_{i,j}$ ne devienne trop grande :

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

On peut aussi envisager un budget local $\sum_j \omega_{i,j} \leq \Omega_{\max}$. Sur le plan **implémentation**, on applique souvent l’inhibition et la saturation comme un “post-traitement” de la mise à jour basique $\Delta_{\text{update}}(i,j)$. L’ordre d’application a un léger impact (on peut d’abord faire l’inhibition, puis clipper). Sur le plan **analyse** mathématique, on conceptualise cela comme une **composition** de fonctions inhib/cut :

$$\omega(t+1) = \text{Clip}\left(\text{Inhib}\left(\omega(t) + \Delta_{\text{update}}(t)\right)\right).$$

Cet enchaînement rend la **dynamique** plus non linéaire, mais confère la possibilité d’une auto-organisation sélective.

E. Extraction de Clusters et Observations

L’**objectif** final d’un SCN est souvent de **révéler** la formation de clusters ou de sous-structures. À l’issue de chaque itération (ou toutes les quelques itérations), on peut :

256. **Seuiller** $\omega_{i,j}(t+1)$: on considère les liens $\omega_{i,j} > \theta$ comme “actifs”.

257. **Chercher** les composantes connexes (si $\omega_{i,j}$ est vue comme un graphe), calculer une **modularité** ou un **score** de clustering.

258. **Visualiser** la structure ou fournir un retour dans un “dashboard” (par exemple, un affichage temps réel).

Sur le plan **mathématique**, cette extraction de clusters ou d’indices (silhouette, modularité, etc.) ne modifie pas la mise à jour ω , mais permet de **surveiller** la progression de l’auto-organisation. Il est parfois possible de s’arrêter si la matrice ω ne change plus beaucoup, ou si le score de clustering atteint un plateau.

Conclusion

La **boucle d’itérations** constitue le **processus vivant** d’un SCN, là où la matrice $\omega(t)$ prend forme et se réorganise au fil du DSL. Ce cycle comprend :

- Le **calcul** ou l’**accès** à la synergie $S(i,j)$,
- La **mise à jour** de base de $\omega_{i,j}(t+1)$ (soit additive, soit multiplicative, ou enrichie de termes supplémentaires),
- L’application éventuelle de **mécanismes** : **inhibition** compétitive, **saturation** ou **budget**,
- Un **post-traitement** (extraction de clusters, scores, visualisations) selon la fréquence souhaitée.

Sur le plan **mathématique**, on décrit tout cela par l'opérateur

$$\omega(t+1) = \text{ExtractClusters}\left(\text{Clip}\left(\text{Inhib}(\omega(t), S(t))\right)\right),$$

mais, en pratique, on sépare le cycle en plusieurs modules ou “étapes” pour la lisibilité et la souplesse. Tant qu'on ne détecte pas de critère d'arrêt ou que le système reste en flux continu, on répète ce cycle, laissant émerger et évoluer les **clusters** internes du SCN.

5.2.2.3. Sortie ou Flux Continu : Stabilisation, ou Mode Perpétuel pour l'Apprentissage Continu

Après la mise en place de la **boucle d'itérations** (§5.2.2.2) dans un **Synergistic Connection Network** (SCN), plusieurs scénarios de déroulement se présentent quant à la **durée** de fonctionnement ou la **finalité** de l'exécution. Sur le plan **mathématique**, on se demande si la séquence $\omega(t)$ tend à un **point fixe** (ou à un cycle), auquel cas l'évolution peut être considérée comme convergente, ou si le SCN demeure en **flux continu** en raison de l'arrivée de nouvelles entités ou de la transformation perpétuelle des données. Sur le plan **ingénierie**, il s'agit de décider s'il convient de **stopper** le processus une fois atteint un certain critère de stabilisation, ou bien de **maintenir** le SCN actif en permanence, pour un apprentissage ininterrompu.

A. Sortie ou Arrêt après Stabilisation

Sur le plan **mathématique**, la dynamique d'un SCN se décrit par une équation du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j)\tau \omega_{i,j}(t)] - \gamma \dots$$

ou sous forme plus générale $\omega(t+1) = F(\omega(t))$, éventuellement enrichie d'inhibition ou de budgets. Si les paramètres η, τ, γ sont choisis de manière à garantir la **stabilité**, il peut exister un **point fixe** ω^* tel que $\omega^* = F(\omega^*)$. Dans ce cas, la suite $\omega(t)$ tend vers ω^* pour $t \rightarrow +\infty$, au moins dans un régime local (ou global, si la non-linéarité est contrôlée).

Une façon d'**implémenter** cette convergence est de surveiller, à chaque itération, la **différence** entre $\omega(t+1)$ et $\omega(t)$. Par exemple, on peut calculer la norme

$$\| \omega(t+1) - \omega(t) \|_2$$

ou toute autre mesure de dissimilarité. Lorsque celle-ci passe sous un **seuil** ε (et le reste pendant quelques itérations), on juge que le SCN est **pratiquement convergé**. Un “Module d’Observation” (ou un simple mécanisme dans le **Noyau**) peut ainsi déclencher l'arrêt du programme.

Le code peut aussi être muni d'un **critère supplémentaire** (temps maximum, nombre d'itérations maximum) pour éviter une boucle sans fin si la convergence n'est jamais rigoureusement atteinte. Dans un usage **off-line**, typiquement, on exécute le SCN sur un lot d'entités, puis on s'arrête quand la structure ω s'est figée. Sur le plan **mathématique**, on aboutit à un ω^* qui reflète la répartition des entités en clusters, ou plus généralement la configuration stable.

Juste avant l'arrêt effectif, il est courant de **sauvegarder** la matrice ω^* (persistance), afin de pouvoir la réutiliser plus tard (rechargement) ou de l'exploiter dans un autre module (étiquetage de clusters, visualisation, etc.).

B. Mode Perpétuel ou Flux Continu (Apprentissage Continu)

Dans certains environnements (robotique multi-agent, systèmes de recommandation en ligne, traitement de flux de données), le SCN n'est pas censé s'achever après un certain nombre d'itérations, car les **données** ou les **entités** continuent à évoluer ou à arriver. On parle alors d'un **SCN en flux continu**, où la mise à jour $\omega(t+1)$ se poursuit indéfiniment, accompagnée de modifications potentiellement incessantes du calcul de synergie $S(i,j)$. D'un point de vue **mathématique**, la dynamique n'est plus strictement $\omega(t+1) = F(\omega(t))$ avec un opérateur fixe, mais peut dépendre du temps :

$$\omega(t+1) = F(\omega(t), S(\cdot, \cdot, t)),$$

car de nouvelles entités \mathcal{E}_{n+1} peuvent survenir, modifiant la taille de la matrice, ou parce que les entités existantes changent de représentation.

Cette **variabilité** implique qu'on n'atteint pas forcément un point fixe, mais plutôt qu'on "poursuit" un **équilibre** mobile ou une adaptation permanente. On peut y voir un système dynamique à paramètres variant dans le temps (non autonome). Sur le plan **implémentation**, on laisse la boucle itérative fonctionner en continu, et on traite les modifications (ajout/suppression d'entités, mise à jour de synergie, etc.) au fur et à mesure. Un "Module Interface" se charge d'**injecter** les nouvelles entités, en allouant la nouvelle ligne/colonne de ω , puis la dynamique reprend son cours, s'ajustant à ce changement.

Cette perspective rappelle l'**apprentissage en ligne** ou l'**apprentissage continu** : à chaque step, on effectue une mise à jour incrémentale de ω . Parfois, le SCN peut connaître des phases **quasi stationnaires** lorsqu'il n'y a pas de perturbation majeure, puis se réorganiser subitement quand un "choc" se produit (ex. un afflux de nouvelles entités ou un changement de représentation).

C. Bascule Entre Sortie et Flux Continu

Un même SCN peut, en pratique, être utilisé dans deux **modes** :

Mode "One-shot" ou "Offline" : on lance la boucle avec un ensemble fixe $\{\mathcal{E}_i\}$ et on arrête après convergence ou après un nombre de pas défini. On récupère alors ω^* et les clusters.

Mode "Online" ou "Continu" : le SCN tourne indéfiniment, prêt à accepter de nouvelles entités via le "Module Interface" (qui ajoute une ligne/colonne), et la matrice ω s'ajuste en temps réel. On ne parle plus d'arrêt, mais d'**écoulement** permanent, parfois ponctué de vérifications (par exemple, un mini-critère de convergence temporaire, avant qu'une nouvelle perturbation n'apparaisse).

Il est également possible de démarrer en "offline" pour amorcer une structuration, puis de **basculer** en "online" si l'application l'exige, c'est-à-dire laisser le SCN s'adapter aux évolutions ultérieures. Sur le plan **mathématique**, cette bascule revient à transformer un opérateur statique F en un opérateur dynamique dans le temps, sans rebooter la matrice ω . Sur le plan **implémentation**, il suffit de ne plus enclencher le "Critère d'Arrêt" et de laisser le SCN réagir aux notifications d'ajout/suppression d'entités.

Conclusion

La **phase finale** dans le cycle de vie d'un SCN ne se résume pas à un simple arrêt : selon l'objectif, on peut :

Terminer l'exécution après stabilisation (ou nombre maximal d'itérations), en récupérant la structure ω finale et en l'analysant (clusters, etc.). D'un point de vue **mathématique**, on se situe alors dans un cadre de **convergence** classique, $\|\omega(t+1) - \omega(t)\| \leq \varepsilon$.

Prolonger l'exécution en mode **flux continu**, où l'on n'a pas d'arrêt global. Le SCN reste actif, prend en compte les éventuelles modifications d'entités ou de synergie, et se reconfigure en continu. Ce "mode perpétuel" correspond à un système dynamique forcé, plus complexe à analyser, mais indispensable pour des applications en ligne (robotique, streaming, intelligence distribuée).

Ainsi, le **Noyau** et les **Modules** (Synergie, Inhibition, Interface) gardent un rôle identique : ils font la mise à jour $\omega(t+1)$. Seule la **politique** de sortie ou de poursuite diffère, assurant que la même **architecture** puisse servir aussi bien dans un scénario "offline" (stabilisation-clusters) que dans un cadre "online" (adaptation permanente).

5.2.3. Exemples d'Organisation

Au-delà du **cycle de vie** (5.2.2) et de la séparation **Noyau vs. Modules** (5.2.1), il est instructif de voir **comment** un SCN peut s'**organiser** dans des configurations variées, en fonction des **contraintes** et des **échelles**. Parmi les solutions les plus courantes :

Un **mono-module** minimal, où tout le code (calcul de synergie, mise à jour, inhibition, etc.) se trouve dans un **même bloc** (5.2.3.1),

Une **architecture plus modulaire**, où chaque fonction (synergie, mise à jour, interface...) est un module ou une classe distincte (5.2.3.2),

Une **architecture distribuée**, déployant plusieurs **sous-SCN** collaborant entre eux (5.2.3.3).

5.2.3.1. Mono-module minimal (tout dans un même bloc)

Lors de la mise en œuvre d'un **Synergistic Connection Network (SCN)**, il est possible de concevoir une **architecture** extrêmement simple et compacte, où toutes les fonctions nécessaires (stockage et mise à jour de ω , calcul de la synergie, inhibition, extraction de clusters, etc.) se retrouvent dans un **unique** module (ou script). Cette approche, qualifiée de "mono-module minimal", vise la **simplicité** et la **rapidité** de développement pour des projets de petite taille ou des expérimentations rapides. Sur le plan **mathématique**, cela revient à écrire l'ensemble de la boucle d'itérations et des opérations connexes au sein d'un seul bloc de code, sans répartition en classes ou en modules séparés.

Dans ce "mono-module", on se contente d'un **unique** script, classe ou notebook — éventuellement appelé *SCN_all_in_one.py* — qui regroupe toutes les étapes. Il initialise la matrice $\omega(t = 0)$, gère la boucle d'itérations $\omega(t + 1) = F(\omega(t))$, calcule la **synergie** $S(i, j)$, applique l'**inhibition** si nécessaire, et, s'il y a lieu, **extrait** les clusters (par un simple code de post-traitement) dans le même fichier. D'un point de vue **organisation**, on peut imaginer un pseudo-code :

(1) Définir entités $\{\mathcal{E}_i\}_{i=1}^n$, charger ou créer $\omega(t = 0)$,

(2) Pour $t = 1 \dots T_{\max}$:
$$\left\{ \begin{array}{l} \text{Calculer } S(i, j) \\ \omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)] - \gamma \sum_k \dots \\ (\text{Application d'une saturation ou d'un clipping si besoin}) \\ \text{Extraction possible de clusters (optionnel)} \end{array} \right.$$

(3) Fin, sauvegarde ou visualisation.

Sur le plan **ingénierie**, tout cela est concentré dans un **même** bloc, sans appel externe à un "Module Synergie" ou un "Module Inhibition" dédié. Les variables globales η, τ, γ résident dans le même fichier, et le calcul de $S(i, j)$ s'inscrit dans une fonction locale.

Le premier intérêt de ce "**tout-en-un**" est la facilité de mise en place. Pour un **prototype** académique ou une démonstration dans un **projet réduit**, il suffit d'un script unique :

259. On déclare un tableau ou un vecteur pour chaque entité \mathcal{E}_i .

260. On crée $\omega(t = 0)$ en initialisant à zéro ou à un bruit aléatoire.

261. On code la boucle $\omega(t + 1) = \omega(t) + \eta[S(i, j) - \tau \omega(t)] + \dots$

262. On insère, si nécessaire, quelques lignes pour l'inhibition (par ex., $-\gamma \sum_{k \neq j} \omega_{i,k}$).

263. On ajoute éventuellement un post-traitement pour détecter les clusters (un simple "threshold" sur $\omega_{i,j}$ et un parcours en composantes connexes).

Cette centralisation rend le script particulièrement **facile à comprendre** pour des novices qui veulent voir la **formule** de mise à jour en action. Sur le plan **mathématique**, tout reste sous la forme d'une unique "big loop" :

$$\text{for } t = 1 \dots T_{\max}: \quad \omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)] - \gamma \sum_k \dots$$

et ainsi de suite.

Dès qu'on souhaite aller au-delà d'un **essai** isolé, la **structure** en un bloc unique devient handicapante. Sur le plan **évolutif**, chaque fois qu'on veut changer la définition de $S(i,j)$ (par exemple passer d'une distance euclidienne à une co-information symbolique), on doit altérer ce même code, parfois avec des embranchements (*if/else*). En outre, si l'on désire intégrer un **recuit simulé** ou un **module** d'ajout dynamique d'entités (pour un flux continu), on insère de plus en plus de blocs logiques dans le même script, risquant de produire un "**super-monolithe**" illisible.

Sur le plan **maintenance**, un gros bloc rend plus difficile la **collaboration** entre développeurs. Un collègue voulant modifier l'inhibition ne peut isoler un "fichier inhibition.py" : il doit naviguer dans des centaines de lignes d'un script commun. Sur le plan **test** unitaire, il est plus compliqué de tester chaque composant séparément : tout est couplé.

Sur le plan **performance** ou **scalabilité**, un script unique ne prévoit généralement pas d'architecture distribuée ou de parallélisation modulaire. En cas de passage à un $O(n^2)$ massif, il devient périlleux d'adapter ce code monolithique pour incorporer des techniques HPC ou des structures d'indexation complexes.

Rien, d'un point de vue **équations**, n'empêche de **fusionner** dans un même bloc :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{update}}(i,j) + \Delta_{\text{inhib}}(i,j)$$

et, si l'on souhaite, d'appeler en fin de boucle une fonction *extract_clusters(w)* qui identifie les composantes fortes. Sur un **petit** dataset (quelques centaines d'entités), c'est parfois la méthode la plus rapide pour un **proof of concept** : on code tout en une journée et on obtient un résultat. La **réduction** en modules n'est pas obligatoire du point de vue mathématique : c'est simplement un **confort** d'ingénierie pour la suite.

Les **petits projets** ou les **travaux pratiques** dans un cours peuvent recourir à ce style "all-in-one". L'étudiant ou le chercheur indique directement la **formule** :

$$\omega_{i,j} \leftarrow \omega_{i,j} + \eta [S(i,j) - \tau \omega_{i,j}] - \gamma \sum_{k \neq j} \omega_{i,k},$$

dans une boucle Python, invoque éventuellement un $\exp(-\| \mathbf{x}_i - \mathbf{x}_j \|)$ pour la synergie, et récupère au final l'état ω . C'est idéal pour un "MVP" (Minimum Viable Product).

En revanche, une extension ultérieure (passer en mode streaming, ou introduire un nouveau calcul de S) demandera de **refactoriser** la base de code en modules séparés (cf. §5.2.3.2).

Conclusion

Le **mono-module minimal** consiste à mettre toutes les composantes du SCN (données, synergie, mise à jour ω , inhibition, etc.) dans un **unique** bloc de code. Cette approche se justifie par :

Simplicité pour des **petits** projets ou des démonstrations : une seule boucle, facile à lire.

Rapidité de prototypage : on écrit directement la formule du DSL sans structurer.

Toutefois, elle montre vite des **limites** pour un usage extensif (ajout de nouvelles entités, recuit, maintenance, test, distribution). On recommande donc cette formule "all-in-one" essentiellement dans des contextes académiques restreints ou pour valider un concept. Dès qu'on anticipe une évolution du code, il convient de se tourner vers une **architecture plus modulaire** (cf. §5.2.3.2) qui scinde la synergie, la mise à jour ω et les autres mécanismes en modules distincts.

5.2.3.2. Architecture Modulaire (Chacun s'occupe d'une fonction)

Lorsqu'un **Synergistic Connection Network (SCN)** a pour ambition d'être maintenu, étendu, ou intégré à des systèmes plus vastes, il est recommandé d'adopter une **architecture modulaire** plutôt que de concentrer tout le code et toutes les fonctionnalités dans un monolithe (cf. §5.2.3.1). Cette architecture consiste à **disséquer** le SCN en

plusieurs **modules** ou **composants**, chacun dédié à une tâche particulière. Sur le plan **mathématique**, on continue de manipuler la même dynamique $\omega(t+1) = F(\omega(t))$ (ou plus détaillée), mais on en sépare clairement les différents “blocs” de calcul. Sur le plan **ingénierie**, on met en place des **interfaces** et des classes (ou fichiers) distincts, afin de pouvoir tester, modifier ou réutiliser chaque composante sans retoucher l’ensemble.

A. Principe Général de la Modularisation

Il s’agit de **découpler** les responsabilités au sein du SCN en divers “modules”, afin de mieux maîtriser la complexité et d’**éviter** que tous les calculs (synergie, mise à jour ω , inhibition, interface, etc.) s’entassent dans un même fichier. On établit donc, au minimum, les blocs suivants :

Un **Noyau (Core)** qui gère la **matrice** ω , orchestre la **boucle** d’itérations et assure la **mise à jour** de base (Δ_{update}).

Un **Module Synergie** qui calcule la **fonction** $S(i,j)$ suivant la nature des entités (euclidienne, cosinus, co-information, etc.).

Un **Module Inhibition/Contrôle** qui introduit des mécanismes complémentaires (inhibition compétitive, saturation, budgets locaux, recuit...).

Un **Module d’Interface** pour gérer l’ajout/suppression d’entités, modifier les paramètres (η, τ, γ), et connecter éventuellement le SCN à un système extérieur (tableau de bord, webservice, etc.).

(Optionnel) Des **modules** supplémentaires, comme un “Module Observateur” (déttection de clusters, visualisation), un “Module RecuitSim” (injection de bruit pour franchir les minima locaux), un “Module Persistence” (sauvegarde/recharge de l’état ω).

Avantages

Dès qu’on s’éloigne d’un simple **prototype**, séparer en modules présente plusieurs atouts notables :

- **Maintenance** améliorée : si la forme de $S(i,j)$ doit évoluer, on ne touche qu’au Module Synergie ; si les règles d’inhibition changent, on modifie le Module Inhibition.
- **Extensibilité** : on peut ajouter un **nouveau** module pour un recuit simulé ou un mécanisme d’apprentissage hiérarchique sans devoir “recoder” le cœur de la boucle.
- **Clarté Mathématique** : chaque bloc de la formule $\omega_{i,j}(t+1) \leftarrow \omega_{i,j}(t) + \dots$ peut être isolé, testable, et documenté. On sait où se trouve $\Delta_{\text{inhibition}}$ et où se trouve Δ_{update} .
- **Test Unitaire** : on peut tester chaque module en isolement (par exemple, vérifier que le Module Synergie renvoie le score $S(i,j)$ correct pour un duo d’entités) avant de lancer l’ensemble.

Implémentation

Logicielle

On peut opter pour une approche **orientée objet** (OOP) en définissant, par exemple, une classe *SCNCore* où se trouvent la matrice ω et la boucle principale, ainsi que des classes *SynergyModule*, *InhibitionModule*, *InterfaceModule*, etc. Chacune possède des méthodes claires (ex. *getSynergy(i,j)*, *applyInhibition(w, i, j)*...), et le **Noyau** les appelle dans le bon ordre à chaque itération. Alternative, on peut rester en mode **fonctionnel** en dispersant les fonctions dans différents fichiers, tout en ayant un “fichier central” qui enchaîne les appels.

Sur le plan **mathématique**, l’équation de la mise à jour globale reste :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{update}}(i,j) + \Delta_{\text{inhibition}}(i,j) + \dots$$

Le Module Synergie intervient pour fournir $S(i,j)$ à la partie $\Delta_{\text{update}}(i,j) = \eta [S(i,j) - \tau \omega_{i,j}(t)]$. Le Module Inhibition fournit $\Delta_{\text{inhibition}}(i,j)$. L’**ordre** d’application peut être séquentiel, reflétant la composition des opérateurs (cf. chap. 4.2.2.2).

B. Exemple de Flux Modulaire en une Itération

Pour illustrer la **séquence** dans un cycle d’itérations :

Pré-calcul ou accès de $S(i, j)$ via le Module Synergie. On peut générer une matrice \mathbf{S} , ou calculer $S(i, j)$ “à la demande” pour chaque (i, j) .

Mise à Jour de base : le Noyau applique l’équation additive ou multiplicative pour $\omega_{i,j}$. Ex. $\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)]$.

Inhibition/Contrôle : on appelle alors le Module Inhibition qui peut réaliser $\omega_{i,j}(t + 1) \leftarrow \omega_{i,j}(t + 1) - \gamma \sum_{k \neq j} \omega_{i,k}(t)$, ou imposer un “budget”.

Saturation (si on veut clipper ω entre 0 et ω_{\max}). Ce peut être un sous-module du contrôle, ou un post-traitement.

Extraction de Clusters / Observations : un Module Observateur, si défini, peut lire la matrice $\omega(t + 1)$ pour en déduire les composantes connexes, un score de modularité, etc.

Interface : en fin de boucle, le Module Interface peut éventuellement enregistrer la progression, traiter une requête d’ajout d’entité, ou modifier η .

Ainsi, on sépare clairement **qui** fait quoi. Les formules mathématiques s’assemblent comme des briques, et le code reflète ce découpage.

C. Comparaison avec l’Approche Mono-Module

Contrairement à un “**all-in-one**” script (voir §5.2.3.1), la modularisation semble plus lourde à configurer pour un très petit projet. Cependant, dès que l’on souhaite :

Ajouter un **nouveau** mode de synergie S (par exemple, un calcul de co-information au lieu d’une distance)

Varier l’**inhibition**

Intégrer un “**Module Recuit**” (ajout de bruit aléatoire pour franchir des minima locaux)

Gérer l’**arrivée** d’entités en flux continu

on réalise qu’une structure modulaire simplifie grandement la maintenance. Chaque module peut évoluer ou être remplacé sans perturber le **Noyau**. Sur le plan **mathématique**, cette séparation clarifie également l’analyse : la partie $\eta[S(i, j) - \tau \omega]$ diffère de la partie “inhibition”, on peut donc étudier leur impact respectif sur la convergence de manière indépendante, puis considérer leur composition.

D. Extensibilité et Scalabilité

Lorsqu’on parle de **mise à l’échelle** (plus grand nombre d’entités) ou d’**extension** (nouveau type d’entités, nouveau paradigme de synergie), l’architecture modulaire fournit un socle robuste. On peut brancher un “Module Parallelisation” ou un “Module Distribution” (voir §5.2.3.3) pour répartir la matrice ω sur plusieurs nœuds. Le **Noyau** demeure responsable de la logique de mise à jour, mais le “Module Distribution” peut implémenter un mécanisme d’échange ou de partition des données. Sur le plan **mathématique**, on reste fidèle à la même équation, mais on modifie la “plomberie” logicielle.

Cette capacité à empiler de nouveaux modules (Recuit, Observateur, etc.) est cruciale pour un SCN utilisé dans un cadre de recherche à long terme ou un projet industriel, où les besoins évoluent.

E. Conclusion

Une **architecture modulaire** pour un SCN implique :

- Un **Core** (Noyau) centré sur la matrice ω et la boucle d’itérations,
- Des **Modules** (Synergie, Inhibition/Contrôle, Interface, Observateur, etc.) qui viennent se **brancher** dessus et apporter des fonctions spécialisées.

Cette séparation rend la **structure** plus **solide** mathématiquement (chaque composant a des formules et un rôle précis) et **manœuvrable** au niveau ingénierie (chacun s’occupe d’une fonction). Par rapport au **mono-module** (5.2.3.1), on gagne en maintenabilité, extensibilité, et facilité d’évolution, au prix d’un effort de conception initial plus important. Pour des **projets** de recherche avancée ou des applications industrielles (gros volumes, scénarios évolutifs), c’est l’option qui offre le meilleur **équilibre** entre complexité et robustesse.

5.2.3.3. Architecture distribuée (plusieurs sous-SCN coopérant)

Lorsque le **nombre d’entités** devient très grand (n potentiellement de l’ordre de 10^5 à 10^7) ou que les entités elles-mêmes sont **réparties** sur plusieurs sites (multi-robots géographiquement dispersés, données multimodales venant de différentes sources, etc.), il n’est plus forcément possible ni souhaitable de gérer **un** unique SCN centralisé. On peut alors opter pour une **architecture distribuée** où plusieurs **sous-SCN** (chacun manipulant un sous-ensemble des entités ou un certain “domaine”) collaborent pour former un ensemble plus vaste.

Cette approche répond aussi à des besoins de **scalabilité** (on ne peut stocker $O(n^2)$ pondérations sur un seul nœud) et de **résilience** (en cas de panne, on ne veut pas perdre tout le SCN). D’un point de vue mathématique, on se situe dans un **système** où la matrice ω est “morcelée” ou “partitionnée”, et où les mises à jour dans chaque bloc coopèrent pour tendre vers une organisation globale.

A. Principe : plusieurs sous-SCN

Dans une **architecture distribuée** destinée à un **Synergistic Connection Network (SCN)**, le principe fondamental consiste à **partager** l’ensemble d’entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ en plusieurs **sous-ensembles** disjoints, chacun étant géré par un **sous-SCN** local. L’objectif est de répartir la **charge** de calcul et de stockage lorsque le nombre total n devient très grand (plusieurs centaines de milliers, voire des millions d’entités), ou lorsque les entités sont **géographiquement** dispersées (agents robotiques sur différents sites, flux multimédias provenant de régions variées, etc.). Sur le plan **mathématique**, on se retrouve avec une matrice ω qui, au lieu d’être stockée et actualisée de manière unique et centralisée, se trouve morcelée ou **partitionnée** en blocs (sous-matrices), chaque bloc traitant ses propres entités.

Pour mettre en place plusieurs **sous-SCN**, on commence par **diviser** l’ensemble $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ en plusieurs parties $\mathcal{V}_1, \dots, \mathcal{V}_m$, par exemple selon un découpage géographique, un découpage thématique, ou un simple partitionnement en blocs d’effectif à peu près égal. On obtient alors

$$\mathcal{V}_1 \cup \mathcal{V}_2 \cup \dots \cup \mathcal{V}_m = \{\mathcal{E}_1, \dots, \mathcal{E}_n\}, \quad \mathcal{V}_p \cap \mathcal{V}_q = \emptyset \text{ pour } p \neq q.$$

Chaque sous-SCN, noté SCN_p , gère localement les pondérations $\omega_{i,j}^{(p)}$ correspondant aux liens où $i, j \in \mathcal{V}_p$. Sur le plan **stockage**, cela signifie qu’il existe une (sous-)matrice $\omega^{(p)}$ que SCN_p manipule de manière analogue à ce qui a été décrit pour un SCN simple. Sur le plan **algorithmique**, SCN_p exécute sa **boucle** d’auto-organisation (calcul de synergie, mise à jour ω , etc.) au sein du bloc \mathcal{V}_p .

Si les entités i et j appartiennent à des blocs différents \mathcal{V}_p et \mathcal{V}_q , on se heurte à la question : comment traiter $\omega_{i,j}$? Dans un SCN **totalement** distribué, il est a priori possible que \mathcal{E}_i dans \mathcal{V}_p soit fortement synergique avec \mathcal{E}_j située dans \mathcal{V}_q . On peut concevoir plusieurs schémas.

Le premier schéma consiste à **calculer** (ou au moins **approcher**) les pondérations inter-blocs, c’est-à-dire $\omega_{i,j}$ pour $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$. On peut dans ce cas parler de “liens inter-SCN”, que l’on stocke dans un “**module passerelle**” ou dans une base partagée. D’un point de vue **mathématique**, chaque SCN_p peut avoir besoin, lors de la mise à jour, d’un terme $\omega_{i,j}$ quand $j \notin \mathcal{V}_p$. Cela suppose des échanges de données entre SCN_p et SCN_q . On peut néanmoins décider d’**ignorer** ou de **sous-échantillonner** ces liens inter-blocs si la synergie est censée être trop faible ou trop rare, réduisant ainsi la complexité globale.

Une autre approche, moins coûteuse en communications, est de n'entretenir les liaisons inter-blocs qu'**épisodiquement** (synchronisation toutes les T itérations), ou de **négliger** explicitement les liens trop distants. On peut écrire, par exemple, un protocole de synchronisation dans lequel SCN_p et SCN_q échangent seulement la portion de ω relative aux entités qui semblent entretenir une synergie notable. Sur le plan **mathématique**, la mise à jour de $\omega^{(p)}(t+1)$ peut faire intervenir un terme de “dépendance inter-blocs” :

$$\omega_{i,j}^{(p)}(t+1) = F^{(p)}\left(\omega_{i,j}^{(p)}(t), S^{(p)}(i,j), \Delta_{\text{inter}}^{(p)}(t)\right),$$

où $\Delta_{\text{inter}}^{(p)}(t)$ regroupe les informations (indices de synergie, pondérations partielles) reçues des autres sous-SCN. Cela se rapproche d'un système **multi-agent** où chaque agent (ici, SCN_p) a sa propre mise à jour locale, mais s'échange parfois des données globales pour harmoniser le tout.

Pour aboutir à une organisation **globale**, il faut que ces sous-SCN ne travaillent pas totalement en vase clos. À intervalles réguliers, on peut effectuer une **coalescence** ou une **synchronisation partielle** : les différents SCN_p consolident les pondérations inter-blocs ou partagent des **vecteurs** d'état (par exemple, la somme $\sum_j \omega_{i,j}^{(p)}$ pour un i qui migrerait dans un autre bloc). D'un point de vue **mathématique**, cela se formalise comme une **itération distribuée** : chaque SCN_p gère les paires internes $\omega_{i,j}^{(p)}$, puis on superpose un opérateur $\text{InterSync}(\omega^{(1)}, \dots, \omega^{(m)})$ qui connecte ou rapproche les valeurs aux frontières. On peut également, dans certains scénarios, faire des “clusters” majoritairement internes, en acceptant que les liens inter-blocs soient minimes ou rarissimes.

Sur le plan **ingénierie**, chaque sous-SCN peut être un **processus** (ou un ensemble de threads) tournant sur un nœud différent. Les communications sur le “réseau” (au sens informatique) portent sur les indices d'entités, les bribes de matrice ω , etc. Sur le plan **mathématique**, on se rapproche d'un “réseau de SCN” ou d'un “réseau de nœuds DSL”, chacun pilotant un sous-ensemble.

Lorsqu'il s'avère nécessaire de distribuer le **SCN** pour gérer un **grand** nombre d'entités ou pour implanter le système sur des sites géographiquement dispersés, on procède en créant plusieurs **sous-SCN** $\text{SCN}_1, \dots, \text{SCN}_m$. Chacun manipule une portion des entités $\mathcal{V}_p \subset \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ et stocke localement la sous-matrice $\omega^{(p)}$. Les liens inter-blocs (entités appartenant à des sous-ensembles différents) sont gérés au travers de **protocoles** de synchronisation ou de communications limitées. Sur le plan **mathématique**, on peut écrire la dynamique de chaque sous-SCN comme $\omega^{(p)}(t+1) = F^{(p)}\left(\omega^{(p)}(t), \Omega^{\text{inter}}(t)\right)$, où $\Omega^{\text{inter}}(t)$ regroupe les données échangées avec les autres sous-SCN. Cette approche **distribuée** permet de franchir les limites de mémoire et de calcul qu'impliquerait une matrice $O(n^2)$ trop imposante, tout en préservant une **cohérence** d'ensemble via les mécanismes de coopération entre blocs.

B. Organisation Logicielle d'une Architecture Distribuée

Lorsqu'il devient indispensable de gérer un **Synergistic Connection Network (SCN)** sur plusieurs machines ou serveurs, il est naturel de **distribuer** non seulement les données (entités et sous-matrices ω) mais aussi la boucle même d'auto-organisation du **Deep Synergy Learning (DSL)**. Cette mise en place “multi-nœuds” ou “multi-blocs” doit alors prendre en compte divers aspects : comment **partitionner** les entités, comment synchroniser les sous-SCN locaux, comment véhiculer les pondérations inter-blocs, et comment assurer la **scalabilité** dans un contexte où l'on peut avoir des centaines de milliers ou des millions d'entités globales.

Une architecture distribuée repose d'abord sur la définition de **nœuds** (ou “workers”, “serveurs”), chacun exécutant un **SCN local**. D'un point de vue **logiciel**, on peut imaginer que chaque nœud exécute :

- 264. Un **Core** local qui gère la sous-matrice $\omega^{(p)}$ correspondant aux entités de son bloc \mathcal{V}_p .
- 265. Des **Modules** spécialisés (Synergie, Inhibition, Interface) adaptés au bloc local.
- 266. Un mécanisme de **communication** (via TCP/IP, RPC, ou un bus de messages) pour échanger des données avec les autres nœuds.

Sur le plan **mathématique**, si l'ensemble global $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ est partitionné en $\mathcal{V}_1, \dots, \mathcal{V}_m$, le nœud p prend en charge toutes les liaisons $\omega_{i,j}$ où $i, j \in \mathcal{V}_p$. En local, il opère sa propre boucle “mise à jour $\omega_{i,j}^{(p)}(t+1) = \dots$ ”. Cela donne

naissance à des **sous-SCN** SCN_p . Pour gérer les liens inter-blocs ($i \in \mathcal{V}_p, j \in \mathcal{V}_q$), on recourt à un mécanisme d'échanges entre nœuds.

Le **rythme** des communications inter-nœuds détermine le type de **synchronisation** dans l'architecture distribuée. On discerne deux grandes approches :

1. Approche Synchrone

Tous les sous-SCN SCN_p réalisent leur mise à jour locale (pour leurs entités internes) sur un même “tour” ou “round”, puis s’arrêtent à une “barrière” de synchronisation pour **échanger** ou **agréger** les pondérations (ou au moins les informations pertinentes) concernant les liens inter-blocs. Par exemple, on peut imaginer un cycle :

- (i) Chaque SCN_p met à jour $\omega^{(p)}(t + 1)$,
- (ii) Attente d'une barrière,
- (iii) Échange de données sur inter-blocs,
- (iv) Prochaine itération.

Cette approche facilite l'**analyse** mathématique : on peut considérer que chaque round complet aboutit à un opérateur global $G(\omega^{(1)}, \dots, \omega^{(m)})$. En contrepartie, on tolère une **latence** plus élevée (chaque nœud doit attendre les autres avant de poursuivre), ce qui peut ralentir le système pour un grand nombre de nœuds.

2. Approche Asynchrone

Ici, chaque sous-SCN tourne en continu et **demande** les valeurs $\omega_{i,j}$ ou \mathbf{x}_j (pour le calcul de synergie) au fur et à mesure à d’autres nœuds, sans synchronisation globale. Cela signifie qu’on peut avoir des liens $\omega_{i,j}$ “en retard” d’un ou plusieurs tours, donc des **incohérences** momentanées plus élevées. On gagne par contre en **efficacité** si le réseau est très large et que l’on n’a pas envie de bloquer tout le monde à chaque round. Sur le plan **mathématique**, on aboutit à une mise à jour $\omega^{(p)}(t + 1) = F^{(p)}(\omega^{(p)}(t), (\text{informations inter-blocs potentiellement datées}))$. L’étude de convergence devient plus complexe, souvent liée aux techniques de systèmes dynamiques asynchrones ou aux algorithmes de type “Gossip”.

Inter-blocs : meta-SCN et résumé des synergies

Pour **connecter** logiquement les sous-SCN, on peut définir un “**meta-SCN**” au niveau supérieur : un graphe dont chaque nœud représente un **bloc** \mathcal{V}_p . Chaque arête représente un **résumé** des liens inter-blocs (par exemple la somme des pondérations $\omega_{i,j}$ pour $i \in \mathcal{V}_p, j \in \mathcal{V}_q$ supérieures à un certain seuil, ou la moyenne des synergies fortes). Ce meta-SCN n'a pas nécessairement le même rôle qu'un SCN classique, mais il peut **guider** l'échange de données : si la synergie inter-blocs $\mathcal{V}_p \leftrightarrow \mathcal{V}_q$ s'avère très faible, on peut réduire la fréquence de synchronisation entre SCN_p et SCN_q .

D'un point de vue **implémentation**, chaque sous-SCN peut posséder un “cache” local des informations sur les entités extérieures (\mathbf{x}_j si on calcule la distance) ou sur $\omega_{i,j}$ inter-blocs, mis à jour selon un protocole de communication. Les mathématiciens y verrait un “système dynamique couplé”, où chaque $\omega^{(p)}$ évolue au gré de données reçues de $\omega^{(q)}$.

Performance et Scalabilité

L'**avantage** principal d'une telle distribution est la **répartition** de la charge : si chaque nœud gère $|\mathcal{V}_p| \approx \frac{n}{m}$ entités, alors la sous-matrice interne $\omega^{(p)}$ représente $O\left(\left(\frac{n}{m}\right)^2\right)$ liens potentiels, ce qui peut devenir **faisable** si m est assez grand pour que $\frac{n}{m}$ demeure raisonnable. Cela permet de traiter un SCN global dont n serait impossible à stocker ou à traiter en un seul bloc mémoire (de l'ordre de $10^5 \sim 10^7$). Les **communications** inter-blocs conservent toutefois un coût non négligeable, que l'on essaie de contrôler (rafraîchissement épisodique, ignorance des synergies trop faibles, compression...).

L'**inconvénient** réside dans la difficulté à assurer la **cohérence** globale : si un agent $i \in \mathcal{V}_p$ veut entretenir une relation forte avec un agent $j \in \mathcal{V}_q$, cela requiert que SCN_p et SCN_q échangent suffisamment (calcul de la synergie, mise à jour conjointe de $\omega_{i,j}$). En mode asynchrone, on accumule d'éventuels retards rendant la convergence mathématique plus délicate. Sur le plan **pratique**, cette distribution reste souvent la seule solution lorsque $O(n^2)$ devient hors de portée sur un unique nœud.

Conclusion (B. Organisation logicielle d'une architecture distribuée)

Pour bâtir un **SCN** distribué, on déploie plusieurs **nœuds**, chacun contenant un **sous-SCN** local (matrice $\omega^{(p)}$, boucle DSL, modules adjoints). Les liens inter-blocs (entre entités se trouvant dans des partitions distinctes) sont gérés via des communications ponctuelles ou régulières, dans un cadre synchrone (barrière à chaque round) ou asynchrone (requêtes à la demande). Cette organisation autorise un **partage** de la mémoire et du calcul, accroissant la **scalabilité** et permettant de gérer un nombre massif d'entités. Elle implique toutefois une **complexité** supplémentaire, tant au niveau **implémentation** (protocole de synchronisation, gestion de la cohérence) qu'au niveau **mathématique** (systèmes dynamiques couplés). Ainsi, l'architecture distribuée prolonge la logique du SCN centralisé vers les **environnements** où la mémoire d'un seul nœud serait trop limitée ou où les entités sont naturellement réparties, tout en préservant, autant que possible, la **cohérence** globale de l'auto-organisation.

C. Scénarios d'Utilisation

Lorsque l'on envisage un **Synergistic Connection Network (SCN)** en mode distribué, plusieurs **scénarios** se présentent où la répartition des entités en différents sous-SCN devient non seulement utile, mais parfois indispensable. Au niveau **mathématique**, tous partagent l'idée d'un **partitionnement** des données ou des agents, chaque sous-SCN gérant localement ses pondérations $\omega^{(p)}$, tandis que les **liens** inter-blocs se manipulent au travers d'échanges. Sur le plan **ingénierie**, cela se traduit par divers contextes d'application (très grand nombre d'entités, systèmes multi-robots, hétérogénéité symbolique/sub-symbolique) nécessitant un **prototype** ou une **infrastructure** distribuée.

1. Très Grand n

Lorsqu'on manipule un **grand** nombre d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$, l'ordre de grandeur de $O(n^2)$ pour la matrice ω peut rendre impossible son traitement et son stockage en une seule machine. Les ressources (mémoire, CPU) sont vite saturées pour des valeurs de n s'élevant à plusieurs centaines de milliers, voire des millions. Dans ce contexte, on décide de **distribuer** l'ensemble des entités sur plusieurs serveurs (ou nœuds), chacun exécutant un **sous-SCN** local, noté SCN_p . Chacun de ces sous-SCN ne prend en charge que $O((n/m)^2)$ liens internes, ce qui peut devenir gérable si m (le nombre de blocs) est choisi de manière judicieuse.

Cette problématique s'inscrit dans la lignée des **algorithmes distribués** pour la **structure** de graphe ou la **clustering** à grande échelle (type “label propagation”, “graph partitioning”, etc.). Un **SCN** distribué se rapproche de ces approches, car on maintient des **pondérations** (ou “labels”/“scores”) localement et on effectue des synchronisations pour converger vers un état global. Au niveau **mathématique**, on conçoit alors que la dynamique $\omega^{(p)}(t+1) = F^{(p)}(\omega^{(p)}(t), \Omega^{\text{inter}}(t))$ se déroule de façon **parallèle** sur chaque nœud, et l'on obtient une **scalabilité** accrue. Les résultats, en termes de convergence, sont plus complexes à analyser que dans un SCN centralisé, mais cette distribution demeure la seule voie dès que n dépasse largement la capacité d'une seule machine.

2. Contexte Multi-Robot ou Multi-Agents

Dans un cadre **multi-robot** (ou multi-agents, plus général), chaque robot peut être perçu comme un “**nœud**” autonome qui gère, localement, un **mini-SCN** représentant ses capteurs, ses états internes, ou les sous-ensembles d'entités qu'il connaît le mieux (en plus de quelques entités externes qu'il héberge symboliquement). Les **interactions** entre robots sont alors modélisées par des **liens** inter-blocs, réels ou potentiels, qui évoluent via messages d'événements ou mesures coopératives.

Sur le plan **mathématique**, on observe alors un **système** de sous-SCN : chaque robot SCN_p exécute la **boucle** DSL pour ses variables locales ($\omega^{(p)}$), et réalise parfois un échange de données (pondérations “intéressantes”, vecteurs de capteurs, etc.) avec ses **voisins**. Les liens inter-robots $\omega_{i \in \mathcal{V}_p, j \in \mathcal{V}_q}$ se mettent à jour moins fréquemment ou selon un protocole asynchrone (messages d'événements). Sur le plan **implémentation**, cela correspond à un ensemble de robots

(ou d'agents virtuels), chacun muni d'un microprocesseur qui fait tourner son **Module Synergie** et sa **Mise à Jour** en local, et qui “**communique**” sur le réseau pour partager l'information relative aux liens inter-blocs.

Une telle **distribution** reflète l'autonomie partielle des robots : chaque entité (robot) est libre de se réorganiser en interne, tout en participant à une **organisation** plus large via la synchronisation inter-blocs. Sur le plan **ingénierie**, on peut ainsi construire des essaims robotiques où l'**intelligence** émerge de la coopération de sous-SCN dispersés, chacun actualisant $\omega_{i,j}$ localement tout en échangeant périodiquement les pondérations concernant des robots “voisins”.

3. Hétérogénéité des Domaines

Un autre scénario met en scène un **SCN** segmenté par **type** ou **domaine** d'entités. Par exemple, un sous-SCN $SCN_{\text{subsymbol}}$ dédié à des entités sub-symboliques (images, embeddings visuels) et un autre sous-SCN SCN_{symbol} consacré à des entités purement symboliques (règles logiques, concepts, ontologies). Ces deux blocs communiquent par l'intermédiaire d'un “**pont**” ou d'entités hybrides, assurant la **fusion** ou la **traduction** entre représentations sub-symboliques et logiques.

D'un point de vue **mathématique**, chaque sous-SCN gère sa dynamique $\omega^{(p)}$ avec une synergie $S^{(p)}$ adaptée (par exemple, un calcul de similarité cosinus pour le bloc sub-symbolique, et une co-information ou une compatibilité sémantique pour le bloc symbolique). Les **entités** hybrides \mathcal{E}_{hyb} se situent à la frontière : elles possèdent deux représentations (une sub-symbolique, une symbolique) ou sont capables de faire correspondre l'une et l'autre. On peut ainsi définir un **coup** de pont $\omega_{i,j}$ lorsque \mathcal{E}_i réside dans le bloc sub-symbolique et \mathcal{E}_j dans le bloc symbolique, la mise à jour de ce lien pouvant dépendre d'un “Module Synergie” spécifique aux transitions visuel ↔ conceptuel.

Sur le plan **ingénierie**, on réalise deux “clusters” logiques de sous-SCN, chacun spécialisé dans un **domaine** (vision vs. sémantique, par exemple). Sur le plan **mathématique**, on obtient un **réseau** distribué où chaque sous-SCN gère un espace de représentation distinct, et où seuls les **entités-pont** facilitent la synchronisation inter-domaine. Cette hétérogénéité induit souvent des **protocoles** de communication adaptés (ex. on transfère un vecteur embedding quand on veut calculer une synergie cross-domaine). Cela permet au **SCN** global d'intégrer des données multiples (images, texte, logiques de règles) sans forcer un format unique.

Conclusion (C. Scénarios d'Utilisation)

Les **architectures distribuées** pour un SCN couvrent diverses situations : on peut les employer lorsque le **nombre** d'entités (n) dépasse largement la mémoire d'une seule machine, ou quand on opère dans un **contexte multi-robot** ou multi-agents, chaque agent local hébergeant un “sous-SCN” et communiquant pour les liens inter-blocs. Il est aussi possible de segmenter un SCN selon les **domaines** (sub-symbolique vs. symbolique) et de relier ces blocs via des **ponts** ou des entités hybrides. Dans tous ces cas, on préserve la philosophie du DSL (mise à jour de ω , synergie, inhibition éventuelle) tout en déportant une partie des calculs et des structures de données, pour **répondre** aux enjeux de **scalabilité**, de **distribution** géographique ou d'**hétérogénéité** des entités. Sur le plan **mathématique**, cela se traduit par un système dynamique **partiellement** couplé, où chaque bloc évolue localement, et où les **échanges** inter-blocs font émerger (ou non) une cohérence à l'échelle globale.

D. Points Mathématiques et Ingénierie Avancés

Lorsqu'un **Synergistic Connection Network (SCN)** est déployé de manière distribuée, la gestion de plusieurs sous-blocs $\{\omega^{(p)}\}$ et la coordination inter-blocs soulèvent des **questions avancées** à la fois sur le plan mathématique (convergence, synchronisation) et sur le plan d'ingénierie (implémentation de services, répartition des données, etc.). Le comportement global d'un tel SCN résulte alors d'un **système** d'équations ou d'itérations couplées, tandis que chaque sous-SCN local SCN_p maintient ses entités \mathcal{V}_p et ses pondérations internes $\omega^{(p)}$.

Convergence d'un SCN Distribué

Sur le plan purement **mathématique**, la convergence d'un SCN distribué peut s'analyser comme un **système dynamique** multi-blocs. Chaque bloc $p \in \{1, \dots, m\}$ met à jour $\omega^{(p)}$ suivant une équation du type

$$\omega^{(p)}(t+1) = F^{(p)}\left(\omega^{(p)}(t), \Omega^{\text{inter}}(t)\right),$$

où $\Omega^{\text{inter}}(t)$ regroupe les informations relatives aux entités ou pondérations situées dans d'autres blocs. Dans une **forme synchrone**, on suppose que tous les blocs procèdent par “rounds” : chaque SCN_p actualise ses liens internes $\omega^{(p)}$ en fonction de synergies $\mathbf{S}^{(p)}$ et de données inter-blocs reçues à la fin du round précédent. On parle alors d'une “barrière” ou d'un “rendez-vous” qui garantit la cohérence temporelle à chaque itération.

Lorsqu'on passe à une **forme asynchrone** (souvent dite “gossip” ou “push-pull”), chaque sous-SCN peut déclencher sa mise à jour $\omega^{(p)}(t+1)$ quand bon lui semble, éventuellement en s'appuyant sur des valeurs de $\omega^{(q)}$ (pour $q \neq p$) reçues plus tôt. Le système s'apparente alors à un **système dynamique non autonome**, où la convergence se détermine par l'existence ou non d'une topologie d'échanges suffisamment riche et d'une fréquence de mise à jour inter-blocs assez élevée pour éviter la divergence. Sur le plan théorique, on retrouve des analyses comparables aux algorithmes de type “consensus” ou “gossip” dans les réseaux multi-agents, qui établissent parfois des résultats de convergence (avec ou sans retards) si la matrice d'adjacence globale satisfait certaines conditions de connexité ou de régularité.

Une **difficulté** réside dans le fait qu'un SCN distribué introduit des non-linéarités (inhibition, saturation, etc.), rendant la démonstration de convergence plus ardue qu'un simple consensus linéaire. On peut devoir imposer des **contraintes** (nombre minimal d'échanges par période, borne sur le retard asynchrone, etc.) pour aboutir à un état final stable ou quasi-stable. Les ingénieurs s'arrangent souvent pour trouver un compromis : on exige par exemple que chaque bloc synchronise (même partiellement) ses liens inter-blocs au moins toutes les K itérations, ou on restreint le SCN aux seuls liens inter-blocs au-dessus d'un certain seuil de similarité.

Implémentation et Organisation des Services

Sur le plan de l'**ingénierie**, une architecture distribuée se concrétise par plusieurs “**services**” ou “**modules**” s'exécutant sur des machines distinctes. Chacun possède son **sous-SCN** local, autrement dit une version de la “Core” (voir §5.2.1.1) et éventuellement des modules “Synergie”, “Inhibition”, etc. propres à son sous-ensemble \mathcal{V}_p . Les communications inter-blocs, indispensables pour maintenir la cohérence, s'implémentent via :

- 267. **Échanges explicites** (RPC, REST, gRPC) : quand un nœud SCN_p veut actualiser $\omega_{i,j}$ pour $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$, il interroge un service sur SCN_q pour obtenir la représentation de \mathcal{E}_j ou la pondération précédente.
- 268. **Synchronisation par rounds** : si la mise à jour est synchrone, on attend la fin d'un “tour” local, puis on échange un “lot” de pondérations inter-blocs.
- 269. **Structures de cache** : parfois, on stocke (en local) une approximation de $\omega_{i,j}$ inter-bloc, actualisée périodiquement, afin de ne pas surcharger le réseau à chaque requête.

Côté **données**, on peut répartir les entités de façon relativement égale pour équilibrer la charge, ou bien on peut adopter un partitionnement sémantique (les entités “visuelles” sur un cluster, les entités “textuelles” sur un autre). Dans un déploiement massif, chaque bloc local peut gérer $\sim 10^4$ ou $\sim 10^5$ entités, permettant de maintenir $(\frac{n}{m})^2$ liaisons internes dans la mémoire d'une seule machine. Les liens inter-blocs $\omega_{i \in \mathcal{V}_p, j \in \mathcal{V}_q}$ se gèrent soit par un stockage partiel (p. ex. un dictionnaire $(i, j) \mapsto \omega_{i,j}$ si $\omega_{i,j}$ est $>$ à un certain seuil), soit par un “recalcul” sur demande en recourant au “Module Synergie” distant.

Applications de l'Architecture Distribuée

Les **applications** d'un SCN distribué se déclinent en plusieurs domaines :

Dans un **Cloud** ou cluster HPC, on veut analyser un énorme graphe ou un volume considérable d'entités. On distribue alors les entités en sous-SCN, chaque sous-SCN tournant sur un ensemble de machines. Les mises à jour itératives de ω ou de $\omega^{(p)}$ progressent en parallèle, synchronisées de temps à autre. Cette configuration répond à un besoin de **scalabilité** : impossible de charger $O(10^{12})$ liaisons sur un unique serveur.

Dans un **environnement multi-robot** (flottille de drones, essaim de robots de service), chaque robot exécute localement un mini-SCN sur ses propres capteurs, ses propres représentations, et discute via messages d'événements ou protocoles ad hoc avec les autres. Les liens inter-robots $\omega_{i \in \mathcal{V}_p, j \in \mathcal{V}_q}$ s'actualisent au fil d'interactions tangibles

(rencontre physique, échange de données). De la sorte, le système complet forme un **SCN** global, potentiellement asynchrone, reflétant la coopération ou la co-information entre robots.

Dans un **contexte Edge computing**, on peut avoir des microcontrôleurs (capteurs, devices IoT) gérant en local un sous-groupe d'entités, puis communiquant avec un "hub" de plus haut niveau. Chaque microcontrôleur agit comme un "petit SCN" local, alors que le hub fait office de "meta-SCN" ou de "SCN central," agrégeant les informations inter-blocs.

Sur le plan **mathématique**, le fonctionnement distribué n'altère pas la logique fondamentale du DSL, mais la décline sous forme d'**itérations couplées**. Sur le plan **implémentation**, on construit un écosystème de **services** (chaque sous-SCN local + un ou plusieurs services de synchronisation) permettant l'apprentissage distribué.

Conclusion (D. Points Mathématiques et Ingénierie Avancés)

Dans une architecture distribuée pour le SCN, les **problématiques** de convergence exigent de considérer un **système** d'équations couplées, que l'on traite de manière synchrone (barrières, rounds) ou asynchrone (approches gossip). Sur le plan **ingénierie**, on met en place des **services** (un sous-SCN local par nœud, un mécanisme de communication inter-blocs) afin de gérer la répartition des entités et la synchronisation nécessaire. Les applications s'avèrent multiples : analyse de grands graphes en **cloud**, systèmes **multi-robots**, configurations **edge** hétérogènes. On apprécie ainsi la **flexibilité** du DSL à se déployer en mode distribué, moyennant des échanges partiels ou complets, et à s'adapter aux différentes topologies de communication. On fait toutefois face à des **défis** supplémentaires : complexité d'implémentation, incohérence passagère en mode asynchrone, et besoin d'un protocole pour gérer les liens inter-blocs. Sur le plan théorique, il reste souvent à **prouver** ou **assurer** la convergence (ou la quasi-convergence) sous certaines hypothèses de connectivité globale, de retard limité, et de fréquence suffisante d'échanges entre sous-SCN.

Conclusion (5.2.3.3)

L'**architecture distribuée** du SCN, fondée sur plusieurs **sous-SCN** coopérant, constitue la **solution** adéquate lorsqu'on atteint de grandes **dimensions** (n énorme), des **contraintes** de localisation (robots dispersés), ou un **besoin** de robustesse (chaque sous-SCN continue si un autre tombe). Sur le plan **mathématique**, on traite alors un système d'équations ou de mises à jour **couplées**, parfois avec synchronisations ponctuelles. Sur le plan **ingénierie**, cela se concrétise par :

- Des **modules** "sous-SCN" déployés sur différents nœuds (serveurs, robots),
- Des **protocoles** de communication (synchronisation, échanges de pondérations ou résumés),
- Une **possible** approche "meta-SCN" au-dessus, gérant les interactions inter-blocs.

Cette architecture s'avère plus **complexe** à mettre en place que les approches **mono-module** (5.2.3.1) ou **modulaire** centralisée (5.2.3.2), mais c'est la **clé** pour la **scalabilité extrême** et l'**adaptation** à des environnements distribués ou massivement parallèles.

5.3. Structures de Données pour la Matrice (\omega)

5.3.1. Dense vs. Sparse

- 5.3.1.1. Avantages/Inconvénients d'une **matrice dense** (accès direct, simple) vs. **liste d'adjacence** si la plupart des (\omega) sont faibles.
- 5.3.1.2. Critères de choix : taille (n), proportion de liaisons fortes.

5.3.2. Indexation et Accès Rapide

- 5.3.2.1. **Hashmap** ((i,j) \mapsto \omega_{i,j}).
- 5.3.2.2. Organisation en "voisinages" (garder les (k) plus forts liens).

5.3.3. Synchronisation en Cas de Distribution

- 5.3.3.1. Si la matrice (\omega) est répartie sur plusieurs nœuds : protocole de communication (verrous, versioning).
- 5.3.3.2. Approche asynchrone vs. synchrone pour la mise à jour des liens inter-sous-SCN.

5.3. Structures de Données pour la Matrice ω

Dans un **SCN**, la matrice ω (de dimension $n \times n$) stocke les **pondérations** $\omega_{i,j}$ représentant la force du lien (ou la synergie actualisée) entre chaque entité \mathcal{E}_i et \mathcal{E}_j . Selon la **taille** n et la **densité** réelle des liens (beaucoup ou peu de valeurs non nulles), on choisira une structure de données appropriée (dense, sparse, etc.). Il s'agit d'un choix crucial, car il conditionne à la fois les **performances** (temps de mise à jour, accès mémoire) et la **flexibilité** (ajout/suppression de liens, extension à un mode distribué).

5.3.1. Dense vs. Sparse

Lorsqu'on parle de “**matrice dense**” vs. “**liste d’adjacence**” ou “**structuration sparse**”, on se réfère à des stratégies différentes pour stocker la grille $\omega_{i,j}$. Il en découle des **avantages** et **inconvénients** mathématiques et pratiques.

5.3.1.1. Avantages/Inconvénients d'une Matrice Dense (Accès Direct, Simple) vs. Liste d'Adjacence si la Plupart des ω sont Faibles

Dans le contexte d'un **Synergistic Connection Network (SCN)**, la **représentation** des pondérations $\omega_{i,j}$ devient cruciale lorsque le nombre d'entités n augmente. L'important défi consiste à choisir la structure de stockage la plus adaptée pour ces liaisons, notamment en considérant si la majorité des valeurs $\omega_{i,j}$ sont faibles ou nulles en raison de mécanismes d'**inhibition** ou de régularisation. Deux approches se distinguent dans ce cadre, l'**utilisation d'une matrice dense** et celle d'une **structure de stockage sparse**, telle qu'une **liste d'adjacence** ou un dictionnaire.

A. Matrice Dense

La représentation par **matrice dense** consiste à allouer un tableau bidimensionnel $W[n][n]$ dans lequel chaque élément $\omega_{i,j}$ est stocké de manière explicite à l'indice (i, j) . Chaque entité i correspond ainsi à une ligne complète de ce tableau, permettant un **accès direct** aux pondérations avec une complexité en temps de $O(1)$ pour toute lecture ou écriture. Par exemple, la somme des liens sortants d'une entité i se calcule rapidement par

$$\sum_{j=1}^n \omega_{i,j},$$

ce qui peut être optimisé par des techniques de vectorisation, telles que les instructions **SIMD**, ou en utilisant des bibliothèques spécialisées telles que **BLAS**.

Néanmoins, cette approche présente un **inconvénient majeur** : elle est extrêmement **consomatrice de mémoire**. En effet, le stockage d'une matrice dense requiert $O(n^2)$ éléments, ce qui peut rapidement conduire à une utilisation excessive de la mémoire lorsque n est grand (par exemple, quelques dizaines de milliers d'entités entraînant un espace de stockage de plusieurs gigaoctets). De plus, si une grande partie des pondérations reste négligeable (valeurs proches de zéro en raison d'un mécanisme d'inhibition), l'allocation mémoire se trouve en partie gaspillée pour stocker ces zéros, et le parcours complet de la matrice, également en $O(n^2)$, peut s'avérer prohibitif pour des mises à jour fréquentes.

B. Liste d'Adjacence ou Stockage Sparse

À l'opposé, le **stockage sparse** ne retient que les paires (i, j) pour lesquelles $\omega_{i,j}$ atteint une valeur significative, par exemple, supérieure à un certain seuil ou figurant dans le « top k » des liaisons pour chaque entité i . Plusieurs formats existent pour implémenter cette approche, notamment le format **CSR (Compressed Sparse Row)**, qui regroupe les liaisons non nulles par ligne, ou encore l'utilisation de dictionnaires associant chaque couple (i, j) à sa valeur correspondante $\omega_{i,j}$.

L'avantage principal d'une telle approche réside dans une **réduction substantielle** de l'utilisation mémoire. Si la densité effective des liaisons est faible, on peut passer d'une complexité en $O(n^2)$ à une complexité en $O(\rho n^2)$ où

$\rho \ll 1$ représente le taux de densité des liaisons significatives. Cette économie de mémoire est particulièrement avantageuse lorsque seules quelques connexions par entité sont réellement non négligeables. Par ailleurs, le parcours des voisins d'une entité i se limite aux entrées présentes dans la liste d'adjacence, ce qui peut considérablement accélérer les algorithmes qui ne nécessitent pas de balayer l'ensemble des n éléments.

Cependant, le **stockage sparse** comporte aussi des inconvénients. L'accès direct à une pondération spécifique $\omega_{i,j}$ n'est plus garanti en $O(1)$ dans tous les cas, en particulier lorsque l'implémentation repose sur une structure de type **hashmap** qui, bien que visant une complexité moyenne de $O(1)$, peut souffrir de surcoûts liés aux collisions ou au ré-hachage. Par ailleurs, lorsqu'on utilise une liste d'adjacence, l'accès à une pondération donnée dépend du nombre de voisins d_i de l'entité i , ce qui introduit une complexité en $O(d_i)$. De plus, la gestion dynamique de ces structures (insertion, suppression ou mise à jour) peut nécessiter des mécanismes supplémentaires, tels que des min-heaps pour conserver les k plus grandes liaisons, ce qui complexifie l'implémentation.

Conclusion (5.3.1.1)

Le choix entre l'utilisation d'une **matrice dense** et d'une **représentation sparse** doit être guidé par le **contexte** d'application. Lorsque n est de taille modérée et que la densité des liaisons fortes est élevée, la simplicité d'accès et l'efficacité d'une matrice dense – avec un accès direct en $O(1)$ – constituent des atouts indéniables. À l'inverse, pour des réseaux de grande taille (avec n atteignant voire dépassant 10^5) où la majorité des valeurs $\omega_{i,j}$ sont faibles ou nulles, l'utilisation d'un **stockage sparse** via liste d'adjacence ou dictionnaire devient prépondérante, permettant d'économiser considérablement de la mémoire et de focaliser les calculs sur les liens significatifs.

D'un point de vue mathématique, la matrice dense représente la solution la plus simple et directe pour accéder aux éléments $\omega_{i,j}$, tandis que la liste d'adjacence offre une efficacité en mémoire et en temps lorsque le réseau se caractérise par une faible densité effective, souvent induite par des mécanismes d'inhibition ou de régularisation. Le choix final doit également prendre en compte d'autres considérations telles que la facilité de persistance des données sur disque ou les exigences spécifiques du traitement des mises à jour. Il est ainsi recommandé que l'**architecture** décrite dans la suite du Chapitre 5 offre la flexibilité nécessaire pour basculer entre ces deux modes de stockage en fonction des besoins sans impacter significativement les modules responsables de la dynamique ou de la synergie.

Ce point de convergence entre **représentation dense** et **représentation sparse** constitue un aspect fondamental de la conception des SCN, et il sera examiné en profondeur dans les chapitres ultérieurs, notamment dans **Chapitre 5**, qui traitera de l'**architecture** générale du SCN, ainsi que dans les sections connexes qui discuteront de la **dynamique** d'auto-organisation et des stratégies d'optimisation des liaisons $\omega_{i,j}$.

5.3.1.2. Critères de Choix : Taille n , Proportion de Liaisons Fortes

Les réflexions portant sur la structure la plus adaptée pour stocker la matrice ω dans un **Synergistic Connection Network (SCN)** s'articulent principalement autour de deux **critères** : la **taille** du réseau (n entités) et la **densité** réelle des liaisons “fortes” (c'est-à-dire la proportion de valeurs $\omega_{i,j}$ restant significatives). Si la taille n demeure modeste ou si, au contraire, la synergie et l'inhibition conduisent à un fort pourcentage de liens non nuls, la **matrice dense** pourra s'avérer plus simple et avantageuse. En revanche, pour un très grand n ou un SCN où l'inhibition et la compétition rendent la plupart des liaisons négligeables, une **représentation sparse** (liste d'adjacence, dictionnaire) permettra d'économiser massivement la mémoire et d'accélérer certaines opérations.

A. Taille n

Lorsque le nombre d'entités n demeure **relativement petit** (typiquement jusqu'à quelques milliers, ou quelques dizaines de milliers), la mise en place d'une **matrice dense** de taille $O(n^2)$ reste tout à fait envisageable. Les mémoires contemporaines peuvent gérer sans difficulté quelques millions d'entrées, et un parcours complet en $O(n^2)$ par itération n'est pas nécessairement prohibitif si on peut paralléliser un peu le calcul (par exemple avec un *double-buffer* synchrone et quelques threads). Dans ce cas, la simplicité d'accès $\omega_{i,j}$ en $O(1)$ justifie largement l'emploi d'une matrice dense si la densité de liens s'avère élevée.

En revanche, si n franchit un seuil critique (quelques dizaines de milliers à plusieurs centaines de milliers), la structure dense devient difficile à stocker (les $O(n^2)$ éléments peuvent représenter des centaines de Go) et le coût de parcours complet en $O(n^2)$ à chaque étape peut s'avérer intenable. Dans un tel scénario, on tentera de **sparsifier** la matrice en ne préservant que quelques liens par entité (typiquement les k liaisons les plus fortes), ce qui rapproche la matrice d'une structure "liste d'adjacence" de taille $O(kn)$.

B. Proportion de Liaisons Fortes (Densité)

On note ρ la proportion de liens "non négligeables" dans le réseau, qu'on peut approximativement évaluer après inhibition ou après un filtrage des liens très faibles. Si ρ est proche de 1, cela signifie que la **plupart** des liaisons $\omega_{i,j}$ sont non nulles et qu'on a un réseau plutôt dense. Une **matrice dense** sera alors cohérente et évitera des surcoûts en recherche ou mise à jour. En revanche, si $\rho \ll 1$, c'est-à-dire que la quasi-totalité de $\omega_{i,j}$ tombent à des valeurs quasi nulles, un **format sparse** (liste d'adjacence, dictionnaire) se révèle bien plus économique en mémoire et plus efficace à parcourir si l'on ne souhaite traiter que les liaisons fortes.

C. Couplage des Deux Critères

Sur le plan **mathématique**, la taille mémoire d'une **matrice dense** est $O(n^2)$, tandis que la taille mémoire d'une **structure sparse** se déduit de $O(\rho n^2)$. Lorsque $\rho \approx 0.001$ (soit 0.1 % de densité), la structure sparse n'occupe qu'un millième de l'espace nécessaire à la dense. À l'inverse, dans un contexte où la densité réelle est $\rho \approx 1$ (ou 50 %), on retombe pratiquement à $O(n^2)$ d'entrées, ce qui enlève l'intérêt d'une représentation "sparse". Les accès aux éléments $\omega_{i,j}$ y deviennent plus complexes (listes, dictionnaires), sans gain notable en mémoire.

La **mise à jour** d'un SCN exige souvent de parcourir toutes les valeurs $\omega_{i,j}$ ou, au moins, toutes les liaisons fortes. Si le réseau demeure *très dense*, ce parcours pourrait rester $O(n^2)$. Dans un format sparse, un parcours complet demande $O(\rho n^2)$, ce qui est avantageux ssi ρ est faible. Pour l'accès "aléatoire" à $\omega_{i,j}$, une matrice dense offre un $O(1)$ direct, alors qu'une structure sparse génère un $O(\log d_i)$ ou $O(d_i)$ (selon les structures) pour retrouver un lien dans la liste d'adjacence.

Conclusion (5.3.1.2)

Le **choix entre matrice dense et liste d'adjacence** (ou stockage sparse) doit s'appuyer sur :

La taille n . En-dessous de quelques milliers à dizaines de milliers, une structure dense peut suffire et simplifier l'implémentation. Au-delà, c'est souvent impraticable d'avoir $O(n^2)$ entrées.

La densité ρ . Si la plupart des liaisons $\omega_{i,j}$ restent ténues ou nulles (après inhibition ou top- k forcé), une structure sparse sauvera la mémoire et le temps de calcul en ne stockant que les liens significatifs.

Les schémas d'opérations récurrents : un besoin fréquent d'accès aléatoire rapide pointe vers un format dense, un besoin de manipuler seulement les "voisins forts" milite en faveur d'une représentation sparse.

Dans un SCN en pratique, il n'est pas rare qu'au début les données soient manipulées de manière dense, puis qu'on introduise une "sparsification" au fil des itérations, conduisant à une taille $O(kn)$. Certains frameworks permettent même de jongler entre représentations selon l'état courant (chap. 5.3.2), assurant ainsi un usage optimal des ressources et l'adaptabilité du SCN.

5.3.2. Indexation et Accès Rapide

Même après avoir choisi un **format général** (dense vs. sparse) pour la matrice ω (cf. 5.3.1), on peut affiner l'**organisation** interne afin de faciliter les **accès** (lecture/écriture) aux pondérations $\omega_{i,j}$. Dans de nombreux cas, on a besoin d'opérations comme :

- `getWeight(i, j)` : accéder rapidement à la pondération d'un lien particulier,
- `sumOfWeights(i)` : sommer les liens d'une entité \mathcal{E}_i (pour l'inhibition ou la normalisation),

- $\text{topLinks}(i, k)$: retrouver les k plus gros liens sortants de i .

Selon la **densité** et la **nature** de la mise à jour (compétition locale, etc.), on peut adopter des **structures** plus sophistiquées, par exemple un **hashmap** associant $(i, j) \mapsto \omega_{i,j}$ (5.3.2.1) ou une **organisation** en “voisinages” (5.3.2.2).

5.3.2.1. Hashmap $(i, j) \mapsto \omega_{i,j}$

L'une des structures de données *sparse* envisageables pour stocker la matrice ω d'un **Synergistic Connection Network (SCN)** consiste à employer une **table de hachage** (ou *hashmap*) dont la clé est le couple (i, j) et la valeur est la pondération $\omega_{i,j}$. Cette représentation diffère d'un tableau dense ou d'une liste d'adjacence classique, puisqu'on ne stocke que les paires (i, j) jugées réellement significatives, tout en profitant, en moyenne, d'un accès $O(1)$ pour lire ou écrire une entrée dès lors que le hachage est bien conçu.

A. Principe

La matrice ω n'est plus représentée comme un bloc mémoire rectangulaire $W[n][n]$, ni comme un ensemble de listes par lignes, mais comme un **dictionnaire** (ou *hashmap*) associant toute paire $(i, j) \in \{1, \dots, n\}^2$ à la valeur $\omega_{i,j}$. Dans un langage comme C++, on peut employer `std::unordered_map<std::pair<int,int>, double>` ; en Python, un simple *dict* avec pour clé un tuple (i, j) . Les liens dont la pondération $\omega_{i,j}$ demeure faible (inférieure à un certain seuil) peuvent être omis du dictionnaire, réduisant ainsi la taille mémorielle lorsque la densité du SCN est faible.

Le système devient

$$H: (i, j) \mapsto \omega_{i,j}$$

où seules les paires (i, j) jugées “actives” (i.e. $\omega_{i,j}$ non négligeable) apparaissent dans la table. Cela diffère d'un **stockage dense**, qui crée $O(n^2)$ emplacements, et d'une **liste d'adjacence** structurée par ligne, qui assigne explicitement à chaque entité i sa liste de voisins.

B. Avantages

Un **avantage clé** de cette approche réside dans l'**accès direct** à $\omega_{i,j}$ en temps moyen $O(1)$. En effet, la **table de hachage** permet de retrouver la valeur associée à la clé (i, j) via un calcul de hachage, sans devoir chercher dans une liste d'adjacence. Cette propriété diffère notablement d'une liste classique, où l'on itère sur tous les voisins de i pour trouver si j y est présent.

De plus, la **suppression** ou l'**insertion** d'un lien $\omega_{i,j}$ (notamment en cas de mise à zéro ou de réactivation d'un lien suite à un changement de synergie) se fait également en $O(1)$ amorti. On bénéficie donc d'une grande **flexibilité** pour gérer les liens en dynamique : la représentation ne dépend pas de structures chaînées ni de tris partiels, et la table de hachage sait s'ajuster relativement aisément à l'ajout/suppression d'entrées.

Enfin, s'il n'existe qu'un nombre réduit de paires (i, j) réellement non nulles (faible densité ρ), la **mémoire** employée se limite à $O(\rho n^2)$. Cela devient un atout majeur dans les scénarios où l'inhibition ou les mécanismes de **sparsification** maintiennent la plupart des pondérations $\omega_{i,j}$ au seuil de zéro, rendant vaine une structure dense.

C. Inconvénients

Une **table de hachage** n'est pas sans **coût caché**. Le temps moyen d'accès en $O(1)$ s'accompagne d'une constante parfois significative, surtout si la structure devient très large (des millions de paires). Les “*re-hashings*” se déclenchent à mesure que le dictionnaire grossit, occasionnant des *slowdowns* sporadiques. Par ailleurs, contrairement à un tableau 2D, il n'existe pas de *layout* mémoire contigu pouvant tirer parti d'instructions vectorielles (SIMD) ou d'algèbre linéaire BLAS.

Un second souci touche au **parcours** par entité i . Dans une liste d'adjacence structurée, on dispose directement d'un vecteur listant les voisins de i . Dans un dictionnaire global $(i, j) \mapsto \omega_{i,j}$, on ne dispose pas en standard d'une répartition

par *buckets* regroupés. Parcourir l'ensemble des liens de i exige alors soit un balayage complet de la table (avec un filtrage sur la clé), soit le maintien en parallèle d'une structure secondaire (ex. un “adjIndex[i]” pointant vers une liste de clés (i, j)). Cela ajoute de la **complexité** de maintenance.

Enfin, si la densité ρ n'est pas si faible ou qu'elle évolue vers un SCN plutôt dense, la table de hachage devient lourde à manipuler. On perd alors l'avantage d'avoir éliminé $O(n^2)$ emplacements, et on hérite des surcoûts en dispersion de clés, en collisions, etc.

D. Analyse Mathématique et Cas d'Usage

On suppose que ρn^2 paires (i, j) soient actives. Le **coût mémoire** du dictionnaire sera $O(\rho n^2)$, plus un overhead lié aux structures internes de hash. En **accès direct**, on a un temps moyen $O(1)$: on exécute $\omega_{i,j} = H[(i, j)]$. Cette propriété rend la hashmap attractive pour les scénarios où la dynamique du SCN réalise souvent des “random access” sur des paires (i, j) . Toutefois, le “parcours complet” des liens sortants d'une entité i ne s'obtient pas nativement en $O(d_i)$, à moins de maintenir une structure secondaire ou un indice.

Les **contextes** où cette structure se défend bien sont ceux où la densité $\rho \ll 1$ (rendant la matrice dense peu rentable) et où l'on multiplie les accès ponctuels $\omega_{i,j}$ dans le code de mise à jour, tout en ajoutant ou supprimant régulièrement des liens au fil de l'évolution. De plus, l'ajout/suppression d'un couple (i, j) reste simple (opération *insert* ou *erase* en $O(1)$ amorti), contrairement à certaines implémentations de liste d'adjacence devant maintenir un tri par ordre de poids.

Conclusion

L'usage d'une **hashmap** $(i, j) \mapsto \omega_{i,j}$ constitue une **approche sparse** flexible, particulièrement utile pour un SCN si l'on pratique un *random access* fréquent sur de multiples paires, que l'on modifie la liste de liens actifs de manière dynamique, et que la densité demeure relativement faible. Les principaux **bénéfices** sont l'accès (moyennement) direct en temps amorti $O(1)$ et la liberté d'ajouter/supprimer des entrées. Les **limites** concernent le **parcours** par entité i (absence de structure locale) et la surcharge des mécanismes de hachage lorsque le nombre de paires (i, j) devient très grand. Dans une **architecture** modulaire (chap. 5), il est crucial de tenir compte de ces avantages et inconvénients au moment de sélectionner le format de stockage, tout en considérant la question de la densité, des types d'opérations dominantes (accès aléatoire vs. parcours par noeud), et de l'évolution potentielle du réseau au cours du temps.

5.3.2.2. Organisation en “Voisinages” (Garder les k plus forts liens)

Il est souvent observé dans un **Synergistic Connection Network** (SCN) que, pour chaque entité \mathcal{E}_i , seul un nombre relativement restreint de liaisons $\omega_{i,j}$ conservent une valeur suffisamment élevée au fil de la dynamique (additive, multiplicative, etc.). Dans ce cas, la **matrice** ω peut être considérablement épurée en se limitant, pour chaque i , aux k liens les plus forts sortants. Ce principe, parfois décrit sous l'appellation “ **k plus forts liens**” ou “**voisinage restreint**”, se rattache à l'idée d'une **sparsification** contrôlée : on impose par construction que chaque entité ne conserve qu'un nombre limité de connexions, ce qui réduit la complexité mémoire et facilite la mise à jour.

A. Principe : Conserver un Voisinage Restreint par Entité

Un SCN défini via $\omega_{i,j}(t)$ peut atteindre au plus $O(n^2)$ liens, ce qui devient impraticable si n s'accroît. L'**organisation en voisinage** pose alors la règle :

$$\forall i, \quad \text{voisins}(i) \subseteq \{1, 2, \dots, n\}, \quad |\text{voisins}(i)| \leq k,$$

où k représente un nombre fixé (ou un paramètre qui peut varier selon l'entité). Pour chaque i , on ne conserve que les $\omega_{i,j}$ les plus grands, éliminant les liens trop faibles. Cette notion s'applique de manière dynamique : si une liaison $\omega_{i,j}$ “surpasse” un des liens actuels au fil des itérations, elle peut intégrer le voisinage de i , tandis que le plus faible lien en sort.

B. Aspects Mathématiques et Logiques

La **dynamique** $\omega_{i,j}(t+1)$ continue de suivre les formules usuelles (par exemple additive), mais un “**opérateur**” TopK_i est appliqué périodiquement (ou à chaque itération) : on trie (ou sélectionne) les liens sortants de i en fonction de $\omega_{i,j}(t+1)$ et on coupe (met à zéro ou ignore) ceux qui ne figurent pas dans les k plus forts. Mathématiquement, cela revient à :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}'(t+1) & \text{si } j \in \text{TopK}_i(\omega_{i,\cdot}'(t+1)), \\ 0 & \text{sinon.} \end{cases}$$

où $\omega_{i,j}'(t+1)$ est la mise à jour brute (sans coupes), puis on applique la *sélection* TopK_i . L’effet combiné se rapproche d’une **inhibition latérale** (les liaisons se “battent” pour figurer dans le top-k). Si k est raisonnable (typiquement un ordre de grandeur bien inférieur à n), la structure finale est fortement *sparse*, ce qui diminue le temps de parcours nécessaire pour des opérations (inhibition globale, calcul partiel, etc.).

C. Structure de Données pour le Voisinage

Afin de maintenir ce **voisinage** de taille $\leq k$ pour chaque entité i , on peut employer plusieurs approches :

Un **heap** de taille k conservant les liens sortants de i . Dès qu’un nouveau lien $\omega_{i,j}$ devient plus grand que le minimum du heap, on insère $(j, \omega_{i,j})$ et éjecte l’élément le plus faible.

Une **liste** maintenue triée, bien que les insertions/suppressions puissent devenir plus coûteuses ($O(k)$).

Un **arbre** équilibré (par ex. un *balanced tree*) de taille $\leq k$, permettant insertions et suppressions en $O(\log k)$.

Dans tous ces cas, la mise à jour du top- k pour $\omega_{i,j}(t)$ s’effectue en $O(\log k)$ lorsqu’on envisage d’ajouter ou de remplacer un lien. Ainsi, si l’on effectue un *random access* sur $\omega_{i,j}$ en dehors du top- k , on le considère nul (ou inexistant). Les liens $\omega_{i,j}$ contenus dans le voisinage sont ceux activement mis à jour et considérés comme “vivants”.

D. Bilan sur la Complexité

La **taille mémoire** globale devient $O(k n)$ à la place de $O(n^2)$. Ceci représente un gain appréciable dès lors que $k \ll n$. À chaque itération, chaque entité i manipule $O(k)$ liens dans son voisinage, et la mise à jour (notamment si on parcourt la liste des voisins pour calculer la somme ou vérifier l’inhibition) s’effectue en $O(k)$. Au besoin, si l’on examine la possibilité de faire “entrer” un nouveau lien, on réalise un insertion-suppression dans la structure $O(\log k)$. Ce schéma devient bien plus soutenable qu’un parcours en $O(n)$ ou $O(n^2)$.

E. Considérations pour la Qualité de l’Auto-Organisation

Le fait de ne maintenir que k liens par entité introduit une forme de **compétition** pour les connexions. Cela peut encourager une meilleure différenciation des clusters et limiter la prolifération de liaisons moyennement fortes. Néanmoins, il faut veiller à ce que k ne soit pas trop restreint, sous peine de **fragmenter** exagérément le réseau ou d’empêcher des liens potentiellement utiles de croître. Dans certains SCN, on commence par un k généreux et on le réduit progressivement pour forcer l’apparition de liens *vraiment* robustes, traduisant une phase de “cristallisation” des clusters.

Conclusion

L’**organisation** en “voisinages restreints” (garder, pour chaque entité, les k plus forts liens sortants) constitue un **outil** puissant de *sparsification*. Sur le plan **mathématique**, cela revient à insérer un opérateur *TopK* dans la boucle de mise à jour, imposant un maximum de k connexions par entité. Sur le plan **ingénierie**, cela se traduit par un stockage de type liste/heap limitant la taille à k . Les **avantages** sont d’alléger massivement la mémoire et d’accélérer la mise à jour quand $k \ll n$. Les **inconvénients** tiennent à la charge de maintenir ce top- k et au risque de compromettre l’exploration de certains liens qui commencent faibles mais pourraient devenir forts. Malgré tout, dans la pratique, cette stratégie offre un excellent compromis, permettant d’évoluer vers un réseau *sparse* plus aisément gérable, et favorisant l’émergence de clusters plus nets au sein du SCN.

5.3.3. Synchronisation en Cas de Distribution

Dans certains **scénarios** de grande envergure ou de localisation multiple, on ne peut plus stocker la **matrice** ω en un seul endroit : on la **répartit** alors sur plusieurs **nœuds** (machines, serveurs, robots, etc.) (cf. 5.2.3.3). Cela soulève la question de la **synchronisation** : comment veiller à ce que les mises à jour $\omega_{i,j}(t+1)$ effectuées sur un nœud SCN_p soient **compatibles** avec celles effectuées sur un autre SCN_q , notamment pour les *liens* ou les *entités* se situant à la frontière entre les deux sous- SCN ?

5.3.3.1. Si la Matrice ω est Répartie sur Plusieurs Nœuds : Protocole de Communication (Verrous, Versioning)

Lorsqu'un **Synergistic Connection Network (SCN)** doit gérer un très grand nombre d'entités, il arrive fréquemment que la taille totale de la matrice ω , potentiellement de l'ordre de $O(n^2)$, dépasse les capacités d'une machine unique. Une approche naturelle consiste alors à **distribuer** les données et le calcul sur plusieurs nœuds ou serveurs, chacun prenant en charge un **sous-ensemble** des entités $\{\mathcal{E}_i\}$. Il s'agit cependant de veiller à maintenir une **cohérence** satisfaisante du processus de mise à jour, en particulier pour les liaisons *inter-blocs* reliant deux entités gérées par des nœuds distincts. C'est à ce niveau que se pose la question d'un **protocole** de communication et de synchronisation incluant verrous ou versioning, afin d'éviter les lectures/écritures incohérentes et de reproduire au mieux la vision "synchrone" du SCN.

Répartition de la Matrice ω en Blocs

Le schéma usuel consiste à découper l'ensemble des entités $\{1, \dots, n\}$ en plusieurs blocs $\mathcal{V}_1, \dots, \mathcal{V}_m$, de sorte que chaque sous- SCN , noté SCN_p , manipule localement les pondérations $\omega_{i,j}$ pour $i, j \in \mathcal{V}_p$. Lorsque le réseau est réellement distribué, on confie à chaque nœud du cluster la responsabilité de stocker et mettre à jour la portion $\omega^{(p)}$ qui correspond à son bloc. Se pose alors la question des liens sortants *inter-blocs* : une pondération $\omega_{i,j}$ avec $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$ doit être physiquement gérée par un nœud donné, ou éventuellement répartie. Une solution commune est de décider que les liaisons (i, j) se rangent avec l'entité i : si $i \in \mathcal{V}_p$, alors SCN_p stocke toutes les $\omega_{i,j}$, même si j dépend d'un autre nœud. Cette règle de "détenteur unique" évite les doublons, mais contraint le protocole à synchroniser la lecture des données sur les $\omega_{k,j}$ impliquées pour, par exemple, l'inhibition ou d'autres couplages.

Mécanismes de Coordination et de Cohérence

L'enjeu principal est de veiller à ce que la **dynamique** $\omega_{i,j}(t+1) = \dots$ demeure mathématiquement correcte, même si plusieurs nœuds mettent à jour simultanément des portions distinctes de la matrice. Dans un *SCN* uniifié, on postule en général un *pas synchrone* :

$$\omega(t+1) = F(\omega(t)),$$

et on souhaite que l'itération $\omega(t) \mapsto \omega(t+1)$ corresponde à un état global cohérent. Dans un réseau distribué, chaque nœud SCN_p ne détient qu'une partie de $\omega(t)$. Pour cette mise à jour synchrone, on peut imposer un mécanisme de **barrière** : l'état $\omega(t)$ est lu de façon globale et stabilisée, chaque nœud calcule sa portion $\omega^{(p)}(t+1)$ en s'appuyant sur les données inter-blocs, puis un échange de *diffs* ou de trames met à jour les liaisons (i, j) traversant les blocs, enfin tous les nœuds valident l'étape et on obtient un état $\omega(t+1)$ complet. Ce fonctionnement est courant en **MPI** (Message Passing Interface), où le cluster exécute une étape de communication collective après le calcul local. D'un point de vue algorithmique, ce mode "synchrone" assure une cohérence stricte : la mise à jour $\omega(t+1)$ se base intégralement sur $\omega(t)$, et la "vision globale" du temps demeure claire.

Dans certaines infrastructures, un mode *asynchrone* est recherché pour gagner en parallélisme et réduire les coûts de synchronisation. Chaque nœud SCN_p avance alors sa portion $\omega^{(p)}(t+1)$ dès que possible, en récupérant, au fil de l'eau, les versions mises à jour (ou pas) par les autres nœuds. On risque toutefois des phénomènes de lecture/écriture obsolètes, qui altèrent la dynamique. C'est ici qu'intervient la notion de **versioning** : on peut associer un identifiant de "round" ou "version" à l'état $\omega(t)$. Chaque nœud publie sa version ver_p quand il a terminé l'itération t , et ne traite les communications inter-blocs que si elles sont marquées d'une version reconnue (ou plus récente). Cela permet de garder

trace de la progression dans le temps et de savoir si on lit $\omega_{i,j}(t)$ ou $\omega_{i,j}(t - 1)$, etc. D'un point de vue mathématique, cette asynchronie fait perdre une partie de la garantie de convergence synchrone, bien que de nombreux systèmes multi-agents adoptent un tel schéma. Les conditions de stabilité deviennent plus délicates à analyser : on se rapproche d'un paradigme "Gossip-based" où chaque nœud a une vision potentiellement décalée des liaisons $\omega_{k,\ell}$.

Verrous et Protocoles de Verrouillage

Lorsqu'on est en présence d'une mise à jour *au fil de l'eau*, une option est d'utiliser un *verrou par bloc* : le nœud SCN_p peut verrouiller la plage mémoire correspondant à $\omega^{(p)}$ avant d'écrire, s'assurer que les autres nœuds ont terminé leurs lectures sur le même round, puis relâcher. Cela évite des écritures concurrentes sur la même portion, au prix de possibles contentions si l'on manipule un grand nombre de blocs. Dans le cadre synchrone plus strict, on recourt à une barrière globale : tous les nœuds terminent le calcul local, communiquent leurs modifications, valident l'étape, et ensuite seulement on passe à l'itération $t + 1$. Cela reproduit exactement la formule $\omega(t + 1) = F(\omega(t))$, tout en répartissant le calcul.

Exemple Mathématique de Mise à Jour Synchrone Distribuée

On peut imaginer un SCN divisé en m blocs $\mathcal{V}_1, \dots, \mathcal{V}_m$. Chaque sous-SCN SCN_p calcule, pour $(i, j) \in \mathcal{V}_p \times \mathcal{V}_p$, la mise à jour :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

puis envoie, pour $(i \in \mathcal{V}_p, j \in \mathcal{V}_q)$ avec $p \neq q$, la nouvelle valeur $\omega_{i,j}(t + 1)$ à SCN_q qui mettra à jour la partie inter-blocs si nécessaire. Après ces échanges, tous attendent une barrière. D'un point de vue global, la mise à jour $\omega(t) \mapsto \omega(t + 1)$ a bien été calculée selon une étape unique $O(n^2)$ logiquement, mais physiquement $O(1)$ par blocs SCN_p .

Conclusion

Le fait de **répartir** la matrice ω sur plusieurs nœuds oblige à définir un protocole de **communication** et de **cohérence**. La **version synchrone** met en place une barrière à chaque itération, puis échange les liens *inter-blocs* avant de valider $\omega(t + 1)$. Ce modèle reproduit la sémantique $\omega(t + 1) = F(\omega(t))$ de manière exacte, au prix d'échanges potentiellement importants si la densité de liaisons inter-blocs est forte. L'**asynchronisme**, en revanche, nécessite un système de versioning ou un mécanisme similaire, afin que chaque nœud sache à quel "round" appartient la pondération qu'il lit. Les verrous ou lock-free partiels interviennent alors pour gérer l'écriture concurrente sur des portions de ω . Sur le plan mathématique, on conserve l'idée d'une évolution couplée $\omega^{(p)} \leftrightarrow \omega^{(q)}$, mais on doit se contenter de garanties de convergence moins strictes si on relâche la synchronie totale. Cette problématique, qui touche simultanément l'**ingénierie** (protocole distribué) et l'**analyse** (stabilité asynchrone), illustre l'importance du choix d'un protocole de mise à jour dans la mise en œuvre d'un SCN à très grande échelle.

5.3.3.2. Approche Asynchrone vs. Synchrone pour la Mise à Jour des Liens Inter-sous-SCN

Dans le cadre d'un **SCN** (Synergistic Connection Network) réparti sur plusieurs nœuds ou machines, il apparaît essentiel de gérer la mise à jour des pondérations $\omega_{i,j}$ qui relient des entités appartenant à différents sous-ensembles, notés $\mathcal{V}_1, \dots, \mathcal{V}_m$. Chaque nœud, désigné par SCN_p , est responsable d'une portion locale de la matrice des pondérations, que nous appellerons $\omega^{(p)}$, englobant les liens intra-blocs, c'est-à-dire pour $(i, j) \in \mathcal{V}_p \times \mathcal{V}_p$. Cependant, la gestion des liens inter-blocs, reliant des entités se trouvant sur des nœuds distincts, nécessite une coordination particulière. Deux approches se dessinent : l'une imposant un **synchronisme strict** (méthode « round-based ») et l'autre permettant une **mise à jour asynchrone**, chaque nœud évoluant à son propre rythme.

Dans la méthode **synchrone**, à chaque itération t , tous les nœuds du SCN disposent d'une vision commune et figée de l'état global des pondérations $\omega(t)$. Concrètement, chaque nœud SCN_p lit l'état local $\omega^{(p)}(t)$ et reçoit par communication les valeurs des liaisons inter-blocs, c'est-à-dire celles pour lesquelles $i \in \mathcal{V}_p$ et j appartient à un autre

sous-ensemble \mathcal{V}_q avec $q \neq p$. À partir de ces informations, chaque nœud calcule simultanément sa mise à jour locale conformément à la règle générale de mise à jour du SCN, par exemple sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

et émet les modifications pertinentes vers les autres nœuds pour mettre à jour les liaisons inter-blocs correspondantes. Une barrière de synchronisation (souvent implémentée via des opérations collectives, par exemple *MPI_Barrier* dans les environnements MPI) garantit que l'ensemble du cluster termine sa mise à jour avant de passer à l'itération suivante. Cette approche, qui reproduit rigoureusement le schéma itératif $\omega(t+1) = F(\omega(t))$ défini en section 2.4.1.1, permet d'appliquer les outils théoriques classiques d'analyse de convergence et de stabilité, puisque l'état global est évalué de manière cohérente à chaque round. Néanmoins, le coût de cette synchronisation peut être élevé, en particulier dans des systèmes distribués géographiquement ou hétérogènes, où la latence de communication impose un ralentissement global si un nœud se trouve en retard.

À l'opposé, dans la **mise à jour asynchrone**, chaque nœud évolue de manière autonome sans attendre la fin des calculs des autres. Dans ce cadre, chaque nœud SCN_p maintient son propre compteur d'itérations et met à jour localement les pondérations $\omega_{i,j}$ pour $i \in \mathcal{V}_p$ en utilisant les informations les plus récentes disponibles concernant les liaisons inter-blocs, qui peuvent provenir de communications précédentes et donc être légèrement décalées dans le temps. Formellement, une mise à jour asynchrone peut être modélisée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j; \tilde{\omega}(t-\delta)) - \tau \omega_{i,j}(t)],$$

où $\tilde{\omega}(t-\delta)$ représente la version retardée des pondérations provenant d'un autre nœud, et δ est le délai de communication. D'un point de vue mathématique, cette approche se rapproche d'une mise à jour de type Gauss-Seidel ou Gossip, et la convergence de la dynamique peut être garantie sous certaines conditions de contraction et de bornage des retards, comme discuté dans la littérature sur les systèmes distribués asynchrones. Les avantages de cette approche résident dans sa flexibilité, puisqu'aucun nœud n'est contraint d'attendre que tous les autres se synchronisent, ce qui améliore le débit global du système et le rend plus tolérant aux pannes ou aux différences de performances entre nœuds. Cependant, la complexité d'analyse de la convergence augmente, car l'état global ω n'est plus instantanément cohérent et peut évoluer de manière légèrement désynchronisée, ce qui requiert l'introduction de techniques supplémentaires pour contrôler les oscillations ou le décalage temporel.

La **comparaison** entre ces deux approches révèle ainsi un compromis fondamental. Le mode **synchrone** assure une cohérence globale et simplifie l'analyse théorique de la dynamique d'auto-organisation, en restant fidèle à l'équation itérative centralisée $\omega(t+1) = F(\omega(t))$ présentée en section 2.4.1.1. En revanche, le mode **asynchrone** offre une meilleure tolérance aux latences et une plus grande flexibilité de calcul dans des environnements distribués, mais au prix d'une complexité supplémentaire dans l'analyse de la convergence et la gestion des retards.

Conclusion (5.3.3.2)

En résumé, dans les environnements distribués, le choix entre une mise à jour **synchrone** et **asynchrone** des liens inter-sous-SCN dépend du compromis entre la nécessité de conserver une cohérence d'état global et les contraintes pratiques liées à la latence de communication et aux performances hétérogènes des nœuds. La mise à jour synchrone, en imposant des barrières globales, garantit une transition claire et une analyse mathématique rigoureuse de la convergence, mais peut ralentir le système si un nœud est en retard. À l'inverse, la mise à jour asynchrone permet à chaque nœud d'avancer à son rythme, améliorant ainsi le débit global, tout en imposant des défis supplémentaires pour assurer la stabilité et la cohérence du SCN. Cette dualité, abordée ici, sera approfondie dans les **chapitres 5.3.3.3** et ultérieurs, où des approches hybrides, combinant les avantages des deux modèles, pourront être étudiées afin d'optimiser la dynamique d'auto-organisation dans des architectures distribuées.

5.4.1. Séparation du Calcul de Synergie

La **fonction** $S(i, j)$ n'est pas toujours "une simple distance euclidienne" : elle peut être un **score** de similarité cosinus pour des embeddings, une **compatibilité** logique pour des entités symboliques, ou un **degré** de réussite commune en robotique... Mieux vaut donc **isoler** cette logique dans un **module dédié** plutôt que la noyer dans le code de mise à jour de ω .

5.4.1.1. Raisons : la Fonction $S(i, j)$ Dépend du Type d'Entités (Symbolique, Sub-Symbolique)

Le **calcul** de la fonction de **synergie** $S(i, j)$ dans un **Synergistic Connection Network (SCN)** se trouve au cœur de la dynamique d'auto-organisation (chap. 4). En pratique, les entités \mathcal{E}_i peuvent revêtir des formes très différentes : certaines seront *sub-symboliques*, d'autres *symboliques*, et le réseau peut même cohabiter avec des entités hybrides ou d'autres types plus spécialisés. Cette diversité amène à séparer le **calcul** de $S(i, j)$ dans un **module** dédié, plutôt que de l'implémenter de manière monolithique dans le cœur du SCN.

Il est fréquent qu'un SCN doive manipuler des entités sub-symboliques, comme des vecteurs d'embeddings pour décrire des images ou des documents, et simultanément des entités symboliques, ancrées dans un ensemble de règles logiques ou de concepts ontologiques. Dans la sphère sub-symbolique, la *synergie* entre deux entités $\mathcal{E}_i, \mathcal{E}_j$ prend couramment la forme d'une **distance** (euclidienne, Minkowski, etc.) ou d'une **similarité** (cosinus, RBF kernel, etc.), éventuellement transformée pour la ramener à un score positif ou dans [0,1]. Dans la sphère symbolique, la synergie peut provenir de la **cohérence** (quantité de conflits logiques, alignement conceptuel, correspondances sémantiques), conduisant à un autre type de calcul. Situer les deux extrêmes (sub-symbolique vs. symbolique) dans une même fonction S réclame parfois une combinaison (pondérée ou non) de différentes mesures. Cette hétérogénéité se formalise plus aisément si un *module de synergie* distinct prend en charge la logique de "comment calculer $S(i, j)$ selon la nature des entités $\mathcal{E}_i, \mathcal{E}_j$ ".

Sur le plan de l'**ingénierie logicielle**, une architecture monolithique exigerait d'inscrire tous les calculs de S dans le *noyau* du SCN. Cela devient peu extensible : dès que l'on introduit un nouveau type d'entité (par exemple un nouveau format sub-symbolique, ou une nouvelle sémantique symbolique), on serait contraint de modifier des parties centrales du code. En revanche, en **séparant** le *Module de Synergie*, on fournit un point d'entrée unique pour tout nouveau schéma de calcul S . Chaque type d'entité peut offrir sa propre méthode de similarité, ou on peut définir un "composite" s'il s'agit d'entités hybrides. Cette modularisation respecte la logique exposée au chapitre 5 : on *encapsule* les dépendances et rend possible l'évolution ou la substitution du module de synergie sans affecter la dynamique $\omega_{i,j}$ elle-même.

Sur le plan mathématique, vouloir assujettir toutes les entités à un unique calcul S peut s'avérer très lourd si l'on mélange sub-symbolique et symbolique dans un même référentiel. En dissociant ce *module de synergie*, on obtient une brique que l'on peut étudier ou tester en isolation. Il est alors plus simple d'exiger certaines propriétés (bornes, symétrie, pseudo-métrique, etc.) pour S , voire de prouver qu'elle satisfait des conditions facilitant la convergence (par exemple un certain caractère contractant ou monotone). L'architecture modulaire valorise cette démarche : la partie centrale du SCN ne se questionne pas sur la "substance" de S , elle suppose seulement qu'à chaque itération, pour toute paire (i, j) , le *module de synergie* fournit un score dans un intervalle voulu ($\mathbb{R}^+, [0,1]$, etc.).

Conclusion

Le fait de *séparer* la logique de **calcul** de la *synergie* $S(i, j)$ dans un **module** autonome répond à trois motifs fondamentaux. D'abord, la **nature** hétérogène des entités (symboliques, sub-symboliques, ou hybrides) appelle des calculs multiples et rend périlleux l'idée d'une unique formule universelle codée en dur. Ensuite, la nécessité d'**évolutivité** (pouvoir introduire de nouveaux types d'entités ou modifier la manière de calculer S) conduit naturellement à isoler cette portion du code. Enfin, envisager S comme un *module* distinct favorise la **cohérence mathématique** et la démonstration de propriétés (normes, distances, etc.), dans la mesure où le *noyau* du SCN n'a pas à gérer ces spécificités. Cet agencement s'inscrit dans la logique globale du chapitre 5, dédiée à la construction d'une architecture modulaire et pérenne pour le SCN.

5.4.1.2. Maintenabilité et Extensibilité

La **séparation** du calcul de la fonction $S(i, j)$ dans un module dédié ne s'explique pas seulement par la différence des types d'entités (symboliques vs. sub-symboliques) mentionnée en (5.4.1.1). Un second motif repose sur la **maintenabilité** et la **flexibilité** à long terme, tant du point de vue mathématique que du point de vue de l'ingénierie logicielle. Le fait d'extraire la logique de calcul de S du cœur du Synergistic Connection Network (*SCN*) offre des garanties de facilité d'évolution, de tests et de documentation, assurant que l'ajout ou la modification de la fonction S n'entraîne pas un remaniement lourd de la dynamique globale.

A. Maintenabilité

Une première dimension tient à la probabilité que la fonction de **synergie** $S(i, j)$ doive évoluer au cours de la vie d'un projet ou d'une plate-forme de recherche. Dans de nombreux contextes, la manière de mesurer la cohérence entre deux entités $\mathcal{E}_i, \mathcal{E}_j$ n'est pas figée : on découvre de nouvelles mesures mieux adaptées, on veut combiner plusieurs critères (sub-symbolique, information mutuelle, co-occurrence symbolique), ou encore adapter la pondération en fonction d'un contexte externe. Si le code calculant S est dispersé dans les routines de mise à jour $\omega_{i,j}(t+1)$, chaque modification risque de se répercuter dans un ensemble de classes ou de fichiers, augmentant fortement la probabilité de régressions ou d'incohérences.

En **séparant** la logique dans un “**module de synergie**”, la mise à jour $\omega_{i,j}(t+1)$ (vue au chapitre 4) se borne à invoquer une fonction ou une interface de type $S(i, j)$. Les opérations internes à la dynamique, comme le calcul d'un terme $\eta[S(i, j) - \tau \omega_{i,j}]$, n'ont pas besoin de connaître les détails de la sous-routine qui évalue la similarité ou la compatibilité. Cela facilite la **maintenance** : si S doit être enrichi, remplacé, ou paramétré plus finement, on ajuste ce seul module. Sur le plan des **tests unitaires**, il devient possible de vérifier séparément que le module Synergie délivre les scores attendus selon divers scénarios, sans déployer toute l'infrastructure du SCN (grande matrice ω , inhibition, etc.). Un correctif sur la fonction S n'affectera pas les autres pans du code, préservant ainsi la stabilité du système.

B. Extensibilité

Une seconde dimension touche à l'**extensibilité** : le système doit pouvoir absorber de **nouveaux** types d'entités ou de **nouvelles** formules de synergie. Un SCN peut commencer en manipulant des entités sub-symboliques vectorielles (ex. embeddings d'images) et, plus tard, intégrer des entités purement symboliques, ou hybrides (avec deux volets de représentation). Si la logique de calcul de S reste centralisée, tout ajout de type exigerait de retravailler le code source à de multiples emplacements. En revanche, dans une approche modulaire, on peut imaginer un *registre* ou une *interface polymorphe* : chaque type d'entité, ou chaque couplage de types (sub-symbolique vs. symbolique), peut déclarer ses routines de calcul $S(i, j)$. L'architecture du SCN sollicite ce module de manière unifiée, évitant les modifications lourdes.

Parallèlement, si l'on veut combiner différentes mesures, par exemple

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(i, j) + (1 - \alpha) S_{\text{sym}}(i, j),$$

on peut créer un **module hybride** qui, au moment du calcul, appelle le module “sub” et le module “sym” avant de fusionner les scores. Cet emboîtement de modules apporte une souplesse maximale, permettant de tester ou de raffiner la pondération α sans altérer la dynamique $\omega_{i,j}(t+1)$.

Cette **extensibilité** correspond à la logique “ouvert/fermé” bien connue en ingénierie logicielle : on est ouvert à l'ajout de nouvelles implémentations de S sans être contraint de modifier le noyau du SCN, qui se contente d'invoquer le module défini par l'utilisateur ou l'équipe de recherche.

C. Vue Globale Math-Implémentation

Sur le plan **mathématique**, envisager la fonction S comme un module isolé clarifie l'analyse de la mise à jour $\omega_{i,j}(t+1)$. On peut poser comme “**contrat**” que $S(i, j) \in [0, 1]$ ou \mathbb{R}^+ , éventuellement bornée par 1, et la dynamique du SCN suppose $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$. Les propriétés de convergence, de stabilité, ou de contraction que l'on étudie (chap. 4) dépendent alors d'hypothèses imposées à S (ex. Lipschitz, monotone, etc.), sans exiger de connaître

les détails internes : une entité purement symbolique ou un embedding vectoriel peut satisfaire le même cahier des charges, du moment que le score se conforme à l'API du module.

Côté **implémentation**, cela signifie doter le *Module Synergie* d'une interface explicite (ex. *double synergy(int i, int j)*) ou d'un mécanisme polymorphe selon la nature des entités. Le *Noyau* ne se préoccupe pas de la façon dont on compare deux entités ; il demande juste “donnez-moi un score de synergie”. Le *Module Synergie* se charge de “dispatch” ou de prise de décision si \mathcal{E}_i est sub-symbolique, symbolique, etc. C'est lui également qui gère l'évolutivité : si l'on veut substituer une nouvelle mesure ou en combiner plusieurs, on peut l'implémenter ici.

Conclusion

La **maintenabilité** et l'**extensibilité** constituent des motivations majeures pour placer le **calcul** de la fonction $S(i, j)$ dans un **module** séparé. D'une part, cette approche confère une grande **stabilité** à la codebase : faire évoluer S ou en ajouter de nouvelles versions ne perturbe pas la dynamique $\omega_{i,j}$. D'autre part, cette structuration favorise l'**ouverture** à de multiples types d'entités et de mesures de synergie, essentiel lorsque l'on manipule des représentations hétérogènes (sub-symboliques, symboliques, ou hybrides) ou lorsque l'on explore de nouvelles combinaisons (chap. 3). Au niveau **mathématique**, la modularité met en exergue la brique “ $S(i, j) \mapsto \omega_{i,j}(t + 1)$ ” sous forme d'un *opérateur de mise à jour*, auquel on peut simplement rattacher la fonction S , distincte et interchangeable, rendant la conception plus cohérente et évolutive sur le long terme.

5.4.2. Méthodes d'Implémentation

Au-delà du **principe** de séparer la logique de la synergie S (5.4.1), on doit choisir **comment** l'implémenter concrètement dans le code. Deux grandes approches se distinguent : l'**Approche Polymorphe** (5.4.2.1) et le **Module Central** (5.4.2.2). La première mise en œuvre est fondée sur l'idée que chaque **entité** sait elle-même évaluer la synergie avec une autre, tandis que la seconde recourt à un **orchestrateur** unique identifiant les types et appliquant la bonne fonction S .

5.4.2.1. Approche Polymorphe : chaque entité sait calculer la synergie avec une autre

Lorsqu'un **Synergistic Connection Network** (SCN) doit intégrer des entités de types hétérogènes – par exemple, des entités **sub-symboliques** issues d'embeddings, des entités **symboliques** définies par des règles, ou encore des entités hybrides qui combinent ces deux aspects – il apparaît judicieux d'adopter une approche **polymorphe** dans la conception logicielle. Cette approche consiste à doter chaque entité de la capacité de calculer elle-même sa **synergie** avec une autre entité, c'est-à-dire de déterminer la valeur de $S(i, j)$ en fonction de sa propre représentation et de celle de son interlocuteur.

L'idée fondamentale repose sur la définition d'une **classe de base** ou d'une **interface** commune, que l'on peut nommer par exemple *Entity*. Cette classe déclare une méthode virtuelle (ou abstraite) destinée à être surchargée par chaque sous-classe spécifique. En notation mathématique, on souhaite qu'une entité \mathcal{E}_i dispose d'une fonction

$$S(i, j) = f(\mathcal{E}_i, \mathcal{E}_j),$$

qui exprime la **similarité** ou la **compatibilité** entre \mathcal{E}_i et \mathcal{E}_j . En termes de programmation orientée objet, on peut définir :

```
class Entity { virtual double synergyWith(const Entity& other) const = 0; ... };
```

Chaque type d'entité hérite de cette classe et fournit sa propre implémentation de la méthode *synergyWith*. Par exemple, une **SubEntity** peut être définie pour des données sub-symboliques, stockant un vecteur $\mathbf{x}_i \in \mathbb{R}^d$. Sa méthode de calcul de synergie peut reposer sur la **similarité cosinus**, telle que :

$$S_{\text{sub}}(i, j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

En revanche, une **SymEntity** représentant une entité symbolique, caractérisée par un ensemble de règles \mathcal{R}_i , peut implémenter la méthode *synergyWith* en fonction d'un indice de compatibilité logique, par exemple :

$$S_{\text{sym}}(i, j) = 1 - \frac{\text{nbContradictions}(\mathcal{R}_i, \mathcal{R}_j)}{\text{nbTotal}(\mathcal{R}_i \cup \mathcal{R}_j)}.$$

Dans le cas où une entité sub-symbolique doit être comparée à une entité symbolique, il est alors possible de définir une fonction hybride $S_{\text{hybrid}}(i, j)$ qui combine de manière pondérée les deux critères, par exemple par :

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(i, j) + (1 - \alpha) S_{\text{sym}}(i, j),$$

avec $\alpha \in [0, 1]$ déterminant l'importance relative de la composante sub-symbolique par rapport à la composante symbolique.

Pour que la méthode *synergyWith* prenne correctement en compte non seulement le type de l'objet courant (celui qui appelle la méthode) mais également le type de l'objet passé en argument, on peut recourir à des techniques de **double dispatch** ou au *Visitor pattern*. Ainsi, si l'on a deux entités de types différents, par exemple une SubEntity et une SymEntity, l'appel

```
double s = e1->synergyWith(*e2);
```

se résoudra dynamiquement en invoquant la méthode adaptée à la combinaison des types, par exemple en appelant *e2.synergyWithSubEntity(*e1)* ou une méthode équivalente, de façon à ce que le calcul de $S(i, j)$ soit correctement orienté.

Cette approche présente de nombreux atouts sur le plan de la **cohésion** et de l'**extensibilité**. Chaque classe d'entité intègre la connaissance de sa propre représentation et de la méthode la plus pertinente pour mesurer sa synergie avec une autre entité, ce qui garantit une haute **cohérence** entre la structure interne d'une entité et la manière dont elle interagit. De plus, l'**extensibilité** est facilitée puisque l'ajout d'un nouveau type d'entité ne nécessite que la création d'une nouvelle classe dérivée implémentant *synergyWith*. En revanche, il faut parfois recourir à des mécanismes de **double dispatch** pour gérer les interactions entre types hétérogènes, ce qui peut introduire une complexité supplémentaire au niveau du design.

Voici un extrait de code illustratif en C++ :

```
// Interface de base pour une entité
class Entity {
public:
    virtual ~Entity() {}
    virtual double synergyWith(const Entity &other) const = 0;
};

// Sous-classe pour une entité sub-symbolique
class SubEntity : public Entity {
public:
    std::vector<double> x; // Représentation par un vecteur

    SubEntity(const std::vector<double>& vec) : x(vec) {}

    // Méthode pour calculer la similarité cosinus
    virtual double synergyWith(const Entity &other) const override {
        const SubEntity* subOther = dynamic_cast<const SubEntity*>(&other);
        if(subOther) {
            return cosineSimilarity(this->x, subOther->x);
        }
        // Cas hybride ou incompatibilité : retourner une valeur par défaut ou calculer autrement
        return 0.0;
    }
};
```

```

    }

};

// Sous-classe pour une entité symbolique
class SymEntity : public Entity {
public:
    std::set<std::string> rules; // Ensemble de règles ou axiomes

    SymEntity(const std::set<std::string>& r) : rules(r) {}

// Méthode pour calculer la compatibilité logique
virtual double synergyWith(const Entity &other) const override {
    const SymEntity* symOther = dynamic_cast<const SymEntity*>(&other);
    if(symOther) {
        return computeLogicalCompatibility(this->rules, symOther->rules);
    }
    return 0.0;
}
};


```

Dans cet exemple, la fonction *cosineSimilarity* calcule la similarité entre deux vecteurs, tandis que *computeLogicalCompatibility* évalue la compatibilité entre deux ensembles de règles. On peut bien entendu complexifier ces fonctions pour tenir compte de schémas hybrides ou de pondérations adaptatives.

De même, en Python, on peut définir une structure de classes pour adopter une approche polymorphe. Voici un exemple succinct :

```

import numpy as np
from abc import ABC, abstractmethod

class Entity(ABC):
    @abstractmethod
    def synergy_with(self, other) -> float:
        pass

class SubEntity(Entity):
    def __init__(self, x: np.ndarray):
        self.x = x # vecteur d'embedding

    def synergy_with(self, other: Entity) -> float:
        if isinstance(other, SubEntity):
            return np.dot(self.x, other.x) / (np.linalg.norm(self.x) * np.linalg.norm(other.x))
        # Pour le cas hybride, on peut définir un score fixe ou une fonction mixte
        return 0.0

class SymEntity(Entity):
    def __init__(self, rules: set):
        self.rules = rules

    def synergy_with(self, other: Entity) -> float:
        if isinstance(other, SymEntity):
            nb_total = len(self.rules.union(other.rules))
            if nb_total == 0:
                return 1.0
            nb_common = len(self.rules.intersection(other.rules))
            # On définit la compatibilité comme le rapport d'intersection sur union
            return nb_common / nb_total

```

```

return 0.0

# Exemple d'utilisation
entity1 = SubEntity(np.array([1.0, 0.0, 0.0]))
entity2 = SubEntity(np.array([0.8, 0.1, 0.0]))
entity3 = SymEntity({"rule1", "rule2", "rule3"})
entity4 = SymEntity({"rule2", "rule4"})

print("Synergie entre SubEntity:", entity1.synergy_with(entity2))
print("Synergie entre SymEntity:", entity3.synergy_with(entity4))

```

Ici, la méthode *synergy_with* est définie de manière polymorphe dans chacune des classes dérivées, et permet de calculer le score de synergie selon la nature de l'entité. Dans un SCN complet, la boucle de mise à jour du réseau se contenterait d'appeler cette méthode pour chaque paire d'entités, sans avoir à connaître les détails internes de leur représentation.

Conclusion

L'**approche polymorphe** permet ainsi de décentraliser le calcul de la synergie en déléguant la responsabilité à chaque entité, qui connaît le mieux sa propre représentation et sait comment la comparer à celle d'un autre objet. Cette méthode, qui repose sur le principe de **Résolution dynamique** (dynamic dispatch), facilite l'extension du SCN à des environnements où coexistent des entités de types divers (sub-symboliques, symboliques ou hybrides). Bien que cette approche puisse nécessiter l'emploi de techniques avancées telles que le **double dispatch** pour gérer toutes les combinaisons possibles, elle présente l'avantage de maintenir une **cohérence** interne élevée dans le code et de permettre une **extensibilité** naturelle. Ce paradigme unifie le calcul de la synergie dans un contexte de programmation orientée objet, tout en permettant de conserver une interface uniforme pour la mise à jour des pondérations du SCN.

5.4.2.2. Module Central : un orchestrateur identifie les types et applique la bonne fonction *S*

Dans un **Synergistic Connection Network (SCN)**, la capacité à quantifier la **synergie** entre deux entités \mathcal{E}_i et \mathcal{E}_j est cruciale pour la mise à jour des pondérations $\omega_{i,j}$. Lorsque les entités proviennent de domaines hétérogènes – qu'il s'agisse de représentations **sub-symboliques**, **symboliques** ou **hybrides** – la logique de calcul de la synergie ne peut plus être implémentée de manière éparse dans chaque classe. Au lieu de cela, une approche centralisée est envisagée afin de regrouper l'ensemble des règles de calcul dans un **module central** qui, tel un orchestrateur, identifie les types d'entités et applique la fonction *S* appropriée pour chaque paire. Nous développerons ici le principe de cette solution, ses avantages théoriques et pratiques, puis nous illustrerons le concept par un exemple de code en Python.

A. Principe d'un Orchestrateur Central

L'idée fondamentale consiste à dissocier la **logique de calcul de la synergie** des classes d'entités elles-mêmes. Plutôt que de faire appel à un polymorphisme distribué – où chaque entité doit implémenter une méthode de comparaison prenant en compte toutes les combinaisons de types (ce qui conduit souvent à une complexité de *double dispatch*) – le SCN intègre un **Module Central**, ici dénommé *SynergyOrchestrator*. Ce module agit comme une table de correspondance, en associant à chaque couple de types d'entités $(\text{type}_i, \text{type}_j)$ une fonction spécifique de calcul de la synergie. Autrement dit, pour des entités \mathcal{E}_i et \mathcal{E}_j , le module central procède ainsi :

$$S(\mathcal{E}_i, \mathcal{E}_j) = f_{(\text{type}_i, \text{type}_j)}(\mathcal{E}_i, \mathcal{E}_j),$$

où $f_{(\text{type}_i, \text{type}_j)}$ désigne la fonction adéquate pour traiter la combinaison des types. Par exemple, si \mathcal{E}_i est une **SubEntity** et \mathcal{E}_j une **SymEntity**, la fonction $f_{(\text{Sub}, \text{Sym})}$ sera appelée pour renvoyer un score, qui pourra être défini, dans un cas simple, comme une combinaison pondérée des scores sub-symboliques et symboliques :

$$S_{\text{hybrid}}(i, j) = \alpha S_{\text{sub}}(i, j) + (1 - \alpha) S_{\text{sym}}(i, j),$$

avec $\alpha \in [0,1]$. Le rôle du *SynergyOrchestrator* est donc de recevoir les identifiants de type fournis par chaque entité, de consulter une table de correspondance interne et de déléguer le calcul à la fonction adéquate.

B. Avantages de la Centralisation de la Logique de Synergie

La centralisation de la logique de calcul présente plusieurs avantages majeurs. Premièrement, la **cohésion** du code est grandement améliorée : toutes les règles de calcul sont concentrées dans un même module, ce qui simplifie la maintenance et la compréhension. En cas d'évolution du modèle ou de la nécessité d'ajouter un nouveau type d'entité, il suffit d'étendre la table de correspondance du module central plutôt que de modifier le code de chacune des classes. Cette approche favorise également une **extensibilité** naturelle puisque, pour ajouter un nouveau type tel que « HybridEntity », il est nécessaire d'implémenter les fonctions spécifiques $f_{(\text{Hybrid}, \cdot)}$ et $f_{(\cdot, \text{Hybrid})}$ dans le module central. De plus, la **séparation** entre les données (contenues dans les entités) et la logique de comparaison (gérée par le module central) permet d'obtenir une architecture plus *data-centric* sur le plan mathématique, dans laquelle la fonction S apparaît comme un opérateur unique paramétré par les attributs des entités.

C. Implémentation en Python

Pour illustrer ce concept dans un environnement de prototypage rapide, considérons un exemple en Python qui met en œuvre ce module central. L'exemple suivant présente une version simplifiée où chaque entité expose un identifiant de type et des données internes (par exemple, un vecteur pour une SubEntity ou un ensemble de règles pour une SymEntity). Le module central effectue un *dispatch* en fonction des types et appelle la fonction de calcul appropriée.

```
import numpy as np

# Définition de constantes pour les types d'entités
TYPE_SUB = "SubEntity"
TYPE_SYM = "SymEntity"

# Fonction de calcul de la similarité cosinus pour des vecteurs (pour SubEntity)
def calc_sub_sub(entity1, entity2):
    x1, x2 = entity1.data, entity2.data
    norm1, norm2 = np.linalg.norm(x1), np.linalg.norm(x2)
    if norm1 == 0 or norm2 == 0:
        return 0.0
    return np.dot(x1, x2) / (norm1 * norm2)

# Fonction de calcul de la compatibilité logique pour des ensembles de règles (pour SymEntity)
def calc_sym_sym(entity1, entity2):
    rules1, rules2 = entity1.data, entity2.data
    union_rules = rules1.union(rules2)
    if not union_rules:
        return 1.0 # Si aucun règle, considérer la compatibilité maximale
    common_rules = rules1.intersection(rules2)
    return len(common_rules) / len(union_rules)

# Fonction hybride pour le cas SubEntity vs SymEntity (ici, on retourne une valeur mixte)
def calc_sub_sym(entity1, entity2):
    # Par exemple, retourner 0.0 pour indiquer qu'on ne rapproche pas sub-symbolique et symbolique
    return 0.0

# Module central d'orchestration de la synergie
class SynergyOrchestrator:
    def __init__(self, alpha=0.5):
        self.alpha = alpha
    # Table de correspondance : (type1, type2) -> fonction de calcul
    self.dispatch_table = {
        (TYPE_SUB, TYPE_SUB): calc_sub_sub,
```

```

        (TYPE_SYM, TYPE_SYM): calc_sym_sym,
        (TYPE_SUB, TYPE_SYM): calc_sub_sym,
        (TYPE_SYM, TYPE_SUB): lambda e1, e2: calc_sub_sym(e2, e1)
    }

def compute_synergy(self, entity1, entity2):
    type1 = entity1.get_type()
    type2 = entity2.get_type()
    key = (type1, type2)
    if key in self.dispatch_table:
        return self.dispatch_table[key](entity1, entity2)
    else:
        # Cas par défaut
        return 0.0

# Classes d'entités utilisant une approche minimaliste
class Entity:
    def __init__(self, entity_type, data):
        self.entity_type = entity_type
        self.data = data # Peut être un vecteur (np.array) ou un ensemble de règles (set)

    def get_type(self):
        return self.entity_type

# Exemple d'entités sub-symboliques et symboliques
sub_entity1 = Entity(TYPE_SUB, np.array([1.0, 0.0, 0.0]))
sub_entity2 = Entity(TYPE_SUB, np.array([0.8, 0.1, 0.0]))
sym_entity1 = Entity(TYPE_SYM, {"rule1", "rule2", "rule3"})
sym_entity2 = Entity(TYPE_SYM, {"rule2", "rule4"})

# Création de l'orchestre de synergie
orchestrator = SynergyOrchestrator()

# Calcul des synergies pour différentes paires
synergy_ss = orchestrator.compute_synergy(sub_entity1, sub_entity2)
synergy_qq = orchestrator.compute_synergy(sym_entity1, sym_entity2)
synergy_hybrid = orchestrator.compute_synergy(sub_entity1, sym_entity1)

print("Synergie entre deux SubEntities :", synergy_ss)
print("Synergie entre deux SymEntities :", synergy_qq)
print("Synergie entre une SubEntity et une SymEntity :", synergy_hybrid)

```

Dans cet exemple, le **SynergyOrchestrator** utilise une table de dispatch pour déterminer quelle fonction de calcul appliquer en fonction des types des entités. Les entités elles-mêmes se contentent de stocker leurs données et de fournir leur identifiant de type via la méthode `get_type()`. Ainsi, l'appel à `orchestrator.compute_synergy(e1, e2)` renvoie le score de synergie approprié pour la paire $(\mathcal{E}_1, \mathcal{E}_2)$.

D. Conclusion

L'approche centralisée proposée par le **Module Central** permet d'isoler la logique du calcul de la synergie dans un composant unique, facilitant ainsi la maintenance, l'extension et la compréhension du code. En centralisant l'algorithme de dispatch selon les types d'entités, il devient possible d'assurer que chaque combinaison (sub–sub, sym–sym, sub–sym) est traitée de manière cohérente et transparente. Ce modèle offre un haut degré de modularité, car les entités se limitent à stocker leurs données, tandis que la responsabilité du calcul de $S(i, j)$ est déléguée à un orchestrateur central. Cela simplifie grandement l'extension du SCN à de nouveaux types d'entités et permet de modifier ou d'améliorer les fonctions de synergie de manière localisée, sans avoir à remanier l'ensemble du système.

Le code Python présenté illustre concrètement ce concept et démontre l'efficacité d'une approche orientée objet pour la gestion des interactions hétérogènes dans un SCN.

Cette solution constitue un exemple clair de la manière dont l'**approche polymorphe** et la **centralisation** des fonctions de synergie s'intègrent dans une architecture logicielle solide, favorisant à la fois la **clarté**, l'**extensibilité** et la **maintenabilité** du système.

5.4.3. Gestion du Coût

Dans un **SCN** (Synergistic Connection Network), le calcul de la **synergie** $S(i, j)$ peut représenter un **gros poste** de dépense algorithmique, surtout lorsqu'on considère un nombre n potentiellement élevé d'entités \mathcal{E}_i . Il est donc impératif d'étudier en détail **comment** évaluer ces synergies, et à **quel rythme** on doit les recalculer. Cette section (5.4.3) s'intéresse à la **gestion du coût** d'un tel calcul, d'abord en constatant qu'un schéma naïf revient à un $O(n^2)$ (5.4.3.1), puis en exposant des **optimisations** (5.4.3.2) visant à réduire ou partiellement approximer ces évaluations.

5.4.3.1. Calcul $O(n^2)$ si on évalue $S(i, j)$ pour toutes les paires

Un moyen direct et souvent intuitif pour obtenir la fonction de synergie $S(i, j)$ dans un Synergistic Connection Network consiste à la calculer de manière exhaustive, c'est-à-dire à parcourir toutes les paires (i, j) avec $i, j \in \{1, \dots, n\}$. Cette approche repose sur la formule

$$S(i, j) = \mathcal{F}(\mathbf{data}(\mathcal{E}_i), \mathbf{data}(\mathcal{E}_j)),$$

où $\mathbf{data}(\mathcal{E}_i)$ représente la représentation interne de l'entité \mathcal{E}_i . Sur le plan algorithmique, on se limite alors à une double boucle :

$$\text{for } i = 1 \dots n \quad \text{for } j = 1 \dots n \quad S[i, j] = \text{calcSynergy}(\mathcal{E}_i, \mathcal{E}_j).$$

Si l'on exécute cette opération à chaque itération du SCN (par exemple pour mettre à jour la matrice $\omega(t+1)$ en fonction de la nouvelle synergie S), le coût total devient $O(T \times n^2)$ pour T itérations. Cette section examine dans le détail les conséquences de ce choix et les raisons qui poussent, en pratique, à le tempérer ou à l'optimiser.

A. Justification et intérêt mathématique

L'exhaustivité du calcul de $S(i, j)$ revêt une certaine élégance théorique : on couvre la totalité des paires d'entités, évitant ainsi de négliger d'éventuelles relations synergiques faibles mais susceptibles de s'amplifier au cours de la dynamique. Si la fonction S est simple à évaluer (par exemple, une similarité cosinus entre vecteurs en dimension modeste), alors le coût unitaire est proche de $O(d)$ et le total en $O(d n^2)$ peut demeurer acceptable tant que n n'est pas trop grand.

Il existe également une grande transparence dans cette méthode : on n'a besoin ni de structures complexes (indexation, arbres k-d, etc.), ni de stratégies pour exclure certaines paires (i, j) . On se contente de remplir ou de rafraîchir une matrice S de dimension $n \times n$. Sur le plan mathématique, la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$ peut ainsi s'appuyer en toute simplicité sur $S(i, j)$ pour n'importe quels i, j , sans hypothèses de sparsité ou de filtrage.

B. Limites en mémoire et en temps

Si n prend des valeurs importantes (par exemple $n \gtrsim 10^5$), la structure $S[i, j]$ devient de taille $O(n^2)$, ce qui peut dépasser les capacités mémoire d'une machine standard. Même pour un n de l'ordre de 10^4 , on atteint une matrice de 100 millions de valeurs, ce qui n'est pas impossible, mais déjà conséquent. À chaque itération du SCN, parcourir la totalité de ces entrées pour recalculer S et en dériver $\omega_{i,j}(t+1)$ s'approche d'un milliard d'opérations, qu'il faut en outre répéter T fois si l'on veut un SCN avec T itérations. Le coût asymptotique $O(T n^2)$ se heurte rapidement à des barrières pratiques lorsque n franchit la dizaine de milliers ou plus.

Au plan numérique, l'hypothèse $O(1)$ pour chaque calcul $S(i, j)$ peut se révéler trompeuse si la dimension des vecteurs est élevée, ou si S implique une logique plus complexe (ex. compatibilité symbolique examinant des structures de règles). On peut alors se retrouver avec $O(T \times n^2 \times \alpha)$, où α traduit le coût unitaire de la fonction S .

C. Dynamique du SCN et réactualisation de S

Une question se pose ensuite : doit-on recalculer $S(i, j)$ à chaque itération, ou est-il suffisant de le faire occasionnellement ? Si les entités \mathcal{E}_i sont stationnaires (leur contenu ne change pas), on peut fixer $S(i, j)$ une fois pour toutes au début et le réutiliser dans la suite des itérations. Le coût $O(n^2)$ se limite alors à l'initialisation. En revanche, dans un cadre plus évolutif (entités modifiées, flux de données), S peut fluctuer dans le temps, imposant une réévaluation fréquente, voire continue.

Cette réévaluation coûteuse se combine alors au schéma de mise à jour $\omega_{i,j}(t+1) = \dots$. Sur un plan purement mathématique, on sait décrire le SCN comme un système d'équations couplées, mais l'implémentation en $O(n^2)$ par itération peut se révéler irréaliste pour de larges échelles. Cela motive une réflexion sur des **optimisations** qui ne traitent pas toutes les paires (i, j) , ou qui recourent à des structures plus sélectives (chap. 5.4.3.2).

D. Architecture logicielle et partition

Si, malgré tout, on choisit le calcul exhaustif S en $O(n^2)$, il est intéressant de se demander comment l'**architecture** logicielle (chap. 5.2.3) peut l'aider à gagner en efficacité. Un partitionnement des entités en sous-blocs, accompagné d'un protocole de communication (synchrone ou asynchrone), peut répartir la charge de calcul S sur plusieurs nœuds. On ne résout cependant pas le facteur $O(n^2)$ total, on ne fait que le distribuer (et éventuellement le paralléliser).

Par ailleurs, la décision de stocker S de façon dense (matrice de taille $n \times n$) s'accompagne d'une problématique de **persistante** (5.3.1) : pour un grand n , la place disque ou mémoire devance souvent les capacités disponibles, à moins de disposer d'une infrastructure de calcul massif. Certains systèmes permettent d'exploiter des GPU ou clusters HPC, mais la multiplication des transferts de données peut introduire des goulets d'étranglement importants.

Conclusion

Le calcul $S(i, j)$ en **approche exhaustive** $O(n^2)$ est la méthode la plus directe pour évaluer la synergie entre toutes les paires, garantissant une couverture intégrale des liens potentiels et une grande **simplicité** au niveau de l'implémentation (double boucle ou matrice dense). Cette solution présente une **transparence** et une **complétude** mathématiques, convenant aux scénarios où n reste **modéré** et où la synergie n'a pas besoin d'être recalculée trop souvent. Cependant, elle devient **prohibitive** à plus grande échelle, tant en temps qu'en mémoire. La section suivante (5.4.3.2) introduit justement les **optimisations** usuelles pour échapper à l'impasse $O(n^2)$: on retiendra notamment l'idée de ne considérer que les plus proches voisins dans l'espace (k-NN, ϵ -radius, index spatial) afin de réduire fortement le nombre de paires (i, j) examinées et calculer S seulement là où elle est susceptible d'être non négligeable.

5.4.3.2. Optimisations : k-NN, ϵ -radius, indexation spatiale

Le calcul exhaustif $S(i, j)$ en $O(n^2)$ (présenté en 5.4.3.1) se révèle rapidement prohibitif pour de grands n . Afin d'y remédier, il est courant de **cibler** le calcul sur un sous-ensemble de paires (i, j) jugées plus pertinentes, ou d'accélérer la recherche de voisinages. Plusieurs stratégies interviennent alors : exploiter la notion de "k plus proches voisins" (k-NN), utiliser un " ϵ -radius" pour ignorer les entités trop distantes, ou recourir à des **structures d'indexation spatiale** (k-d trees, vantage-point trees, ANN). Ces solutions permettent, d'un point de vue mathématique et algorithmique, de **réduire** le nombre de calculs $S(i, j)$ ou d'en **accélérer** l'identification, et donc de sortir du carcan $O(n^2)$.

A. k-NN (k plus proches voisins)

Les techniques de "k plus proches voisins" consistent à limiter la recherche de synergie $S(i, j)$ aux seuls j qui se trouvent dans le **top-k** de i en termes de proximité ou de similarité. Si la représentation \mathbf{x}_i est un vecteur dans \mathbb{R}^d , alors la "distance" (euclidienne ou autre) sert de guide : au lieu de comparer \mathbf{x}_i à toutes les \mathbf{x}_j , on se borne aux k entités les plus similaires. En pratique, on détermine

$$\text{kNN}(i) = \{j \mid \mathbf{x}_j \text{ est parmi les } k \text{ plus proches de } \mathbf{x}_i\}.$$

Le calcul effectif de la k-NN dans un espace vectoriel naïf reste $O(n)$ par requête, menant à $O(n^2)$ global pour tous les i . Cependant, on combine généralement cette idée avec des **structures d'index** plus avancées (cf. infra) ou des algorithmes d'**approximate nearest neighbors** (FAISS, Annoy, etc.), qui réduisent la complexité à $O(n \log n)$ (ou un peu plus) en moyenne. On obtient alors un stockage final $O(k n)$ pour la synergie, au lieu de $O(n^2)$.

D'un point de vue mathématique, cette approche revient à nier la pertinence de la synergie au-delà de ces k voisins, hypothèse justifiée par une décroissance rapide de S ou une complémentarité faible en dehors du voisinage local. Dans le contexte d'un SCN (Synergistic Connection Network), cela recoupe la notion de "voisinage restreint" ou "parsimonie", évitant de consacrer des ressources à des comparaisons lointaines et souvent négligeables.

B. ϵ -radius : filtrer par distance

Une seconde famille de méthodes, souvent apparentée, repose sur la définition d'un **rayon ϵ** . On ne s'intéresse qu'aux entités \mathcal{E}_j qui se trouvent à une distance inférieure à ϵ de \mathbf{x}_i (selon un certain metric ou kernel). Cela fait sens si la synergie se base sur un *décroissement* rapide avec la distance :

$$S(i, j) = f(\|\mathbf{x}_i - \mathbf{x}_j\|) \quad \text{avec } f(r) \approx 0 \text{ si } r > \epsilon.$$

Ainsi, pour chaque \mathbf{x}_i , on exécute une **recherche** dans un rayon ϵ . Là encore, en mode naïf, on testerait toutes les \mathbf{x}_j ($O(n)$ par entité, menant à $O(n^2)$ global). On recourt donc à des **structures** spécialisées (k-d tree, vantage-point tree, etc.) ou à des algorithmes d'**indexation** accélérant la "range search". Le résultat est une localité : si \mathbf{x}_j est trop éloigné, on ne calcule même pas S .

Ce rayon ϵ est adapté selon le **domaine** et la **géométrie** (en robotique, en vision...). Sur un plan mathématique, on introduit un "seuil" d'influence spatiale. Dans un SCN, cela se traduit par :

$$\omega_{i,j}(t+1) = 0 \quad \text{si } \|\mathbf{x}_i - \mathbf{x}_j\| > \epsilon,$$

et donc aucun calcul explicite de S pour ces paires.

C. Indexation spatiale

Afin de rendre efficaces les recherches "k plus proches voisins" ou " ϵ -radius", on utilise des **structures d'indexation** dans l'espace :

- **k-d tree**, vantage-point tree, ball tree, R*-tree, etc. Dans un espace \mathbb{R}^d de dimension modeste ($<= 20-30$), ces arbres réduisent le parcours moyen pour un range search ou un k-NN search, passant d'un $O(n)$ naïf à $O(n^\alpha)$, $\alpha < 1$.
- **Approximate nearest neighbors (ANN)** si la dimension est plus grande, recourant à des solutions comme Faiss (Facebook AI Similarity Search), Annoy (Spotify), HNSW, etc. Elles ne donnent pas toujours *exactement* le top-k ou l'exact ϵ -voisinage, mais en pratique leur "faible erreur" suffit dans un SCN où l'on ne cherche pas la perfection.

Ces techniques, d'un point de vue mathématique, constituent une pré-organisation de l'espace \mathbf{x}_i , où l'on sait rapidement restreindre la recherche dans une zone d'intérêt ou un "cluster potentiel". Sur le plan ingénierie, on construit l'index (qui peut se mettre à jour si les entités changent), puis on exécute des requêtes k-NN ou range à chaque besoin de calcul S . Cela ramène l'effort global parfois à $O(n \log n)$ ou $O(n^{1+\epsilon})$, bien plus viable que $O(n^2)$.

D. Bénéfices et limites

Ces trois idées (k-NN, ϵ -radius, indexation spatiale) sont souvent **combinées** :

- Sur le **plan mathématique**, on accepte une **localité** : la synergie S s'avère non négligeable que dans un "voisinage" restreint, qu'il soit déterminé par un rayon ϵ ou par une "frontière de k plus proches".

- Cela diminue drastiquement le **nombre** de paires (i, j) à évaluer, passant d'un potentiel $O(n^2)$ à un $O(n k)$ ou $O(n \log n)$ si l'indexation fonctionne bien.
- La limite est que cette hypothèse *peut* omettre certains liens si la synergie n'est pas strictement localisée en distance. Toutefois, pour de nombreux cadres sub-symboliques (embeddings, distances décroissantes), c'est parfaitement cohérent.

Sur un plan pratique, ces méthodes requièrent la **mise à jour** ou la **reconstruction** périodique de l'index (si les entités évoluent ou si de nouvelles arrivent), ce qui ajoute un surcoût, mais généralement bien moindre que le balayage complet $O(n^2)$.

L'efficacité dépend aussi de la **dimension** : un k-d tree peut perdre en performance si d est trop élevé. Dans ce cas, les algorithmes d'**approximate nearest neighbors** prennent le relais, tolérant une petite erreur mais préservant une grande rapidité pour trouver les "meilleurs" voisins.

Conclusion

Les **optimisations k-NN**, ϵ -radius, et **indexation spatiale** forment un **socle** de stratégies pour éviter ou amortir le coût $O(n^2)$ dans le calcul de la synergie $S(i, j)$. Elles consistent à **focaliser** la recherche sur un **voisinage local** jugé le plus pertinent, que ce soit en limitant le nombre (k) de voisins ou en imposant un rayon ϵ . D'un point de vue mathématique, on suppose une **localisation** de S autour de points proches ; sur le plan ingénierie, on s'appuie sur des **structures** (k-d tree, ANN) pour accélérer ce ciblage. L'effet global est de passer d'un régime $O(n^2)$ parfois inabordable, à un schéma plus abordable $O(n \log n)$ ou $O(kn)$, rendant possible le calcul fréquent ou continu de S dans un SCN à plus grande échelle.

5.5. Module de Mise à Jour ω et Inhibition/Contrôle

5.5.1. Rappels des Règles DSL

- 5.5.1.1. Equation additive $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$.
- 5.5.1.2. Variantes (multiplicative, etc.).

5.5.2. Inhibition et Compétition

- 5.5.2.1. Formule $-\gamma \sum_{k \neq j} \omega_{i,k}$ dans la mise à jour.
- 5.5.2.2. Paramétrage γ .
- 5.5.2.3. Application concrète : limiter la somme des liens sortants par entité.

5.5.3. Saturation

- 5.5.3.1. Clipping $\omega_{i,j} \leftarrow \min(\omega_{i,j}, \omega_{\max})$.
- 5.5.3.2. Réglage de ω_{\max} et son effet sur l'émergence de clusters.

5.5.4. Recuit Simulé ou Bruit

- 5.5.4.1. Ajouter un terme stochastique $\xi_{i,j}$ dans la mise à jour.
- 5.5.4.2. Diminution progressive de la “température” pour éviter de rester piégé dans un minimum local.

Dans l'architecture globale du SCN, on retrouve un **Module** spécifiquement dédié à la **mise à jour** des pondérations $\omega_{i,j}$. Ce module occupe une place centrale : il assure la transition $\omega(t) \rightarrow \omega(t+1)$ à chaque itération, en se basant sur la **synergie** $S(i,j)$ (chap. 5.4) et d'éventuels **mécanismes** de contrôle (inhibition, saturation, recuit stochastique, etc.). En ce sens, il concrétise la boucle fondamentale du Deep Synergy Learning (DSL) décrite en chapitre 4.

Dans cette section (5.5), nous commençons par rappeler les **règles** DSL les plus courantes pour la mise à jour ω (5.5.1), avant de détailler la manière dont on peut y introduire une **inhibition compétitive** (5.5.2), une **saturation** (5.5.3) et, si besoin, un **recuit simulé** ou un **bruit stochastique** (5.5.4). Sur le plan mathématique, chaque composante (inhibition, saturation, bruit) peut être vue comme un opérateur supplémentaire venant se superposer à la formule de base, modulant la convergence et la formation des clusters.

5.5.1. Rappels des Règles DSL

Les travaux exposés en chapitres 2 et 4 introduisent déjà la dynamique d'un SCN : on y voit comment $\omega_{i,j}$ se met à jour en fonction de $S(i,j)$, d'un taux d'apprentissage η et d'un paramètre de décroissance τ . Ici, nous nous concentrerons sur l'**implémentation** et l'**intégration** de cette mise à jour dans le **Module** correspondant, tout en rappelant les éléments mathématiques essentiels.

Voici une version développée, structurée et enrichie de la section 5.5.1.1. **Équation additive** $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$, suivie d'une implémentation en Python qui traduit concrètement ce schéma de mise à jour.

5.5.1.1. Équation additive $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$

Les **mises à jour additives** représentent l'une des formulations les plus intuitives et classiques dans l'analyse d'un **Synergistic Connection Network (SCN)**. Le principe fondamental est de faire évoluer chaque **pondération** $\omega_{i,j}$ en fonction d'un terme correctif qui combine deux contributions opposées : d'une part, l'**attraction** induite par la **synergie** $S(i,j)$ qui tend à augmenter $\omega_{i,j}$ et, d'autre part, une **décroissance linéaire** proportionnelle à $\omega_{i,j}$ elle-même, contrôlée par le paramètre τ . Cette dualité assure que la mise à jour guide $\omega_{i,j}$ vers un **équilibre** ou **point fixe** donné par

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

A. Forme canonique et point fixe

La **formule additive** est donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Ici, le **taux d'apprentissage** $\eta > 0$ détermine l'amplitude des corrections à chaque itération, tandis que $\tau > 0$ module la **force de décroissance** appliquée à $\omega_{i,j}(t)$. Au **point fixe** de cette dynamique, lorsque $\omega_{i,j}(t+1) = \omega_{i,j}(t) = \omega_{i,j}^*$, l'équation se simplifie en

$$\omega_{i,j}^* = \omega_{i,j}^* + \eta [S(i,j) - \tau \omega_{i,j}^*] \Rightarrow S(i,j) = \tau \omega_{i,j}^*,$$

ce qui implique immédiatement

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cette solution montre que, dans des conditions idéales et en l'absence d'effets additionnels (tels que l'inhibition ou le bruit), la **pondération** tend vers un équilibre directement proportionnel à la **synergie** et inversement proportionnel à τ .

B. Interprétation mathématique et convergence

En réécrivant l'équation de mise à jour, on peut mettre en évidence un schéma linéaire :

$$\omega_{i,j}(t+1) = (1 - \eta \tau) \omega_{i,j}(t) + \eta S(i,j).$$

Ce schéma se présente comme une combinaison linéaire entre l'**état** antérieur et un **terme forçant** la valeur vers $\eta S(i,j)$. Pour que le système converge de manière stable vers $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$, il est impératif que la contraction imposée par le facteur $(1 - \eta \tau)$ soit suffisante, ce qui se traduit par la condition

$$\eta \tau < 2.$$

Sous cette hypothèse, l'écart $\left| \omega_{i,j}(t) - \frac{S(i,j)}{\tau} \right|$ décroît de façon exponentielle au fil des itérations, garantissant une **convergence stable**.

De plus, on peut associer à cette mise à jour une vision en termes de **pseudo-énergie**. En définissant une fonction potentielle

$$\mathcal{J}(\omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{i,j}^2,$$

la condition de premier ordre pour minimiser \mathcal{J} (c'est-à-dire $\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = 0$) conduit exactement à

$$-S(i,j) + \tau \omega_{i,j} = 0 \Rightarrow \omega_{i,j} = \frac{S(i,j)}{\tau}.$$

Ainsi, le système peut être interprété comme effectuant une **descente de gradient** sur \mathcal{J} , orientant les mises à jour vers un minimum de cette fonction potentielle.

C. Implémentation dans un SCN

Du point de vue **algorithmique**, l'implémentation de la règle additive est souvent réalisée via une boucle itérative qui applique, pour chaque paire (i, j) , la formule de mise à jour. Une approche recommandée consiste à utiliser un **double-buffer** afin d'éviter la modification de $\omega(t)$ en temps réel lors du calcul des mises à jour, garantissant ainsi la cohérence des valeurs lues pour l'ensemble des paires. L'architecture logicielle typique comporte un module de **mise à jour** chargé de parcourir la matrice de pondérations, de récupérer les valeurs de la **synergie** $S(i, j)$ à partir d'un module dédié, et d'appliquer la formule suivante :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)].$$

Ce module peut également intégrer des mécanismes complémentaires tels que l'**inhibition** (par exemple, un terme supplémentaire $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$) ou des techniques de **clipping** pour maintenir les valeurs dans un intervalle souhaité.

D. Exemple de Comportement et Implémentation en Python

Pour illustrer concrètement la mise à jour additive, considérons un exemple simple en Python. Supposons que la synergie $S(i, j)$ soit donnée et stationnaire, et que l'on souhaite observer la convergence de la pondération $\omega_{i,j}(t)$ vers la valeur $\frac{S(i, j)}{\tau}$.

Voici une implémentation en Python :

```

import numpy as np
import matplotlib.pyplot as plt

# Paramètres de la mise à jour
eta = 0.05      # Taux d'apprentissage
tau = 1.0        # Facteur de décroissance
num_iterations = 50 # Nombre d'itérations

# Supposons que S(i,j) est donné et constant
# Pour cet exemple, nous considérons une paire d'entités pour laquelle S(i,j) = 0.8
S_ij = 0.8

# Initialisation de la pondération omega avec un bruit faible (près de 0)
omega = 0.0 + np.random.uniform(-0.01, 0.01)

# Liste pour stocker l'évolution de omega
omega_history = [omega]

# Boucle de mise à jour selon la règle additive
for t in range(num_iterations):
    # Calcul du delta selon la règle additive
    delta = eta * (S_ij - tau * omega)
    # Mise à jour de omega
    omega = omega + delta
    # Enregistrement de l'état courant
    omega_history.append(omega)

# Calcul de la valeur théorique de convergence (point fixe)
omega_fixed = S_ij / tau

# Affichage graphique de l'évolution de omega
plt.figure(figsize=(8, 5))

```

```

plt.plot(omega_history, marker='o', linestyle='-', color='b', label=r'$\omega_{i,j}(t)$')
plt.axhline(y=omega_fixed, color='r', linestyle='--', label=r'$S(i,j)/\tau$')
plt.title("Convergence de la pondération $\omega_{i,j}(t)$ selon la règle additive")
plt.xlabel("Itérations $t$")
plt.ylabel(r"$\omega_{i,j}(t)$")
plt.legend()
plt.grid(True)
plt.show()

```

Dans cet exemple, nous initialisons la pondération $\omega_{i,j}(0)$ avec une valeur proche de 0 et nous appliquons la mise à jour additive sur 50 itérations. Le graphique montre que $\omega_{i,j}(t)$ converge progressivement vers la valeur théorique $\omega_{i,j}^* = \frac{S(i,j)}{\tau} = 0.8$. Le schéma de mise à jour est exprimé mathématiquement par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + 0.05 (0.8 - 1.0 \omega_{i,j}(t)).$$

La condition de stabilité, $\eta \tau < 2$, est vérifiée puisque $0.05 \times 1.0 = 0.05 < 2$, assurant ainsi une **convergence stable** sans oscillations.

E. Conclusion

La règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

représente une approche simple et efficace pour la mise à jour des pondérations dans un SCN. Elle conduit chaque $\omega_{i,j}$ à converger vers le point fixe $\frac{S(i,j)}{\tau}$ lorsque la synergie est stationnaire et que les paramètres η et τ sont choisis de manière appropriée. L'implémentation en Python illustre comment, à l'aide d'un **double-buffer** conceptuel et d'une boucle itérative, la dynamique mathématique se traduit en une routine algorithmique concrète. Ainsi, cette approche constitue le cœur du mécanisme d'auto-organisation dans un SCN, permettant de fusionner la théorie (descente de gradient et pseudo-énergie) avec la pratique (mise à jour itérative des pondérations) dans une infrastructure logicielle robuste.

Ce développement fournit à la fois une compréhension approfondie de la mise à jour additive dans le cadre du DSL et un exemple d'implémentation en Python qui permet de visualiser la convergence des pondérations vers leur équilibre théorique.

Voici une présentation détaillée de la section **5.5.1.2. Variantes (multiplicative, etc.)** rédigée dans un style académique, intégrant de nombreuses formules mathématiques et une implémentation complète en Python avec visualisation graphique.

5.5.1.2. Variantes (multiplicative, etc.)

Dans le cadre du **Synergistic Connection Network (SCN)**, la règle additive présentée en section 5.5.1.1 constitue l'approche la plus simple pour mettre à jour la matrice des pondérations $\omega_{i,j}$. Cependant, d'autres formulations permettent de modifier la nature de la correction appliquée à ces pondérations. L'une des alternatives notoires est l'**approche multiplicative**, qui, au lieu d'ajouter un delta fixe à $\omega_{i,j}$, applique un facteur multiplicatif qui dépend de la valeur courante de $\omega_{i,j}$. Ce type de mise à jour est particulièrement pertinent dans les systèmes où l'on souhaite que les liens déjà forts se renforcent plus rapidement, tout en diminuant proportionnellement les liens faibles.

A. Formulation Multiplicative

La règle multiplicative de base se définit par l'équation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)[1 + \eta(S(i,j) - \tau\omega_{i,j}(t))].$$

Dans cette formulation, le terme $\eta(S(i,j) - \tau\omega_{i,j}(t))$ représente la correction relative appliquée à $\omega_{i,j}(t)$. Si ce terme est positif, le facteur multiplicatif dépasse 1 et le lien se renforce de façon proportionnelle à sa valeur actuelle, favorisant ainsi une dynamique « effet boule de neige ». À l'inverse, lorsque le terme est négatif, le facteur est inférieur à 1 et $\omega_{i,j}(t)$ diminue.

L'intérêt de cette approche réside dans sa capacité à **amplifier** les liens déjà établis. En effet, un lien dont $\omega_{i,j}(t)$ est élevé bénéficiera d'une multiplication plus forte (à condition que $S(i,j) - \tau\omega_{i,j}(t)$ reste positif), tandis qu'un lien faible aura du mal à sortir de l'état marginal, renforçant ainsi la polarisation des connexions au sein du SCN.

B. Comparaison avec l'Approche Additive

Dans la mise à jour **additive**, la modification est donnée par un incrément absolu :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau\omega_{i,j}(t)],$$

ce qui signifie que la correction appliquée ne dépend pas de la valeur courante de $\omega_{i,j}(t)$. Ce schéma induit un mouvement linéaire vers le point fixe $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$.

Dans le cas multiplicatif, la correction est proportionnelle à $\omega_{i,j}(t)$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)[1 + \eta\Delta(t)] \quad \text{avec} \quad \Delta(t) = S(i,j) - \tau\omega_{i,j}(t).$$

Ainsi, si $\omega_{i,j}(t)$ est déjà élevé, la mise à jour entraîne une augmentation plus importante, tandis qu'un lien faible sera mis à jour de manière moins significative. Cette **dynamique exponentielle** peut favoriser la formation de clusters très marqués, mais elle nécessite également une gestion rigoureuse des paramètres pour éviter une divergence (risque d'explosion) ou des oscillations.

C. Autres Variantes et Approches Hybrides

Outre l'approche purement multiplicative, il est envisageable de combiner les aspects additifs et multiplicatifs pour obtenir un schéma **semi-additif** ou **hybride**. Par exemple, une formulation hybride peut être exprimée par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \alpha\omega_{i,j}(t)[S(i,j) - \tau\omega_{i,j}(t)] + \beta[S(i,j) - \tau\omega_{i,j}(t)],$$

où α et β sont des coefficients permettant d'ajuster l'influence relative des termes multiplicatif et additif. Un tel schéma offre une flexibilité accrue, permettant de moduler finement la dynamique des mises à jour en fonction des besoins spécifiques du système.

D. Implémentation Python avec Visualisation

Pour illustrer la règle multiplicative, nous proposons ci-après une implémentation en Python qui simule l'évolution de la pondération $\omega_{i,j}$ pour une paire d'entités, puis affiche la courbe de convergence à l'aide de **matplotlib**.

```
import numpy as np
import matplotlib.pyplot as plt

# Paramètres de mise à jour
eta = 0.05    # Taux d'apprentissage
tau = 1.0     # Facteur de décroissance
num_iterations = 50 # Nombre d'itérations
S_ij = 0.8   # Synergie S(i,j) pour la paire considérée
```

```

# Initialisation de la pondération omega avec un bruit faible autour de 0
omega = 0.0 + np.random.uniform(-0.01, 0.01)

# Liste pour stocker l'évolution de omega
omega_history = [omega]

# Boucle de mise à jour multiplicative
for t in range(num_iterations):
    # Calcul du delta multiplicatif
    delta = eta * (S_ij - tau * omega)
    # Mise à jour multiplicative : on multiplie la valeur courante par un facteur
    omega = omega * (1 + delta)
    # Optionnel : on peut appliquer un clipping pour éviter les valeurs négatives
    if omega < 0:
        omega = 0
    omega_history.append(omega)

# Calcul de la valeur théorique de convergence (point fixe) pour comparaison
omega_fixed = S_ij / tau

# Affichage graphique de l'évolution de omega
plt.figure(figsize=(8, 5))
plt.plot(omega_history, marker='o', linestyle='-', color='blue', label=r'$\omega_{i,j}(t)$')
plt.axhline(y=omega_fixed, color='red', linestyle='--', label=r'$S(i,j)/\tau$')
plt.title("Convergence de la pondération $\omega_{i,j}(t)$ avec la règle multiplicative")
plt.xlabel("Itérations $t$")
plt.ylabel(r"$\omega_{i,j}(t)$")
plt.legend()
plt.grid(True)
plt.show()

```

Dans cet exemple, nous initialisons la pondération $\omega_{i,j}(0)$ avec une valeur proche de 0 et appliquons la mise à jour multiplicative pour 50 itérations. La formule utilisée est :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \left[1 + \eta (S(i,j) - \tau \omega_{i,j}(t)) \right],$$

ce qui signifie que la valeur de $\omega_{i,j}(t)$ est multipliée par le facteur $\left[1 + \eta (S(i,j) - \tau \omega_{i,j}(t)) \right]$. La figure générée montre la convergence de $\omega_{i,j}(t)$ vers la valeur théorique $\frac{S(i,j)}{\tau} = 0.8$ de manière exponentielle, tout en illustrant le comportement non linéaire inhérent à la formulation multiplicative.

E. Conclusion

La variante multiplicative

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) \left[1 + \eta (S(i,j) - \tau \omega_{i,j}(t)) \right]$$

offre une alternative intéressante à la règle additive en amplifiant proportionnellement la correction selon la valeur courante de $\omega_{i,j}(t)$. Cette approche peut accélérer la formation de liens forts et favoriser des dynamiques de polarisation, tout en nécessitant un contrôle rigoureux des paramètres pour éviter une croissance explosive. L'implémentation en Python présentée ci-dessus illustre la convergence vers le point fixe et offre un moyen visuel d'appréhender le comportement de la mise à jour multiplicative dans un SCN. Ce type de schéma, éventuellement combiné avec des approches hybrides ou semi-additives, constitue une composante essentielle dans l'ensemble des stratégies d'auto-organisation que l'on retrouve dans les SCN, et sert de base pour des développements ultérieurs dans le cadre de systèmes plus complexes et adaptatifs.

5.5.2.1. Formule $-\gamma \sum_{k \neq j} \omega_{i,k}$ dans la mise à jour

Dans un SCN, il est souvent souhaitable d'instaurer une **compétition** entre les liens issus d'une même entité. En d'autres termes, lorsqu'une entité \mathcal{E}_i renforce la connexion avec l'une de ses cibles \mathcal{E}_j via la pondération $\omega_{i,j}$, il est pertinent de freiner simultanément la croissance des autres connexions $\omega_{i,k}$ pour $k \neq j$. Cette idée, qui s'inspire notamment des mécanismes d'**inhibition latérale** observés dans les réseaux neuronaux biologiques, se formalise mathématiquement par l'ajout d'un terme négatif dans la règle de mise à jour.

A. Idée Générale de l'Inhibition Compétitive

L'objectif est de limiter la capacité d'une entité à « investir » simultanément dans de nombreux liens forts. On introduit ainsi un **budget** de pondération pour chaque entité \mathcal{E}_i . En d'autres termes, si \mathcal{E}_i a déjà alloué une part importante de son « capital » de connexion à certains partenaires, il lui reste moins de ressources pour renforcer d'autres liens. Ce mécanisme permet d'obtenir une **spécialisation** des connexions et d'éviter une croissance simultanée excessive qui pourrait conduire à un réseau peu discriminant.

B. Sens et Rôle du Terme d'Inhibition

Le terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ introduit une pression négative sur la mise à jour du lien $\omega_{i,j}(t)$. Plus précisément, si l'ensemble des connexions issues de l'entité \mathcal{E}_i est déjà élevé, la somme $\sum_{k \neq j} \omega_{i,k}(t)$ sera grande et la correction négative sera plus importante. Cela se traduit par une limitation de l'augmentation de $\omega_{i,j}(t)$, ce qui force l'entité à concentrer ses ressources sur un nombre restreint de liens – idéalement ceux présentant une synergie forte. En d'autres termes, ce mécanisme de **compétition latérale** veille à ce que l'augmentation d'un lien ne se fasse qu'au détriment des autres, créant ainsi un environnement où seuls quelques liens majeurs émergent.

C. Modélisation Mathématique

Considérons la règle de mise à jour additive classique pour la pondération d'un lien, qui s'exprime sans inhibition par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Pour intégrer la notion de compétition entre les connexions issues d'une même entité, on ajoute un terme d'inhibition :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Ici, η est le taux d'apprentissage qui détermine l'amplitude de la correction, τ régule la décroissance linéaire de $\omega_{i,j}$, et γ contrôle l'intensité de l'inhibition compétitive. Cette équation montre que, quelle que soit la synergie $S(i,j)$, le renforcement du lien $\omega_{i,j}$ sera toujours freiné par la somme des autres liens issus de \mathcal{E}_i .

D. Effets sur la Dynamique du Réseau

L'introduction du terme d'inhibition a plusieurs conséquences notables sur la dynamique du SCN :

- **Spécialisation des Connexions** : Seuls les liens pour lesquels la synergie $S(i,j)$ est suffisamment élevée pourront dépasser l'effet inhibiteur, tandis que les autres resteront faibles. Cela favorise l'émergence de clusters où chaque entité concentre ses connexions sur un nombre limité de partenaires.
- **Prévention de l'Emballlement Global** : Même si plusieurs paires présentent une synergie élevée, l'inhibition globale empêche que toutes les connexions soient simultanément fortes, ce qui pourrait diluer l'information sur la structure réelle du réseau.

- **Risque d'Oscillations :** Si le paramètre γ est trop élevé, le frein imposé peut entraîner des oscillations dans la dynamique des mises à jour, ce qui nécessitera alors des ajustements ou l'introduction de mécanismes de saturation pour stabiliser le comportement.

E. Implémentation Python avec Visualisation

Pour illustrer la dynamique induite par la mise à jour avec inhibition compétitive, nous proposons ci-après une implémentation en Python. Le script simule l'évolution de la pondération $\omega_{i,j}(t)$ pour une entité \mathcal{E}_i connectée à plusieurs cibles \mathcal{E}_j et affiche l'évolution de ces poids au cours des itérations.

```

import numpy as np
import matplotlib.pyplot as plt

# Paramètres de la mise à jour
eta = 0.05      # Taux d'apprentissage
tau = 1.0        # Facteur de décroissance
gamma = 0.02     # Coefficient d'inhibition
num_iterations = 100 # Nombre d'itérations
n_links = 5      # Nombre de liens sortants pour l'entité E_i

# Pour cet exemple, fixons une synergie S(i,j) pour chaque lien j
# On peut imaginer que l'entité a une forte synergie avec certains liens et faible avec d'autres
# Par exemple : S = [0.8, 0.7, 0.3, 0.2, 0.1]
S = np.array([0.8, 0.7, 0.3, 0.2, 0.1])

# Initialisation de la matrice des poids pour l'entité E_i (une seule ligne)
# On initialise chaque poids à une petite valeur aléatoire proche de 0
np.random.seed(42)
omega = np.random.uniform(0, 0.05, size=n_links)

# Stockage de l'évolution des poids pour visualisation
omega_history = np.zeros((num_iterations + 1, n_links))
omega_history[0, :] = omega.copy()

# Boucle de mise à jour
for t in range(num_iterations):
    # Calcul de la somme des poids pour l'entité E_i
    total_weight = np.sum(omega)

    # Mise à jour pour chaque lien j
    for j in range(n_links):
        # Calcul du terme d'update de base (sans inhibition)
        delta_update = eta * (S[j] - tau * omega[j])
        # Calcul du terme d'inhibition : somme des poids des autres liens (k != j)
        inhibition = gamma * (total_weight - omega[j])
        # Mise à jour additive avec inhibition
        omega[j] = omega[j] + delta_update - inhibition
        # Assurer que le poids reste positif (clipping à 0)
        if omega[j] < 0:
            omega[j] = 0
    # Enregistrer les poids de l'itération actuelle
    omega_history[t + 1, :] = omega.copy()

# Calcul théorique du point fixe pour chaque lien sans inhibition (pour comparaison)
omega_fixed = S / tau

```

```

# Affichage graphique de l'évolution des poids
plt.figure(figsize=(10, 6))
for j in range(n_links):
    plt.plot(omega_history[:, j], marker='o', label=f'$\omega_{i,{j+1}}(t)$')
plt.axhline(y=omega_fixed[0], color='C0', linestyle='--', label=r'$S(1)/\tau$')
plt.axhline(y=omega_fixed[1], color='C1', linestyle='--', label=r'$S(2)/\tau$')
plt.axhline(y=omega_fixed[2], color='C2', linestyle='--', label=r'$S(3)/\tau$')
plt.axhline(y=omega_fixed[3], color='C3', linestyle='--', label=r'$S(4)/\tau$')
plt.axhline(y=omega_fixed[4], color='C4', linestyle='--', label=r'$S(5)/\tau$')
plt.title("Évolution des pondérations $\omega_{i,j}(t)$ avec inhibition compétitive")
plt.xlabel("Itérations $t$")
plt.ylabel("$\omega_{i,j}(t)$")
plt.legend()
plt.grid(True)
plt.show()

```

F. Explications de l'Implémentation

Dans ce script Python :

- **Initialisation**

On fixe les paramètres $\eta = 0.05$, $\tau = 1.0$ et $\gamma = 0.02$. Le nombre de liens (représentant les connexions sortantes d'une entité \mathcal{E}_i) est fixé à 5, avec des valeurs de synergie $S(i,j)$ définies par un vecteur $S = [0.8, 0.7, 0.3, 0.2, 0.1]$. Les poids ω sont initialisés à des valeurs aléatoires proches de 0.

- **Mise à jour**

La mise à jour pour chaque poids suit la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left(S(j) - \tau \omega_{i,j}(t) \right) - \gamma \left(\sum_{k=1}^{n_{links}} \omega_{i,k}(t) - \omega_{i,j}(t) \right).$$

Ici, la somme des poids de toutes les connexions (sauf le lien considéré) est utilisée pour appliquer l'inhibition compétitive, qui réduit la mise à jour si d'autres liens sont déjà forts. Un mécanisme de **clipping** garantit que les poids ne deviennent pas négatifs.

- **Visualisation**

L'évolution des poids est enregistrée dans la matrice *omega_history*, et à la fin, un graphique montre la trajectoire de chaque $\omega_{i,j}(t)$ au fil des itérations. Les lignes horizontales en pointillés indiquent les valeurs théoriques $\frac{S(i,j)}{\tau}$ (sans inhibition) pour référence.

G. Conclusion

Le terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ introduit une **inhibition compétitive** qui force l'entité à concentrer ses ressources sur un nombre limité de liens forts. Ce mécanisme prévient une croissance excessive et favorise la spécialisation des connexions, contribuant ainsi à la formation de clusters plus définis dans le SCN. La mise à jour globale, qui combine un terme d'attraction (provenant de $S(i,j)$) et un terme d'inhibition, se traduit par la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

L'implémentation Python présentée ci-dessus permet de simuler cette dynamique et d'en observer la convergence ou l'émergence de comportements particuliers, tels que la concentration des liens pour certaines connexions au détriment d'autres, phénomène essentiel dans la formation de clusters dans un SCN.

Ce développement, combinant explications mathématiques et implémentation pratique, illustre la manière dont un terme d'inhibition peut être intégré dans la mise à jour des pondérations pour favoriser une **auto-organisation** efficace dans un Deep Synergy Learning.

5.5.2.2. Paramétrage γ

Dans un **Synergistic Connection Network** (SCN), le coefficient γ apparaît dans le terme d'inhibition compétitive et joue un rôle crucial dans la manière dont une entité \mathcal{E}_i répartit son « potentiel de connexion » entre ses divers liens $\omega_{i,k}$. Pour comprendre ce paramétrage, rappelons d'abord la règle de mise à jour additive de base, à laquelle on ajoute ensuite un terme d'inhibition.

A. Rôle conceptuel et formulation mathématique

Sans inhibition, la mise à jour des pondérations dans un SCN s'exprime par la formule additive classique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où :

- $\eta > 0$ est le **taux d'apprentissage**,
- $S(i,j)$ représente la **synergie** (ou similarité) entre les entités \mathcal{E}_i et \mathcal{E}_j ,
- $\tau > 0$ est le paramètre régulant la décroissance linéaire.

Afin d'incorporer une **compétition** entre les différents liens issus de la même entité \mathcal{E}_i , on ajoute un terme d'inhibition qui est proportionnel à la somme des pondérations des autres connexions :

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

ce qui conduit à la formule complète :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Ici, le coefficient γ contrôle l'intensité de cette inhibition compétitive. Un γ élevé signifie que l'entité \mathcal{E}_i est fortement contrainte à limiter simultanément plusieurs connexions fortes. Ainsi, si un lien $\omega_{i,j}$ tend à se renforcer en raison d'une synergie élevée, la somme des autres liens $\sum_{k \neq j} \omega_{i,k}(t)$ va exercer une pression négative qui réduira l'incrément pour $\omega_{i,j}$.

B. Impact sur la dynamique du réseau

Le paramétrage de γ détermine de manière critique la façon dont les ressources de connexion d'une entité se répartissent :

- **Sélectivité accrue** : Un γ suffisamment grand impose une forte compétition, de sorte que l'entité ne pourra renforcer que quelques liens dominants, tandis que les autres resteront faibles. Ce mécanisme favorise la formation de clusters bien définis, dans lesquels chaque entité investit principalement dans un sous-ensemble de connexions.
- **Prévention de la croissance excessive** : Même lorsque plusieurs synergies $S(i,j)$ sont élevées, l'inhibition empêche une croissance simultanée de tous les liens, limitant ainsi le risque d'un emballage global des pondérations.

- **Risques d'oscillations** : Toutefois, un réglage trop agressif de γ peut entraîner des oscillations ou un comportement chaotique, car la compétition latérale devient trop forte. Dans ce cas, les mises à jour peuvent s'alternancer de manière excessive, empêchant une convergence stable vers des valeurs fixes.

En résumé, γ agit comme un **facteur de contrainte** qui module la spécialisation des liens d'une entité et qui, en modulant la pression compétitive, influence directement la formation des clusters dans le réseau.

C. Ajustement empirique et stratégie de régulation

Le choix de γ doit être ajusté en fonction des exigences du problème et du comportement souhaité du SCN. En pratique, plusieurs approches sont envisageables :

- **Essais par paliers** : Il est courant de tester des valeurs telles que $\gamma = 0.01, 0.05, 0.1, 0.2$, etc., et d'observer la dynamique des mises à jour.
- **Adaptation dynamique** : On peut également concevoir une règle d'auto-ajustement où γ évolue en fonction de la somme des pondérations ou d'un indicateur de congestion. Par exemple, on pourrait définir une mise à jour de γ par :

$$\gamma(t+1) = \gamma(t) + \alpha \left[\sum_j \omega_{i,j}(t) - \beta \right],$$

où α et β sont des paramètres ajustables. Ce mécanisme permet de maintenir la somme des connexions autour d'un niveau prédéfini.

- **Normalisation** : Dans certaines applications, il est souhaitable de normaliser la somme des pondérations pour qu'elle reste inférieure à une valeur maximale (par exemple, 1). Ceci peut être réalisé en combinant γ avec d'autres techniques de **clipping** ou de normalisation des poids.

D. Implémentation Python avec Visualisation

Nous proposons ci-dessous une implémentation complète en Python qui simule la dynamique d'inhibition compétitive avec paramétrage de γ . Ce script calcule l'évolution des pondérations $\omega_{i,j}(t)$ pour une entité donnée disposant de plusieurs liens et affiche la trajectoire de chaque poids au cours des itérations.

```
import numpy as np
import matplotlib.pyplot as plt

# Paramètres de la mise à jour
eta = 0.05      # Taux d'apprentissage
tau = 1.0       # Facteur de décroissance
gamma = 0.05    # Coefficient d'inhibition (à ajuster pour observer différents comportements)
num_iterations = 150 # Nombre d'itérations
n_links = 5     # Nombre de liens sortants pour une entité E_i

# Définition des synergies S(i,j) pour chaque lien j
# Pour cet exemple, nous supposons que S(i,j) varie pour simuler différents niveaux de complémentarité
# Exemple : l'entité a une forte synergie avec les deux premiers liens, puis des synergies décroissantes.
S = np.array([0.8, 0.75, 0.4, 0.3, 0.2])

# Initialisation des pondérations omega pour l'entité E_i (un vecteur de taille n_links)
np.random.seed(42)
omega = np.random.uniform(0, 0.05, size=n_links)

# Stockage de l'évolution des pondérations pour la visualisation
omega_history = np.zeros((num_iterations + 1, n_links))
omega_history[0, :] = omega.copy()
```

```

# Simulation de la dynamique de mise à jour avec inhibition compétitive
for t in range(num_iterations):
    # Calcul de la somme totale des poids de l'entité E_i
    total_weight = np.sum(omega)

    # Mise à jour de chaque lien pour l'entité E_i
    for j in range(n_links):
        # Terme d'update additif classique
        delta_update = eta * (S[j] - tau * omega[j])
        # Terme d'inhibition compétitive: on soustrait la somme des poids des autres liens
        inhibition = gamma * (total_weight - omega[j])
        # Mise à jour totale de omega[j]
        omega[j] = omega[j] + delta_update - inhibition
        # Application d'un clipping pour éviter des valeurs négatives
        if omega[j] < 0:
            omega[j] = 0
    # Enregistrement des poids de l'itération courante
    omega_history[t + 1, :] = omega.copy()

# Calcul théorique du point fixe sans inhibition (pour comparaison) : omega* = S / tau
omega_fixed = S / tau

# Affichage graphique de l'évolution des pondérations
plt.figure(figsize=(10, 6))
for j in range(n_links):
    plt.plot(omega_history[:, j], marker='o', linestyle='-', label=f'$\omega_{j+1}(t)$')
for j in range(n_links):
    plt.axhline(y=omega_fixed[j], linestyle='--', color=f'C{j}', label=f'S(j+1)/tau')
plt.title("Évolution des pondérations $\omega_{i,j}(t)$ avec inhibition compétitive ($\gamma = {:.2f}$)".format(gamma))
plt.xlabel("Itérations $t$")
plt.ylabel("$\omega_{i,j}(t)$")
plt.legend(loc="upper right")
plt.grid(True)
plt.show()

```

E. Explications de l'Implémentation

Dans ce script :

- Initialisation et Paramètres :**

Nous fixons les paramètres $\eta = 0.05$, $\tau = 1.0$ et $\gamma = 0.05$. Le nombre de liens ($n_{links} = 5$) correspond aux connexions sortantes d'une entité \mathcal{E}_i . Le vecteur de synergies S est défini pour simuler différents niveaux de complémentarité entre l'entité et ses cibles (par exemple, 0.8, 0.75, 0.4, 0.3, 0.2). Les pondérations ω sont initialisées avec de petites valeurs aléatoires proches de zéro.

- Mise à jour des Pondérations :**

Pour chaque itération, nous calculons d'abord la somme totale des poids pour l'entité. Pour chaque lien j , nous appliquons la mise à jour additive standard $\Delta_{\text{update}} = \eta(S(j) - \tau \omega[j])$ puis soustrayons le terme d'inhibition γ multiplié par la somme des autres poids. Ce terme d'inhibition permet de réduire la capacité de l'entité à renforcer simultanément plusieurs liens. Enfin, nous appliquons un mécanisme de clipping pour garantir que les poids restent positifs.

- **Visualisation :**

La trajectoire de chaque pondération $\omega_j(t)$ est enregistrée et affichée sur un graphique. Des lignes horizontales en pointillés indiquent les valeurs théoriques $\frac{S(j)}{\tau}$ pour référence, ce qui permet de comparer la convergence effective avec le point fixe attendu en l'absence d'inhibition.

F. Conclusion

L'ajout du terme d'inhibition $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ permet d'instaurer une **compétition** entre les liens d'une même entité, ce qui force cette dernière à concentrer ses ressources sur un nombre limité de connexions fortes. Ce mécanisme aide à la formation de **clusters** plus distincts et empêche la croissance simultanée de tous les liens. La formule de mise à jour complète :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t)$$

est ainsi mise en œuvre de manière simple en Python, et la simulation graphique permet d'observer l'effet de l'inhibition compétitive sur la dynamique des pondérations. Ce développement illustre l'importance du paramètre γ et montre comment, en ajustant sa valeur, on peut influencer la spécialisation et la stabilité des connexions dans un SCN.

5.5.2.3. Application concrète : limiter la somme des liens sortants par entité

Dans de nombreux scénarios d'application d'un **Synergistic Connection Network** (SCN), il est essentiel que chaque entité \mathcal{E}_i ne se lie pas de manière excessive à l'ensemble des autres entités. En effet, dans des systèmes inspirés par le fonctionnement des réseaux neuronaux ou par des modèles de ressources limitées en systèmes cognitifs, il est souvent souhaitable de borner la somme des pondérations associées aux liens sortants de \mathcal{E}_i afin de forcer cette entité à "choisir" les partenaires les plus pertinents. Le mécanisme d'inhibition compétitive, introduit sous la forme du terme

$$-\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

permet justement de réaliser cette contrainte. En ajoutant ce terme dans la règle de mise à jour, la dynamique de $\omega_{i,j}$ se voit modifiée de sorte que la croissance d'un lien est freinée par la présence d'autres liens déjà établis.

A. Principe et justification

La règle de mise à jour additive classique sans inhibition s'exprime par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η est le taux d'apprentissage et τ le paramètre de décroissance. L'ajout d'un terme d'inhibition introduit une compétition entre les liens sortants d'une même entité. La formule devient alors

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Le terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ agit comme un prélèvement ou une pénalité sur chaque lien $\omega_{i,j}$ en fonction de la somme des autres liens sortants de \mathcal{E}_i . On peut ainsi interpréter γ comme le **coefficent d'inhibition**, qui définit le budget de connexion de l'entité. Lorsque la somme des autres liens est importante, l'inhibition est plus forte, ce qui freine davantage l'augmentation de $\omega_{i,j}$.

Ce mécanisme de limitation permet d'éviter que l'entité répartisse ses ressources de manière uniforme sur l'ensemble des connexions, favorisant ainsi une spécialisation où seules quelques connexions significatives émergent, tandis que les autres restent faibles. D'un point de vue neuro-inspiré, cela correspond à l'idée d'**inhibition latérale**, dans laquelle un neurone, une fois fortement activé, limite l'activation simultanée de ses voisins. Cette analogie trouve également

un écho dans les modèles économiques où une taxe ou un coût additionnel est appliqué lorsque les ressources sont réparties sur trop de partenariats.

B. Étude mathématique

Afin d'analyser l'effet de ce mécanisme sur la somme totale des liens sortants, nous notons

$$\Sigma_i(t) = \sum_j \omega_{i,j}(t).$$

En sommant la règle de mise à jour sur tous les j pour une entité \mathcal{E}_i , on obtient :

$$\Sigma_i(t+1) = \sum_j \omega_{i,j}(t) + \eta \sum_j [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_j \sum_{k \neq j} \omega_{i,k}(t).$$

Le double-somme $\sum_j \sum_{k \neq j} \omega_{i,k}(t)$ se simplifie en $(n_i - 1) \Sigma_i(t)$, où n_i est le nombre de liens sortants de l'entité \mathcal{E}_i . Ainsi, l'inhibition introduit un terme proportionnel à $\Sigma_i(t)$ lui-même, ce qui tend à stabiliser la somme totale des liens, empêchant ainsi une croissance incontrôlée.

C. Implémentation Python avec Visualisation

Nous proposons ci-dessous une implémentation en Python qui simule la dynamique de mise à jour avec inhibition compétitive pour une entité \mathcal{E}_i disposant de plusieurs liens. L'exemple permet de visualiser comment la somme des liens se régule et comment les différents liens évoluent au fil des itérations.

```
import numpy as np
import matplotlib.pyplot as plt

# Paramètres de la simulation
eta = 0.05      # Taux d'apprentissage
tau = 1.0       # Facteur de décroissance
gamma = 0.1     # Coefficient d'inhibition (à ajuster pour voir l'effet sur la somme)
num_iterations = 150 # Nombre total d'itérations
n_links = 10    # Nombre de liens sortants pour une entité E_i

# Définir les synergies S(i,j) pour chaque lien j.
# Ici, S est un vecteur de valeurs simulant différentes "forces" de synergie.
# On suppose que certains liens ont une synergie élevée et d'autres faible.
S = np.array([0.8, 0.75, 0.6, 0.55, 0.5, 0.4, 0.35, 0.3, 0.25, 0.2])

# Initialisation des pondérations omega pour l'entité E_i avec de faibles valeurs aléatoires
np.random.seed(42)
omega = np.random.uniform(0, 0.05, size=n_links)

# Stocker l'évolution des pondérations pour visualisation
omega_history = np.zeros((num_iterations + 1, n_links))
sum_history = np.zeros(num_iterations + 1)
omega_history[0, :] = omega.copy()
sum_history[0] = np.sum(omega)

# Simulation de la mise à jour avec inhibition compétitive
for t in range(num_iterations):
    # Calcul de la somme des pondérations de l'entité E_i à l'itération t
    total_weight = np.sum(omega)

    # Mise à jour de chaque lien pour l'entité E_i
    for j in range(n_links):
```

```

# Terme d'update additif classique
delta_update = eta * (S[j] - tau * omega[j])
# Terme d'inhibition compétitive : somme des autres liens
inhibition = gamma * (total_weight - omega[j])
# Mise à jour totale
omega[j] = omega[j] + delta_update - inhibition
# Clipping : s'assurer que les pondérations restent positives
if omega[j] < 0:
    omega[j] = 0

# Enregistrement des pondérations et de la somme totale pour la visualisation
omega_history[t + 1, :] = omega.copy()
sum_history[t + 1] = np.sum(omega)

# Calcul théorique du point fixe pour chaque lien sans inhibition (pour comparaison) : omega* = S / tau
omega_fixed = S / tau

# Affichage graphique de l'évolution de chaque pondération
plt.figure(figsize=(12, 6))
for j in range(n_links):
    plt.plot(omega_history[:, j], marker='o', linestyle='-', label=f'$\omega_{j+1}(t)$')
for j in range(n_links):
    plt.axhline(y=omega_fixed[j], linestyle='--', color=f'C{j}', label=f'$\omega_{j+1}^* / \tau$')

plt.title("Évolution des pondérations $\omega_{i,j}(t)$ avec inhibition compétitive ($\gamma = {:.2f}$)".format(gamma))
plt.xlabel("Itérations $t$")
plt.ylabel("$\omega_{i,j}(t)$")
plt.legend(loc="upper right", fontsize=8)
plt.grid(True)
plt.show()

# Affichage graphique de l'évolution de la somme des pondérations par entité
plt.figure(figsize=(10, 5))
plt.plot(sum_history, marker='s', linestyle='-', color='purple', label='$\Sigma_i(t)$')
plt.title("Évolution de la somme des liens sortants $\Sigma_i(t)$")
plt.xlabel("Itérations $t$")
plt.ylabel("$\Sigma_i(t)$")
plt.legend()
plt.grid(True)
plt.show()

```

D. Explications détaillées de l'implémentation

Dans cet exemple, nous modélisons une entité \mathcal{E}_i dotée de $n_{links} = 10$ connexions sortantes, chacune caractérisée par une **synergie** $S(i, j)$ fixée. Les pondérations $\omega_{i,j}(t)$ sont initialisées avec de faibles valeurs aléatoires pour simuler un départ quasi nul.

La mise à jour des pondérations s'effectue en deux parties. D'une part, le terme additive classique, $\eta [S(i, j) - \tau \omega_{i,j}(t)]$, tend à faire converger chaque lien vers la valeur $\frac{S(i, j)}{\tau}$ en l'absence d'inhibition. D'autre part, le terme d'inhibition, $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$, représente le prélèvement effectué sur le lien $\omega_{i,j}$ en fonction de la somme des autres pondérations. Ce mécanisme force l'entité à limiter la somme totale de ses connexions, conduisant à une répartition sélective des ressources.

Le script enregistre, à chaque itération, l'évolution de chaque pondération ainsi que la somme totale $\Sigma_i(t) = \sum_j \omega_{i,j}(t)$. Ces données sont ensuite visualisées par deux graphiques : l'un montre l'évolution individuelle de chaque $\omega_{i,j}$

(avec des lignes horizontales indiquant les valeurs théoriques sans inhibition), et l'autre trace l'évolution de la somme totale des liens de l'entité. Ainsi, on peut observer comment le mécanisme d'inhibition stabilise la somme des pondérations et favorise la spécialisation.

E. Conclusion

L'introduction du terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ dans la mise à jour permet d'instaurer une compétition interne entre les liens sortants d'une entité. Ce mécanisme de **budget** impose que l'entité ne puisse renforcer simultanément tous ses liens, favorisant ainsi la formation de clusters denses et la spécialisation des connexions. L'implémentation Python présentée illustre concrètement cette dynamique en montrant, à travers des graphiques, la convergence des pondérations et la régulation de leur somme. Le réglage approprié de γ est crucial pour obtenir un équilibre entre l'amplification des liens forts et le freinage des liens faibles, garantissant ainsi une auto-organisation efficace et cohérente dans le SCN.

Ce développement, associant explications mathématiques et implémentation pratique, fournit une base solide pour comprendre et exploiter le mécanisme d'inhibition compétitive dans un cadre de Deep Synergy Learning.

5.5.3. Saturation

La **saturation** constitue un autre mécanisme majeur pour contrôler la croissance des pondérations $\omega_{i,j}$ dans un SCN. Même si l'on dispose d'une dynamique de base (section 5.5.1) et d'un module d'inhibition/contrôle (section 5.5.2), il subsiste fréquemment un risque d'emballlement lorsque plusieurs liens $\omega_{i,j}$ se renforcent simultanément ou lorsqu'une seule connexion prend une ampleur disproportionnée. La **saturation** permet alors de **clorer** la valeur d'un lien au-delà d'un certain plafond ω_{\max} .

D'un point de vue **mathématique**, la saturation introduit une **barrière** supérieure : même si l'itération calcule une mise à jour positive qui pousserait $\omega_{i,j}$ à dépasser ω_{\max} , on la "coupe" (clipping) pour éviter qu'elle ne franchisse ce seuil. Dans cette section (5.5.3), nous décrirons le **clipping** (5.5.3.1) et son impact, puis nous analyserons comment choisir ω_{\max} et en quoi cela influe sur la structure de clusters (5.5.3.2).

5.5.3.1. Clipping : $\omega_{i,j} \leftarrow \min(\omega_{i,j}, \omega_{\max})$

Dans un **Synergistic Connection Network** (SCN), les mécanismes d'inhibition compétitive (voir notamment la section 5.5.2.2) permettent de réguler la croissance collective des liens sortants d'une entité \mathcal{E}_i . Néanmoins, il reste possible qu'un ou plusieurs liens, par leur dynamique interne ou en raison de paramètres mal ajustés (par exemple, un taux d'apprentissage η trop élevé ou une décroissance τ trop faible), atteignent des valeurs excessives. Pour éviter cette situation, on introduit le **clipping**, qui consiste à borner chaque pondération individuellement en imposant que :

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

A. Principe du Clipping

Le **clipping** agit comme un opérateur de **saturation** appliqué en post-traitement à la mise à jour des poids. La formule de base d'une mise à jour additive dans un SCN s'exprime généralement par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Lorsque cette règle aboutit à une valeur de $\omega_{i,j}(t+1)$ supérieure à un seuil prédéfini ω_{\max} , le clipping intervient en forçant :

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Ce procédé garantit que, quelle que soit la synergie $S(i,j)$ ou les conditions locales de mise à jour, aucune connexion ne pourra dépasser la valeur ω_{\max} . Cette opération est particulièrement pertinente pour éviter une croissance

incontrôlée des poids, phénomène pouvant être observé lorsque des paramètres de mise à jour conduisent à une dynamique exponentielle ou à des oscillations.

B. Rôle Mathématique et Effets sur la Convergence

D'un point de vue **mathématique**, le clipping agit comme une contrainte non linéaire dans le système. Sans clipping, le point fixe théorique pour chaque lien, en l'absence d'inhibition supplémentaire, serait donné par

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Cependant, lorsque $\frac{S(i,j)}{\tau}$ dépasse la valeur ω_{\max} , le clipping force le lien à se stabiliser à ω_{\max} plutôt que de continuer à croître. En d'autres termes, le système évolue vers un "point fixe saturé". On obtient ainsi une dynamique par morceaux où la mise à jour se déroule selon la règle linéaire jusqu'à ce que $\omega_{i,j}$ atteigne ω_{\max} , puis la valeur est fixée.

Cette approche présente l'avantage de prévenir l'emballage de certaines pondérations, tout en permettant aux autres liens de converger vers leurs valeurs théoriques lorsque celles-ci sont inférieures à ω_{\max} . En conséquence, le SCN est amené à concentrer ses ressources sur un nombre limité de liens forts, favorisant la formation de clusters plus distincts et économiquement répartis, sans pour autant perdre l'information sur la synergie.

C. Implémentation en Python avec Graphiques

Nous présentons ci-dessous un code Python complet illustrant la dynamique d'un SCN avec mise à jour additive, incluant l'inhibition compétitive et l'opération de clipping. Ce script simule la mise à jour de la pondération d'un lien et trace son évolution au fil des itérations, ainsi que l'évolution de la somme des liens sortants pour une entité donnée.

```
import numpy as np
import matplotlib.pyplot as plt

# Paramètres de la simulation
eta = 0.05      # Taux d'apprentissage
tau = 1.0       # Facteur de décroissance
gamma = 0.1     # Coefficient d'inhibition compétitive
omega_max = 0.7 # Valeur de clipping maximale pour chaque lien
num_iterations = 150 # Nombre d'itérations
n_links = 10    # Nombre de liens sortants pour une entité E_i

# Définition des synergies S(i,j) pour l'entité E_i (vecteur de synergies pour chaque lien)
# Supposons que les synergies varient pour illustrer des cas où certains liens devraient théoriquement dépasser omega_max.
S = np.array([0.8, 0.75, 0.6, 0.55, 0.5, 0.4, 0.35, 0.3, 0.25, 0.2])

# Initialisation des pondérations omega pour l'entité E_i avec de faibles valeurs aléatoires
np.random.seed(42)
omega = np.random.uniform(0, 0.05, size=n_links)

# Stockage de l'évolution des pondérations pour visualisation
omega_history = np.zeros((num_iterations + 1, n_links))
sum_history = np.zeros(num_iterations + 1)
omega_history[0, :] = omega.copy()
sum_history[0] = np.sum(omega)

# Simulation de la mise à jour avec inhibition compétitive et clipping
for t in range(num_iterations):
    total_weight = np.sum(omega) # Somme des pondérations pour l'entité E_i à l'itération t
    for j in range(n_links):
        # Calcul du terme de mise à jour additive
```

```

delta_update = eta * (S[j] - tau * omega[j])
# Calcul du terme d'inhibition compétitive : somme des autres liens (pour k != j)
inhibition = gamma * (total_weight - omega[j])
# Mise à jour additive avec inhibition
new_omega = omega[j] + delta_update - inhibition
# Application du clipping pour s'assurer que new_omega ne dépasse pas omega_max
new_omega = min(new_omega, omega_max)
# Garantie de non-négativité
omega[j] = max(new_omega, 0)
# Enregistrement de l'évolution des pondérations et de la somme totale
omega_history[t + 1, :] = omega.copy()
sum_history[t + 1] = np.sum(omega)

# Calcul théorique du point fixe sans clipping pour comparaison (omega* = S / tau)
omega_fixed = S / tau

# Affichage graphique de l'évolution de chaque pondération
plt.figure(figsize=(12, 6))
for j in range(n_links):
    plt.plot(omega_history[:, j], marker='o', linestyle='-', label=f'$\omega_{i,j}(t)$')
    # Ajout d'une ligne horizontale pour la valeur théorique sans clipping, seulement si elle est inférieure à omega_max
    if omega_fixed[j] <= omega_max:
        plt.axhline(y=omega_fixed[j], linestyle='--', color=f'C{j}', label=f'S({j+1})/\tau')

plt.title("Évolution des pondérations $\omega_{i,j}(t)$ avec inhibition compétitive et clipping")
plt.xlabel("Itérations $t$")
plt.ylabel("$\omega_{i,j}(t)$")
plt.legend(loc="upper right", fontsize=8)
plt.grid(True)
plt.show()

# Affichage graphique de l'évolution de la somme des pondérations pour l'entité E_i
plt.figure(figsize=(10, 5))
plt.plot(sum_history, marker='s', linestyle='-', color='purple', label="$\Sigma_i(t)$")
plt.title("Évolution de la somme des liens sortants $\Sigma_i(t)$")
plt.xlabel("Itérations $t$")
plt.ylabel("$\Sigma_i(t)$")
plt.legend()
plt.grid(True)
plt.show()

```

D. Explications Complètes

Dans cet exemple, nous simulons la dynamique d'un lien sortant d'une entité E_i possédant $n_{links} = 10$ connexions. Chaque lien $\omega_{i,j}(t)$ évolue selon une règle additive classique à laquelle on ajoute un terme d'inhibition compétitive et, enfin, une opération de clipping. Plus précisément :

- **Mise à jour additive :**

Chaque pondération est mise à jour en ajoutant le terme $\eta [S(j) - \tau \omega_{i,j}(t)]$. Ce terme tend à rapprocher $\omega_{i,j}$ de la valeur théorique $S(j)/\tau$.

- **Inhibition compétitive :**

La somme $\sum_{k \neq j} \omega_{i,k}(t)$ est calculée pour l'entité, et le terme $-\gamma \times (\text{total des autres liens})$ est soustrait à la mise à jour de chaque lien. Ainsi, si d'autres liens sont déjà élevés, l'augmentation de $\omega_{i,j}$ sera freinée.

- **Clipping :**

Après avoir appliqué la mise à jour, nous utilisons l'opérateur $\min(\cdot, \omega_{\max})$ pour garantir que la valeur de $\omega_{i,j}(t+1)$ ne dépasse jamais le seuil fixé ω_{\max} . Ce mécanisme empêche toute croissance excessive, même si la synergie $S(j)$ et les autres paramètres favoriseraient une augmentation trop importante.

- **Visualisation :**

Nous traçons deux graphiques : le premier montre l'évolution individuelle de chaque lien $\omega_{i,j}(t)$ au fil des itérations, avec en point de référence les valeurs théoriques $S(j)/\tau$ lorsque celles-ci sont inférieures à ω_{\max} . Le second graphique trace l'évolution de la somme totale des liens sortants $\Sigma_i(t)$ pour l'entité, permettant de vérifier l'effet du mécanisme de "budget" sur la répartition globale des ressources.

E. Conclusion

L'opération de **clipping** constitue un mécanisme essentiel pour stabiliser la dynamique des pondérations dans un SCN, en imposant une borne supérieure ω_{\max} à chaque lien. Ce procédé, combiné avec l'inhibition compétitive, permet de contrôler la répartition des ressources de connexion d'une entité, empêchant ainsi l'emballage des poids et favorisant la formation de clusters bien définis. L'exemple en Python présenté ici, avec ses graphiques, illustre concrètement comment la mise à jour de $\omega_{i,j}$ converge vers des valeurs régulées et comment la somme totale des liens se stabilise sous l'effet de ces mécanismes.

Ce développement, intégrant explications mathématiques et implémentation pratique, fournit une base solide pour comprendre et exploiter le mécanisme de clipping dans le cadre du Deep Synergy Learning.

5.5.3.2. Réglage de ω_{\max} et son effet sur l'émergence de clusters

Lorsqu'on introduit un **clipping** dans la mise à jour des pondérations, nous imposons la contrainte suivante :

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Ce procédé, bien qu'il puisse paraître comme un simple ajustement, a un impact majeur sur la **dynamique** du SCN et la qualité des **clusters** qui émergent. En effet, le choix de la valeur ω_{\max} détermine dans quelle mesure chaque lien $\omega_{i,j}$ pourra se développer, et par conséquent comment les entités \mathcal{E}_i vont concentrer leurs ressources de connexion.

A. Importance Fondamentale du Choix de ω_{\max}

Le mécanisme de clipping agit localement sur chaque connexion en contraignant la valeur maximale atteinte par $\omega_{i,j}$. Formellement, dans la mise à jour additive classique, nous avons :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Lorsque le résultat de cette mise à jour dépasse ω_{\max} , l'opérateur de clipping force :

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Ainsi, le point fixe théorique sans clipping, qui serait $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$, est remplacé par :

$$\omega_{i,j}^* = \min\left(\frac{S(i,j)}{\tau}, \omega_{\max}\right).$$

Ce réglage permet de moduler la **sélectivité** du réseau : un ω_{\max} faible restreint la force maximale de chaque lien, favorisant une répartition plus uniforme, tandis qu'un plafond élevé permet à certains liens de devenir très forts et de se démarquer, ce qui peut conduire à des clusters hiérarchisés.

B. Impact sur la Structure de Clusters

Du point de vue de la formation de clusters, le choix de ω_{\max} influence directement la topologie du réseau :

- Si ω_{\max} est trop bas, même les liens qui devraient naturellement devenir forts seront limités. Cela peut conduire à une homogénéisation des connexions où la différenciation entre les liens forts et faibles est atténuée. Dans ce cas, le SCN aura tendance à former des groupes moins contrastés, car toutes les connexions atteignent presque le même niveau maximal.
- Inversement, un ω_{\max} élevé permet à certains liens de se développer pleinement selon leur synergie intrinsèque $S(i,j)$. Si plusieurs liens sont en compétition, il est alors possible qu'un ou quelques liens dominent l'ensemble des connexions sortantes d'une entité, menant à une formation de clusters très marqués, dans lesquels certains liens sont beaucoup plus forts que d'autres.

Ainsi, la valeur de ω_{\max} se révèle être un paramètre de **régulation** crucial. En le combinant avec d'autres mécanismes tels que l'inhibition (qui limite la somme totale des connexions d'un nœud), on obtient un contrôle fin sur la manière dont les entités sélectionnent leurs partenaires privilégiés dans le réseau.

C. Ajustement Empirique et Adaptatif

Dans la pratique, le choix de ω_{\max} peut être réalisé de façon empirique. Par exemple, si l'on connaît l'ordre de grandeur des valeurs de $S(i,j)/\tau$, il est judicieux de fixer ω_{\max} en fonction de ces valeurs. Une approche consiste à définir

$$\omega_{\max} = \alpha \cdot \max_{i,j} \left\{ \frac{S(i,j)}{\tau} \right\},$$

où $\alpha \geq 1$ est un facteur multiplicatif qui permet de ne pas trop contraindre la dynamique lorsque certaines synergies sont très élevées. Dans certains scénarios, il est également envisageable d'adapter dynamiquement ω_{\max} au fil des itérations (similaire à un recuit simulé), afin de permettre une phase exploratoire initiale suivie d'une stabilisation plus stricte.

D. Implémentation Python Complète avec Graphiques

Nous présentons ci-dessous un exemple complet en Python qui illustre l'effet du paramétrage de ω_{\max} sur la dynamique d'un SCN. Ce script simule la mise à jour d'une matrice de pondérations pour une entité possédant plusieurs liens, intègre le clipping, et affiche l'évolution des pondérations au fil des itérations ainsi que la répartition finale sous forme de heatmap.

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Paramètres de la simulation
eta = 0.05      # Taux d'apprentissage
tau = 1.0        # Facteur de décroissance
gamma = 0.1      # Coefficient d'inhibition compétitive
num_iterations = 150 # Nombre d'itérations
n_links = 10     # Nombre de liens sortants pour une entité

# Paramètre de clipping : on va comparer différents réglages
omega_max_values = [0.5, 1.0, 2.0]

# Définition des synergies S(i,j) pour l'entité E_i
# Pour illustrer, nous fixons des valeurs différentes pour chaque lien.

```

```

S = np.linspace(0.8, 0.2, n_links) # Synergies décroissantes de 0.8 à 0.2

# Fonction de mise à jour avec inhibition et clipping
def simulate_update(omega_max, eta, tau, gamma, num_iterations, S):
    # Initialisation des pondérations avec de faibles valeurs aléatoires
    np.random.seed(42)
    omega = np.random.uniform(0, 0.05, size=n_links)
    omega_history = np.zeros((num_iterations + 1, n_links))
    omega_history[0, :] = omega.copy()

    # Simulation de la mise à jour sur num_iterations itérations
    for t in range(num_iterations):
        total_weight = np.sum(omega) # Somme des pondérations pour l'entité E_i à l'itération t
        for j in range(n_links):
            # Calcul de la mise à jour additive de base
            delta_update = eta * (S[j] - tau * omega[j])
            # Calcul du terme d'inhibition : somme des autres liens pour l'entité E_i
            inhibition = gamma * (total_weight - omega[j])
            # Nouvelle pondération avant clipping
            new_omega = omega[j] + delta_update - inhibition
            # Application du clipping : la nouvelle valeur ne doit pas dépasser omega_max
            new_omega = min(new_omega, omega_max)
            # On garantit que la pondération reste positive
            omega[j] = max(new_omega, 0)
        omega_history[t + 1, :] = omega.copy()

    return omega_history

# Simulation pour chaque valeur de omega_max
histories = {}
for omega_max in omega_max_values:
    histories[omega_max] = simulate_update(omega_max, eta, tau, gamma, num_iterations, S)

# Affichage des trajectoires pour chaque réglage de omega_max
plt.figure(figsize=(12, 8))
for omega_max, history in histories.items():
    for j in range(n_links):
        plt.plot(history[:, j], label=f'Lien {j+1}, ω_max={omega_max}', alpha=0.7)
    plt.title(f'Évolution des pondérations pour ω_max = {omega_max}')
    plt.xlabel("Itérations")
    plt.ylabel("Pondération ω")
    plt.legend(fontsize=8, loc="upper right")
    plt.grid(True)
    plt.show()

# Affichage d'une heatmap finale pour comparer les distributions de pondérations
plt.figure(figsize=(14, 4))
for idx, omega_max in enumerate(omega_max_values):
    final_weights = histories[omega_max][-1, :].reshape(1, -1)
    plt.subplot(1, len(omega_max_values), idx+1)
    sns.heatmap(final_weights, annot=True, cmap="viridis", cbar=True, vmin=0, vmax=max(omega_max_values))
    plt.title(f'ω_max = {omega_max}')
    plt.xlabel("Lien index")
    plt.ylabel("Entité E_i")
plt.tight_layout()
plt.show()

```

F. Explications Détaillées

Dans cet exemple, nous considérons une entité \mathcal{E}_i qui possède $n_{\text{links}} = 10$ connexions, chacune ayant une synergie fixée $S(j)$ variant linéairement de 0.8 à 0.2. La mise à jour de chaque lien suit l'équation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Après avoir calculé la mise à jour, nous appliquons le **clipping** :

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Nous comparons trois valeurs de ω_{\max} : 0.5, 1.0 et 2.0. La fonction *simulate_update* effectue la mise à jour sur un nombre fixé d'itérations et retourne l'historique des pondérations pour chaque lien. Ensuite, nous traçons, pour chaque réglage, l'évolution des valeurs de $\omega_{i,j}(t)$ ainsi qu'une heatmap finale montrant la distribution des pondérations après la convergence.

L'analyse graphique permet de constater que :

- Pour un ω_{\max} faible (par exemple 0.5), les pondérations sont strictement limitées, ce qui peut conduire à une uniformisation des liens et à une perte de différenciation dans la formation de clusters.
- Pour un ω_{\max} plus élevé (par exemple 1.0 ou 2.0), les liens qui bénéficient d'une synergie forte peuvent atteindre des valeurs proches de leur point fixe théorique (sous réserve de l'effet inhibiteur), permettant ainsi la mise en évidence de clusters plus marqués et hiérarchisés.

G. Conclusion

Le paramétrage de ω_{\max} est crucial pour contrôler la dynamique des pondérations dans un SCN. Le clipping, en imposant une saturation locale sur chaque lien, agit comme une contrainte de ressources permettant de limiter l'emballlement de certains poids et de favoriser une spécialisation des connexions. En ajustant ω_{\max} et en combinant cet effet avec l'inhibition compétitive, on module la formation des clusters et on contrôle la répartition des ressources de connexion. L'implémentation Python présentée ici, avec ses visualisations, offre un exemple concret de l'impact du réglage de ω_{\max} sur l'émergence des structures dans un SCN, fournissant ainsi un outil précieux pour l'expérimentation et l'analyse en Deep Synergy Learning.

Cette approche intégrée, alliant explications mathématiques détaillées et implémentation pratique, permet d'illustrer la puissance du clipping dans la stabilisation de la dynamique et dans la formation de clusters au sein d'un SCN.

5.5.4. Recuit Simulé ou Bruit

En plus des mécanismes d'inhibition (section 5.5.2) et de saturation (section 5.5.3), un autre procédé couramment employé dans la mise à jour des pondérations $\omega_{i,j}$ est l'**injection de bruit** ou l'utilisation d'une **méthode de recuit simulé**. Ces approches visent à **perturber** périodiquement le système pour **éviter** qu'il ne se fige trop tôt dans un minimum local, ou pour mieux explorer l'espace des configurations possibles. Sur le plan mathématique, elles introduisent un **terme stochastique** ou un **paramètre de "température"** qui se modifie au fil des itérations.

5.5.3.2. Réglage de ω_{\max} et son effet sur l'émergence de clusters

Dans un **Synergistic Connection Network** (SCN), la mise à jour des pondérations $\omega_{i,j}$ se fait typiquement selon une règle additive telle que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le taux d'apprentissage et $\tau > 0$ représente le coefficient de décroissance. Afin d'empêcher qu'un ou plusieurs liens ne deviennent excessivement forts — situation qui pourrait fausser la dynamique du réseau et masquer la spécialisation souhaitée —, on introduit un **mécanisme de clipping**. Ce procédé consiste à contraindre chaque valeur $\omega_{i,j}(t+1)$ par la relation

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}),$$

ce qui force chaque connexion à ne pas dépasser un plafond fixé, ω_{\max} . Sur le plan mathématique, en l'absence de clipping le point fixe théorique pour la mise à jour additive serait

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau}.$$

Or, dans le cas où $\frac{S(i,j)}{\tau} > \omega_{\max}$, la dynamique est « coupée » et le lien se stabilise à ω_{\max} . Cette contrainte a plusieurs conséquences sur la formation des clusters au sein du SCN :

- **Uniformisation versus différenciation** : Si ω_{\max} est trop faible, même les liens qui devraient être forts seront plafonnés à une valeur inférieure, aboutissant à une homogénéisation des connexions et à des clusters moins différenciés.
- **Hiérarchisation des liens** : Un ω_{\max} élevé permet à certains liens de se développer pleinement (selon leur synergie intrinsèque $S(i,j)$) tandis que d'autres restent faibles. On obtient alors une structure de clusters plus contrastée, avec des liens dominants entre certaines entités qui se distinguent nettement des connexions moins significatives.
- **Interaction avec l'inhibition** : Le clipping agit sur chaque lien individuellement, alors que l'inhibition (par exemple, un terme du type $-\gamma \sum_{k \neq j} \omega_{i,k}$) régule la somme des connexions sortantes d'une entité. Ensemble, ces deux mécanismes permettent à l'entité de concentrer son « budget de connexion » sur quelques partenaires privilégiés.

La détermination de ω_{\max} est donc cruciale : elle doit être choisie en fonction de l'échelle des synergies $S(i,j)$ et du coefficient τ pour permettre une discrimination effective entre les connexions. Dans certains cas, il est même envisageable de faire évoluer ω_{\max} au fil des itérations (similaire à un recuit simulé) afin de favoriser une phase d'exploration initiale suivie d'une consolidation des clusters.

Implémentation Python avec Visualisations

Nous proposons ci-dessous un exemple complet en Python. Ce script simule un SCN où n entités sont organisées en clusters grâce à une matrice de synergie \mathbf{S} dotée d'une structure en blocs. La mise à jour des pondérations suit la règle additive avec clipping :

$$\omega_{i,j}(t+1) = \min(\omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)], \omega_{\max}).$$

Nous étudierons l'impact de différents réglages de ω_{\max} sur la formation des clusters.

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# -----
# Paramètres de la simulation
# -----
np.random.seed(42)

n = 50          # nombre total d'entités
```

```

n_clusters = 3          # nombre de clusters souhaités
eta = 0.05              # taux d'apprentissage
tau = 1.0                # coefficient de décroissance
num_iterations = 200      # nombre d'itérations
# On va tester différents plafonds
omega_max_values = [0.3, 0.7, 1.5]

# -----
# Création d'une matrice de synergie S
# -----
# On suppose que les entités sont réparties en n_clusters,
# avec une forte synergie intra-cluster et une faible synergie inter-cluster.

# Attribution aléatoire de clusters
cluster_labels = np.random.choice(n_clusters, n)

# Initialisation de la matrice S
S = np.zeros((n, n))
for i in range(n):
    for j in range(n):
        if cluster_labels[i] == cluster_labels[j]:
            # Synergie élevée pour les entités dans le même cluster
            S[i, j] = 0.8
        else:
            # Synergie faible pour les entités dans des clusters différents
            S[i, j] = 0.2
# On s'assure que la diagonale est nulle (pas de self-connexion)
np.fill_diagonal(S, 0)

# -----
# Fonction de simulation avec clipping
# -----
def simulate_scn(omega_max, eta, tau, num_iterations, S):
    n = S.shape[0]
    # Initialisation des pondérations avec un bruit faible
    omega = np.random.uniform(0, 0.05, (n, n))
    np.fill_diagonal(omega, 0)
    omega_history = np.zeros((num_iterations+1, n, n))
    omega_history[0] = omega.copy()

    # Mise à jour itérative
    for t in range(num_iterations):
        # Calcul de la nouvelle matrice selon la règle additive
        omega_new = omega + eta * (S - tau * omega)
        # Application du clipping
        omega_new = np.minimum(omega_new, omega_max)
        # S'assurer que les self-connections restent nulles
        np.fill_diagonal(omega_new, 0)
        # Mettre à jour pour la prochaine itération
        omega = omega_new.copy()
        omega_history[t+1] = omega.copy()

    return omega_history

# -----
# Exécution de la simulation pour différents omega_max

```

```

# -----
results = {}
for omega_max in omega_max_values:
    results[omega_max] = simulate_scn(omega_max, eta, tau, num_iterations, S)

# -----
# Visualisation des trajectoires moyennes des pondérations
# -----
plt.figure(figsize=(12, 8))
for omega_max in omega_max_values:
    # Calcul de la moyenne des pondérations hors diagonale pour chaque itération
    mean_values = [np.mean(omega_history[np.triu_indices_from(omega_history, k=1)])]
    for omega_history in results[omega_max]:
        plt.plot(mean_values, label=f"\omega_max = {omega_max}")
    plt.xlabel("Itérations")
    plt.ylabel("Pondération moyenne (hors diagonale)")
    plt.title("Évolution de la pondération moyenne selon \omega_max")
    plt.legend()
    plt.grid(True)
    plt.show()

# -----
# Visualisation finale : Heatmaps des matrices \omega
# -----
plt.figure(figsize=(18, 5))
for idx, omega_max in enumerate(omega_max_values):
    final_omega = results[omega_max][-1]
    plt.subplot(1, len(omega_max_values), idx+1)
    sns.heatmap(final_omega, cmap="viridis", vmin=0, vmax=max(omega_max_values), annot=False)
    plt.title(f"Matrice \omega finale (\omega_max = {omega_max})")
    plt.xlabel("Index j")
    plt.ylabel("Index i")
plt.tight_layout()
plt.show()

```

Explanations

A. Principe du Clipping et Réglage de ω_{\max}

La formule de mise à jour initiale est donnée par

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Pour éviter que certains liens ne deviennent trop forts, nous appliquons ensuite le **clipping** :

$$\omega_{i,j}(t + 1) \leftarrow \min(\omega_{i,j}(t + 1), \omega_{\max}).$$

Cette contrainte force chaque connexion à être bornée, ce qui, en fonction de la valeur choisie pour ω_{\max} , influencera la capacité des liens à atteindre leur point fixe théorique $\frac{S(i,j)}{\tau}$. Si $\frac{S(i,j)}{\tau}$ excède ω_{\max} , alors la croissance est limitée par cette borne.

B. Impact sur l'Émergence de Clusters

Dans notre simulation, la matrice de synergie S est construite de manière à favoriser une forte cohésion au sein d'un même cluster (valeur de 0.8) et une faible cohésion entre des entités de clusters différents (valeur de 0.2). Le clipping intervient ensuite pour influencer la distribution finale des poids $\omega_{i,j}$.

- Un ω_{\max} faible force toutes les pondérations à rester en dessous d'un seuil strict, ce qui peut engendrer des clusters moins différenciés.
- Un ω_{\max} élevé permet aux liens d'exprimer pleinement leurs différences, conduisant à une hiérarchisation plus marquée des connexions.

C. Implémentation et Visualisation

Le script Python présenté ci-dessus simule la dynamique de mise à jour du SCN sur un nombre donné d'itérations et pour plusieurs valeurs de ω_{\max} . La fonction *simulate_scn* réalise la mise à jour des poids selon la règle additive, applique le clipping, et stocke l'historique de la matrice de pondérations. Les graphiques générés montrent l'évolution moyenne des pondérations ainsi qu'une heatmap de la matrice finale, permettant d'observer l'impact du réglage de ω_{\max} sur la structure émergente.

Conclusion

L'ajustement de ω_{\max} joue un rôle clé dans la régulation des connexions au sein d'un SCN. Par l'intermédiaire du **clipping**, on impose une borne sur chaque pondération, empêchant ainsi un emballage individuel et favorisant une répartition plus sélective des ressources de connexion. L'implémentation en Python ci-dessus, accompagnée de visualisations graphiques, permet d'illustrer concrètement comment la dynamique évolue pour différentes valeurs de ω_{\max} et comment ce paramètre influence l'émergence des clusters dans le réseau. Cette approche offre ainsi un outil précieux pour expérimenter et affiner la structure auto-organisée d'un SCN dans des contextes variés d'apprentissage profond et de systèmes multi-agents.

5.5.4.1. Ajout d'un Terme Stochastique dans la Mise à Jour

A. Principes Généraux et Analogie avec le Recuit Simulé

Dans un SCN, la mise à jour classique des pondérations se réalise selon une règle additive déterministe de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le taux d'apprentissage et $\tau > 0$ représente le coefficient de décroissance. Cette mise à jour tend à conduire chaque $\omega_{i,j}$ vers un point fixe théorique $\omega_{i,j}^* = \frac{S(i,j)}{\tau}$. Toutefois, dans un environnement purement déterministe, le réseau peut se figer dans des configurations localement stables qui ne sont pas optimales, car certains liens potentiellement pertinents pourraient être négligés en raison du verrouillage dans un minimum local.

Pour pallier ce problème, un **terme stochastique** est ajouté à la règle de mise à jour. Ce terme, généralement de la forme $\alpha \xi_{i,j}(t)$, apporte une perturbation aléatoire contrôlée :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha \xi_{i,j}(t).$$

Ici, $\xi_{i,j}(t)$ est une variable aléatoire tirée d'une distribution centrée, par exemple $\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma^2)$ pour une distribution normale ou $\mathcal{U}(-\delta, \delta)$ pour une distribution uniforme, et $\alpha > 0$ ajuste l'amplitude relative de cette perturbation par rapport au terme déterministe.

Ce procédé s'inspire du **recuit simulé**, une méthode d'optimisation stochastique dans laquelle l'injection de bruit permet d'éviter un blocage prématûr dans un minimum local, tout en favorisant l'exploration de l'espace de recherche. Le terme stochastique permet ainsi à la dynamique du SCN de « rebondir » hors d'un point fixe potentiellement sous-optimal et d'explorer d'autres configurations qui pourraient conduire à une meilleure organisation globale du réseau.

B. Modélisation Mathématique et Impact sur la Dynamique

La dynamique complète, avec l'ajout du bruit, s'exprime par :

$$\Delta \omega_{i,j}(t) = \omega_{i,j}(t+1) - \omega_{i,j}(t) = \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha \xi_{i,j}(t).$$

En l'absence de bruit ($\alpha = 0$), la suite $\{\omega_{i,j}(t)\}$ converge vers $\omega_{i,j}^* \approx \frac{S(i,j)}{\tau}$ (ou s'éteint si $S(i,j)$ est trop faible). L'ajout du terme $\alpha \xi_{i,j}(t)$ modifie cette convergence : plutôt que de converger exactement vers un point fixe, la dynamique converge vers une **distribution** autour du point fixe déterministe. La variance de cette distribution est fonction de l'amplitude α et de la variance σ^2 de la variable aléatoire.

En outre, la possibilité de faire varier α au fil du temps (par exemple en appliquant une décroissance exponentielle du type

$$\alpha(t+1) = \delta \alpha(t) \quad \text{avec} \quad 0 < \delta < 1,$$

) permet d'initier la dynamique dans un régime d'**exploration** (fort bruit initial) et de la laisser se stabiliser en réduisant progressivement le niveau de perturbation, à l'instar d'un processus de recuit.

C. Potentiel d'Exploration et Risques d'Instabilité

L'introduction d'un terme stochastique offre plusieurs avantages en termes d'exploration de l'espace de configuration. Parfois, même si la dynamique déterministe se rapproche d'un attracteur, de légères fluctuations permettent au système de s'extraire de minima locaux peu profonds pour atteindre des configurations plus globalement optimales. Cependant, un niveau de bruit excessif (un α trop grand ou une variance trop élevée pour $\xi_{i,j}(t)$) peut empêcher la convergence, générant des oscillations persistantes ou même un comportement chaotique.

Le défi consiste donc à **équilibrer** le niveau d'exploration induit par le terme stochastique avec la nécessité de convergence pour que le SCN puisse former des **clusters** stables et significatifs.

D. Implémentation Python Complète avec Graphiques

Nous présentons ci-dessous une implémentation en Python qui simule la mise à jour des pondérations dans un SCN avec ajout d'un terme stochastique. Le code inclut la visualisation graphique de la dynamique de quelques liens pour observer l'effet du bruit et de la décroissance de α .

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Fixation d'une graine pour la reproductibilité
np.random.seed(42)

# -----
# Paramètres de la simulation
# -----
n = 50 # Nombre d'entités (pour la matrice, on considère une simulation dense)
eta = 0.05 # Taux d'apprentissage
tau = 1.0 # Coefficient de décroissance
initial_alpha = 0.1 # Amplitude initiale du terme stochastique
delta = 0.99 # Facteur de décroissance de alpha (baisse de température)
sigma = 0.1 # Écart-type pour la distribution normale du bruit
num_iterations = 300 # Nombre total d'itérations

# -----
# Création d'une matrice de synergie S
# -----
# Pour simplifier, nous supposons que S(i,j) est constant et uniforme
```

```

# sur le réseau, par exemple S(i,j)=0.8 pour i != j.
S = np.full((n, n), 0.8)
np.fill_diagonal(S, 0) # Pas de self-synergie

# -----
# Initialisation des pondérations ω
# -----
omega = np.random.uniform(0, 0.05, (n, n))
np.fill_diagonal(omega, 0)

# Stockage de l'historique pour la visualisation
omega_history = np.zeros((num_iterations + 1, n, n))
omega_history[0] = omega.copy()

# Initialisation de alpha
alpha = initial_alpha
alpha_history = [alpha]

# -----
# Simulation de la mise à jour avec terme stochastique
# -----
for t in range(num_iterations):
    # Mise à jour alpha (recuit simulé)
    alpha *= delta
    alpha_history.append(alpha)

    # Mise à jour de ω pour chaque paire (i,j)
    omega_next = omega.copy()
    for i in range(n):
        for j in range(n):
            if i != j:
                # Calcul déterministe de la mise à jour additive
                deterministic_update = eta * (S[i, j] - tau * omega[i, j])
                # Terme stochastique: bruit tiré de N(0, sigma^2)
                stochastic_term = alpha * np.random.normal(0, sigma)
                # Mise à jour totale
                omega_next[i, j] = omega[i, j] + deterministic_update + stochastic_term
                # Option de clipping pour éviter des valeurs négatives
                if omega_next[i, j] < 0:
                    omega_next[i, j] = 0
    # Mise à jour de la matrice pour la prochaine itération
    omega = omega_next.copy()
    omega_history[t+1] = omega.copy()

# -----
# Visualisation des résultats
# -----
# Exemple de visualisation : moyenne des pondérations au fil du temps
mean_omega = [np.mean(omega_history[t][np.triu_indices(n, k=1)]) for t in range(num_iterations+1)]
plt.figure(figsize=(12, 6))
plt.plot(mean_omega, label="Pondération moyenne")
plt.xlabel("Itérations")
plt.ylabel("Valeur moyenne de ω (hors diagonale)")
plt.title("Évolution de la pondération moyenne avec terme stochastique")
plt.legend()
plt.grid(True)

```

```

plt.show()

# Visualisation d'une heatmap de la matrice finale ω
plt.figure(figsize=(8, 6))
sns.heatmap(omega_history[-1], cmap="viridis", annot=False)
plt.title("Heatmap de la matrice ω finale")
plt.xlabel("Index j")
plt.ylabel("Index i")
plt.show()

# Visualisation de l'évolution de alpha
plt.figure(figsize=(12, 4))
plt.plot(alpha_history, label="α (Amplitude du bruit)")
plt.xlabel("Itérations")
plt.ylabel("α")
plt.title("Diminution du paramètre α au fil des itérations")
plt.legend()
plt.grid(True)
plt.show()

```

Explications Complètes

Principe et Formulation

La mise à jour déterministe initiale est donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Pour introduire un **terme stochastique**, nous ajoutons une perturbation $\alpha \xi_{i,j}(t)$ où $\xi_{i,j}(t)$ est une variable aléatoire suivant une distribution normale centrée $\mathcal{N}(0, \sigma^2)$ (vous pouvez également utiliser une distribution uniforme). Le paramètre α contrôle l'amplitude du bruit. Ainsi, la mise à jour devient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha \xi_{i,j}(t).$$

En outre, pour favoriser une phase d'exploration initiale, on peut permettre à α de décroître au fil des itérations selon :

$$\alpha(t+1) = \delta \alpha(t) \quad \text{avec} \quad 0 < \delta < 1.$$

Impact sur la Dynamique

L'ajout du terme stochastique permet au réseau de s'extraire des minima locaux en introduisant une variabilité qui, dans certains cas, peut mener à un réaménagement des clusters. La dynamique ainsi obtenue est alors celle d'un **processus stochastique** qui converge non pas vers un unique point fixe, mais vers une distribution autour de ce point, dont la variance dépend de α et σ .

Lorsque le bruit est élevé, le système explore un plus grand nombre de configurations avant de se stabiliser. À mesure que α décroît (baisse de « température »), le réseau se stabilise progressivement, permettant la formation de clusters plus robustes et adaptés à la variabilité des données.

Implémentation et Visualisations

Le code Python ci-dessus simule la mise à jour des pondérations $\omega_{i,j}$ dans un SCN avec injection de bruit. Nous utilisons des bibliothèques standards telles que **NumPy** pour le calcul matriciel et **Matplotlib** ainsi que **Seaborn** pour la visualisation.

- La **fonction de simulation** initialise une matrice ω avec de faibles valeurs aléatoires, puis met à jour cette matrice selon la règle stochastique.
- À chaque itération, le paramètre α est multiplié par δ pour simuler le recuit simulé.
- La mise à jour est effectuée pour chaque paire (i, j) (hors diagonale) en ajoutant le terme déterministe et le terme stochastique.
- Finalement, plusieurs visualisations sont proposées : l'évolution de la pondération moyenne, la heatmap de la matrice finale, et la décroissance de α .

Ces visualisations permettent d'observer comment le bruit influence la convergence des liens et la formation de clusters, offrant ainsi une compréhension concrète de l'impact de l'**ajout d'un terme stochastique** dans la dynamique d'un SCN.

Conclusion

L'ajout d'un **terme stochastique** dans la mise à jour des pondérations, via la formule

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha \xi_{i,j}(t),$$

permet de surmonter le risque de figement dans des minima locaux et favorise l'exploration de nouvelles configurations. Le paramètre α (avec une décroissance programmée) joue un rôle crucial en équilibrant l'exploration initiale et la stabilisation ultérieure. L'implémentation en Python ci-dessus, accompagnée de graphiques explicatifs, illustre concrètement comment cette dynamique peut être réalisée et visualisée dans un SCN. Ce procédé constitue une stratégie efficace pour améliorer la **flexibilité** et la **robustesse** d'un réseau auto-organisé dans des applications variées telles que la robotique multi-agent ou l'apprentissage non supervisé.

5.5.4.2. Diminution Progressive de la “Température” pour Éviter de Rester Piégé dans un Minimum Local

Dans le cadre d'un **Synergistic Connection Network** (SCN), la mise à jour des pondérations $\omega_{i,j}$ repose sur une dynamique qui, dans sa version déterministe, tend à figer les valeurs autour d'un point fixe. Or, cette stabilité peut être trompeuse puisqu'elle risque de conduire le système à se retrouver bloqué dans un **minimum local** qui ne reflète pas l'optimum global de la synergie. Pour pallier ce problème, il est pertinent d'introduire un **terme stochastique** dans la mise à jour, de manière à favoriser l'**exploration** des configurations alternatives. Afin de contrôler cette exploration, on rend l'amplitude du bruit variable dans le temps en introduisant le concept de "**température**" qui décroît progressivement.

A. Principes du Recuit Simulé et Rôle de la Température

Le principe du **recuit simulé** est inspiré du processus de refroidissement en métallurgie, où un matériau est chauffé à haute température pour permettre aux atomes de se réarranger librement, puis refroidi lentement pour stabiliser une structure ordonnée. Dans le contexte d'un SCN, la mise à jour stochastique des pondérations s'exprime par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha(t) \xi_{i,j}(t),$$

où η est le taux d'apprentissage, τ le coefficient de décroissance, et $\alpha(t)$ représente la **température** à l'itération t . La variable aléatoire $\xi_{i,j}(t)$ est tirée d'une distribution centrée (par exemple, $\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma^2)$). La fonction de température $\alpha(t)$ est conçue pour décroître au fil du temps afin de favoriser une phase initiale d'exploration, suivie d'une phase de stabilisation. Une loi exponentielle de décroissance peut être utilisée, par exemple :

$$\alpha(t+1) = \delta \alpha(t) \quad \text{avec} \quad 0 < \delta < 1.$$

Ainsi, au début, lorsque $\alpha(t)$ est élevé, le bruit introduit des fluctuations importantes qui permettent au SCN d'explorer un vaste espace de solutions et d'échapper à des minima locaux peu profonds. Au fur et à mesure que $\alpha(t)$ décroît, les fluctuations diminuent, ce qui permet au réseau de se stabiliser autour d'une configuration optimale.

B. Impact sur la Dynamique du Système

La dynamique globale devient alors celle d'un **processus stochastique non autonome** où la mise à jour des pondérations est influencée par un bruit dont l'amplitude évolue avec t . En l'absence de bruit ($\alpha = 0$), le système converge généralement vers le point fixe déterministe :

$$\omega_{i,j}^* \approx \frac{S(i,j)}{\tau}.$$

Avec le terme stochastique, la suite $\{\omega_{i,j}(t)\}$ ne converge pas vers une valeur unique, mais plutôt vers une **distribution** centrée autour de ce point fixe. La variance de cette distribution dépend de la magnitude de $\alpha(t)$ et de σ . Le processus de décroissance de $\alpha(t)$ assure qu'après une phase d'exploration, le système « refroidit » et se stabilise progressivement, permettant la formation de **clusters** robustes et la convergence vers une configuration qui maximise la synergie globale.

C. Implémentation Python Complète avec Visualisation

Le code Python suivant simule cette dynamique dans un SCN en utilisant la mise à jour stochastique avec recuit simulé. Nous générerons également plusieurs graphiques pour illustrer l'évolution des pondérations, la décroissance de la température, et une heatmap finale de la matrice ω .

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Fixer une graine pour la reproductibilité
np.random.seed(42)

# Paramètres de la simulation
n = 30 # nombre d'entités (pour la simulation dense)
eta = 0.05 # taux d'apprentissage
tau = 1.0 # coefficient de décroissance
alpha0 = 0.2 # température initiale (amplitude du bruit)
delta = 0.98 # facteur de décroissance de la température (0 < delta < 1)
sigma = 0.1 # écart-type du bruit (pour N(0, sigma^2))
iterations = 300 # nombre d'itérations

# Création d'une matrice de synergie S (par exemple, S(i,j)=0.8 pour i != j)
S = np.full((n, n), 0.8)
np.fill_diagonal(S, 0) # pas de synergie sur la diagonale

# Initialisation de la matrice des pondérations ω
omega = np.random.uniform(0, 0.05, (n, n))
np.fill_diagonal(omega, 0)

# Stocker l'évolution de ω et de la température
omega_history = np.zeros((iterations + 1, n, n))
omega_history[0] = omega.copy()
alpha_history = [alpha0]

# Température initiale
alpha = alpha0
```

```

# Simulation de la dynamique avec recuit simulé
for t in range(iterations):
    # Mise à jour de la température (recuit simulé)
    alpha *= delta
    alpha_history.append(alpha)

    # Initialiser une nouvelle matrice pour ω(t+1) (double-buffer)
    omega_next = omega.copy()
    for i in range(n):
        for j in range(n):
            if i != j:
                # Calcul déterministe de la mise à jour additive
                delta_det = eta * (S[i, j] - tau * omega[i, j])
                # Génération du bruit aléatoire (N(0, sigma^2))
                noise = alpha * np.random.normal(0, sigma)
                # Mise à jour complète
                omega_next[i, j] = omega[i, j] + delta_det + noise
                # Option de clipping pour éviter des valeurs négatives
                if omega_next[i, j] < 0:
                    omega_next[i, j] = 0
    # Passage à l'itération suivante
    omega = omega_next.copy()
    omega_history[t+1] = omega.copy()

# -----
# Visualisations
# -----

# 1. Évolution de la température alpha au fil des itérations
plt.figure(figsize=(10, 4))
plt.plot(alpha_history, color='blue', lw=2)
plt.xlabel("Itérations")
plt.ylabel("Température ( $\alpha$ )")
plt.title("Diminution Progressive de la Température")
plt.grid(True)
plt.show()

# 2. Évolution de la pondération moyenne (hors diagonale) au fil des itérations
mean_omega = [np.mean(omega_history[t][np.triu_indices(n, k=1)]) for t in range(iterations+1)]
plt.figure(figsize=(10, 4))
plt.plot(mean_omega, color='green', lw=2)
plt.xlabel("Itérations")
plt.ylabel("Pondération moyenne ( $\omega$ )")
plt.title("Évolution de la Pondération Moyenne dans le SCN")
plt.grid(True)
plt.show()

# 3. Heatmap de la matrice  $\omega$  à la fin de la simulation
plt.figure(figsize=(8, 6))
sns.heatmap(omega_history[-1], cmap="viridis", annot=False)
plt.title("Heatmap Finale de la Matrice  $\omega$ ")
plt.xlabel("Index j")
plt.ylabel("Index i")
plt.show()

# 4. Visualisation de l'évolution de quelques liens spécifiques

```

```

# Choisissons 5 liens aléatoires pour suivre leur évolution
num_links_to_plot = 5
links = []
while len(links) < num_links_to_plot:
    i, j = np.random.randint(0, n, 2)
    if i != j and (i, j) not in links:
        links.append((i, j))

plt.figure(figsize=(12, 6))
for (i, j) in links:
    link_values = [omega_history[t][i, j] for t in range(iterations+1)]
    plt.plot(link_values, label=f"\omega({i},{j})")
plt.xlabel("Itérations")
plt.ylabel("Valeur de \omega")
plt.title("Évolution des Pondérations de Quelques Liens Sélectionnés")
plt.legend()
plt.grid(True)
plt.show()

```

Explications et Analyse

Principe et Formulation

La mise à jour stochastique des pondérations dans un SCN s'exprime par la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha(t) \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ est une variable aléatoire centrée, typiquement distribuée selon $\mathcal{N}(0, \sigma^2)$. La **température** $\alpha(t)$ décroît selon une loi exponentielle définie par

$$\alpha(t+1) = \delta \alpha(t),$$

ce qui signifie que le bruit est fort en début de simulation (favorisant l'exploration) et diminue progressivement pour stabiliser la configuration finale.

Impact sur la Dynamique

Grâce à ce mécanisme de recuit simulé, le système peut sortir des minima locaux en autorisant temporairement des fluctuations importantes dans les pondérations. La dynamique converge ensuite vers une distribution autour de la valeur théorique $\frac{S(i,j)}{\tau}$, avec une variance qui diminue au fil du temps lorsque $\alpha(t)$ tend vers zéro. Ce procédé permet d'obtenir des **clusters** plus robustes et mieux adaptés aux données.

Implémentation et Visualisations

Le code Python présenté simule la mise à jour itérative de la matrice ω dans un SCN, en intégrant un terme stochastique dont l'amplitude diminue progressivement. Plusieurs graphiques sont générés pour visualiser :

- La décroissance de la température α ,
- L'évolution de la pondération moyenne (hors diagonale),
- Une heatmap de la matrice ω à la fin de la simulation,
- L'évolution temporelle de quelques liens spécifiques.

Ces visualisations permettent d'observer comment le **recuit simulé** aide le système à explorer l'espace des configurations dans un premier temps, puis à se stabiliser pour former des clusters cohérents.

Conclusion

L'introduction d'un **terme stochastique** avec une **température décroissante** (recuit simulé) dans la mise à jour des pondérations permet de surmonter le risque de piégeage dans des minima locaux, en favorisant l'exploration initiale avant de stabiliser la dynamique du SCN. La formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha(t) \xi_{i,j}(t)$$

associe ainsi la correction déterministe à une perturbation contrôlée, dont l'amplitude $\alpha(t)$ décroît progressivement. L'implémentation Python fournie illustre de manière complète ce mécanisme, et les visualisations permettent de suivre la diminution progressive de la température, l'évolution de la moyenne des pondérations, et la formation d'une structure finale dans le SCN. Ce procédé offre une stratégie efficace pour améliorer la **flexibilité** et la **robustesse** des réseaux auto-organisés dans divers domaines d'application.

5.6. Interfaces Entrée-Sortie et Gestion en Temps Réel

5.6.1. Ajouter / Retirer une Entité

- 5.6.1.1. $\text{addEntity}(\mathcal{E}_i)$: initialisation des liens $\omega_{i,\dots}$.
- 5.6.1.2. $\text{removeEntity}(\mathcal{E}_i)$: suppression de ses liens, mise à jour de la structure.

5.6.2. Mise à Jour Périodique vs. Événementielle

- 5.6.2.1. Stratégie batch : attend un certain nombre d'événements avant un recalcul massif.
- 5.6.2.2. Stratégie streaming : mise à jour locale itérative.

5.6.3. Extraction de Clusters / Sous-Réseaux

- 5.6.3.1. $\text{getTopLinks}(i, k)$, $\text{getAllClusters}(\theta)$.
- 5.6.3.2. Usage externe (visualisation, classification), design des formats de sortie (JSON, CSV, etc.).

5.6. Interfaces Entrée-Sortie et Gestion en Temps Réel

Dans l'environnement d'un **SCN** (Synergistic Connection Network), la **gestion** des entités en temps réel revêt une importance cruciale : il n'est pas rare qu'au cours de la simulation ou d'une application concrète, on doive **ajouter** de nouvelles entités (apparition de nouveaux objets, agents, données) ou **supprimer** des entités devenues obsolètes ou inutiles. Dans la même veine, il peut s'avérer nécessaire d'opérer des mises à jour non pas selon un calendrier fixe (batch), mais en mode "flux" (streaming), de sorte à s'adapter rapidement à des changements fréquents. La section (5.6) traite ainsi des **interfaces** d'entrée-sortie qui permettent de gérer le SCN "en direct", et explique comment ajouter ou retirer des entités (5.6.1), quelles stratégies de mise à jour choisir (périodique ou événementielle, 5.6.2), et enfin comment extraire ou visualiser des **clusters** à la volée (5.6.3).

5.6.1. Ajouter / Retirer une Entité

L'une des fonctionnalités clés d'une interface temps réel est la possibilité de **modifier** dynamiquement l'ensemble $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ que le SCN prend en charge. Nous décrivons ici les opérations fondamentales `addEntity` et `removeEntity`, indispensables pour maintenir la cohérence de la structure ω .

5.6.1.1. `addEntity(\mathcal{E}_i)` : Initialisation des Liens $\omega_{i,\dots}$

L'opération `addEntity(\mathcal{E}_i)` permet d'incorporer une *nouvelle entité* \mathcal{E}_i dans un SCN (Synergistic Connection Network) en cours de fonctionnement. Son importance se justifie par la nécessité, dans de nombreux contextes pratiques, d'intégrer des données apparaissant dynamiquement : un flux multimédia de grande ampleur, l'arrivée d'utilisateurs dans un réseau social ou la mise en service d'agents supplémentaires dans un système robotique. Sur le plan **mathématique**, l'ajout d'une entité modifie la dimension de la matrice ω qui passe de $(n \times n)$ à $((n + 1) \times (n + 1))$, et requiert une procédure d'initialisation des **liaisons** $\omega_{i,j}$ associées au nouvel index i .

Dans la plupart des applications, on choisit de **réservier** un *identifiant* supplémentaire — souvent $i = n + 1$ — et de créer les lignes et colonnes nécessaires. Les connexions $\omega_{i,j}$ et $\omega_{j,i}$ sont alors insérées avec des valeurs nulles ou très faibles, comme

$$\omega_{i,j}(0) = 0, \quad \omega_{j,i}(0) = 0, \quad \forall j \in \{1, \dots, n\}.$$

Cette option garantit que le nouvel élément \mathcal{E}_i ne perturbe pas immédiatement la structure du réseau et autorise la **dynamique** du DSL à faire croître ou décroître ses liaisons de manière autonome selon la règle

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

ou selon la version intégrant des mécanismes d'**inhibition** ou de **bruit**. Dans certains schémas plus rapides, il est concevable de fixer

$$\omega_{i,j}(0) \propto S(i, j) \quad \text{et} \quad \omega_{j,i}(0) \propto S(j, i),$$

afin d'accorder une forme de "démarrage accéléré" à \mathcal{E}_i en exploitant l'information de synergie dès son arrivée.

Du point de vue **ingénierie**, cette démarche s'accompagne souvent de l'allocation nécessaire en mémoire pour stocker les poids $\omega_{i,\dots}$. Dans le cas d'un **SCN** dense, cela implique de redimensionner la matrice, tandis que dans une implémentation **sparse**, il suffit d'ajouter un *champ* ou une *structure* associée au nouvel identifiant i . Il est aussi crucial de gérer les index pour que chaque module du DSL (mise à jour, synergie, parsimonie) reconnaissse la nouvelle entité. L'insertion d'une *API* claire pour la fonction `addEntity(e)` assure la cohérence entre la phase d'ajout et la poursuite de l'apprentissage local.

Sur le plan **mathématique**, cette insertion d'entités à la volée modifie le caractère du système, passant d'une évolution dans un espace de dimension n^2 à un espace de dimension $(n + 1)^2$. La topologie du **SCN** s'en trouve enrichie : dans les itérations ultérieures, le nouveau nœud \mathcal{E}_i noue des liens selon

$$\Delta \omega_{i,j}(t) = \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

ou dans les variantes multiplicatives ou inhibées, sans avoir besoin de réinitialiser l'ensemble du réseau. Cette propriété est essentielle lorsque l'on traite un **flux continu** : chaque arrivée d'information préserve la structure accumulée jusque-là, tout en offrant à la nouvelle entité la possibilité de s'insérer de façon fluide dans les *clusters* existants ou d'en faire émerger de nouveaux.

Conclusion

La fonction `addEntity(\mathcal{E}_i)` illustre la capacité d'un SCN à *croître* de manière dynamique, facteur déterminant pour les applications où l'on ne peut se contenter d'une architecture fixe. En initialisant les liens à des valeurs nulles ou faibles, on mise sur la **règle de mise à jour** locale pour ajuster progressivement $\omega_{i,j}$ en accord avec la fonction de **synergie**. L'implémentation doit nécessairement gérer l'allocation et la synchronisation des données, laissant à la **dynamique** du DSL le soin de décider des connexions à consolider ou à supprimer. Une telle approche rend le système "vivant" et adaptable, au sens où l'insertion d'une nouvelle entité ne remet pas en question les clusters déjà formés, mais offre plutôt au réseau la possibilité d'acquérir de nouveaux liens ou de se réorganiser pour incorporer la *vitalité* apportée par \mathcal{E}_i .

5.6.1.2. `removeEntity(\mathcal{E}_i)` : Suppression de ses Liens et Mise à Jour de la Structure

Dans un **SCN** (Synergistic Connection Network) dynamique, la possibilité de retirer une entité déjà présente est aussi cruciale que l'ajout abordé en **Section 5.6.1.1**. Les raisons de cette suppression sont multiples : données devenues caduques, agent quittant un système multi-agent, ou simple volonté d'alléger la structure pour des motifs de performance. Le point essentiel est de gérer la disparition de \mathcal{E}_i de façon cohérente, afin d'éviter toute corruption ou incohérence dans les indices et dans la mise à jour des liaisons ω .

A. Motivations et Cadre Mathématique

La suppression d'une entité \mathcal{E}_i dans un SCN implique de **réduire** la dimension de l'espace des connexions. Lorsque la matrice ω était de taille $(n \times n)$, le retrait d'une entité transforme cet espace en $(n - 1) \times (n - 1)$, à moins de conserver les emplacements vides pour ne pas réindexer. Du point de vue des règles d'**auto-organisation** (voir **Section 5.5** sur la mise à jour), toute entité disparue ne doit plus influer sur la synergie ou la dynamique de **clusters**. Concrètement, les liaisons $\omega_{i,\cdot}$ et $\omega_{\cdot,i}$ doivent cesser d'exister, et la matrice ω n'a plus à entretenir la structure associée à l'index i .

Les circonstances menant à une telle suppression sont variées. Dans un réseau temps réel, une **donnée** peut n'être plus valide après un délai, ou bien se révéler obsolète au sens où sa synergie $\{S(i,j)\}$ n'est plus calculée. Dans un réseau social ou un système multi-agent, un **utilisateur** ou un **robot** peut quitter la scène et libérer les ressources y afférentes. Sans suppression, des lignes et colonnes fantômes peuvent se maintenir, produisant un bruit inutile et encombrant la topologie du SCN.

B. Opération de Suppression et Conséquences sur la Structure

La commande `removeEntity(\mathcal{E}_i)` doit éliminer toutes les références à l'entité \mathcal{E}_i dans l'ensemble des liaisons. Cela équivaut à annuler ou à effacer les *lignes* $\omega_{i,j}$ pour tout j et les *colonnes* $\omega_{j,i}$ pour tout j . L'implémentation diffère selon que la matrice ω est en format **dense** ou **sparse**. Dans un format dense, on peut choisir de mettre à zéro tous les $\omega_{i,j}$ et $\omega_{j,i}$, quitte à laisser une ligne morte, ou alors reconstruire une matrice $(n - 1) \times (n - 1)$ en réindexant toutes les entités. Dans un format sparse (hashmaps ou listes d'adjacence), il suffit de balayer les entrées associées à l'index i et de les supprimer.

Cette manipulation a un impact direct sur la **dynamique** du SCN : toute entité \mathcal{E}_k qui présentait des liens $\omega_{k,i}$ non négligeables perd désormais ce canal de synergie et peut se réorganiser dans les itérations ultérieures. Du point de vue mathématique, on retire un ensemble de degrés de liberté, et la dynamique locale

$$\omega_{k,j}(t+1) = \omega_{k,j}(t) + \eta[S(k,j) - \tau \omega_{k,j}(t)]$$

se poursuit en l'absence de l'index i . Dans le cas d'une **inhibition compétitive**, la somme $\sum_m \omega_{k,m}$ peut également s'en trouver modifiée, libérant ainsi de la "place" pour d'autres connexions.

C. Cohérence Logique et Implantation dans l'API

Une **API** clairement définie, par exemple `removeEntity(e)`, assure la cohérence de l'opération. Cette fonction :

- 270. Identifie l'entité \mathcal{E}_i (via son ID).
- 271. Supprime ou met à zéro l'ensemble $\omega_{i,j}$ et $\omega_{j,i}$.
- 272. Informe éventuellement les autres modules (mise à jour, synergie, monitoring) que \mathcal{E}_i n'est plus active.

Si le SCN fonctionne en mode **temps réel**, il peut être préférable de gérer cette suppression en fin d'itération, afin de ne pas altérer les boucles de mise à jour en cours. Dans un **cadre distribué** (voir **Section 5.2.3.3**), il est également important d'annoncer la disparition aux autres nœuds afin qu'ils ne continuent pas de calculer $\omega_{i,j}$ pour des valeurs qui ne sont plus censées exister.

Les considérations de **performance** et de **stabilité** justifient l'implémentation d'une telle opération. Sur le plan de la **complexité**, maintenir des entités obsolètes génère un surcoût systématique, tant en calcul de synergie $\{S(i,j)\}$ qu'en stockage de ω . Sur le plan de la **stabilité**, la persistance d'entités inactives peut orienter les mises à jour vers de fausses configurations ou influencer négativement les clusters.

D. Conséquences Dynamiques et Réorganisation de Clusters

La disparition d'une entité \mathcal{E}_i peut entraîner une **réorganisation** notable si \mathcal{E}_i occupait un rôle clé (par exemple, un "hub" au sein d'un cluster). Les entités \mathcal{E}_k qui appuyaient leur synergie sur la présence de \mathcal{E}_i vont connaître un nouvel équilibre. Sur le plan purement mathématique, les règles d'**update** (cf. **Section 5.5**) s'appliquent désormais sur l'espace $(n-1)$. Les pondérations associées à d'autres liens peuvent se renforcer ou diminuer afin de compenser la perte de contribution de $\omega_{k,i}$.

Si l'on considère un SCN avec règles d'**inhibition** (voir **Section 5.5.2**) qui contraint $\sum_j \omega_{k,j} \leq \Omega_{\max}$, la suppression de \mathcal{E}_i libère un potentiel de croissance pour d'autres liaisons de \mathcal{E}_k , modifiant parfois la morphologie des clusters existants.

Conclusion

La fonction `removeEntity(\mathcal{E}_i)`, symétrique de `addEntity(\mathcal{E}_i)` présentée en **Section 5.6.1.1**, assure la **souplesse** d'un SCN évolutif. En retirant proprement toutes les liaisons associées à \mathcal{E}_i , on évite toute confusion dans la dynamique locale, on prévient le gaspillage de ressources et on préserve la cohérence mathématique de la règle de mise à jour. La dimension du système se trouve ainsi ajustée de manière flexible, permettant un fonctionnement *continu* même dans un flux de données ou d'entités changeant. L'opération de suppression, si elle est correctement orchestrée, aboutit à un réseau plus **robuste** et mieux adapté à la réalité des applications, qu'il s'agisse de flux de données éphémères ou d'agents intermittents dans un système distribué.

5.6.2. Mise à Jour Périodique vs. Événementielle

Au sein d'un SCN (Synergistic Connection Network) gérant des données ou des agents en **temps réel**, on peut se demander **comment** orchestrer la dynamique de mise à jour des pondérations ω lorsque l'environnement évolue (ajout/suppression d'entités, arrivée de nouvelles informations, etc.). Deux stratégies s'opposent en général : la **mise à**

jour par batch (périodique) et la **mise à jour événementielle** (streaming). Tandis que la première attend qu'un certain nombre d'événements se soit accumulé avant de relancer un "recalcul" global, la seconde ajuste les pondérations localement à chaque événement ou de manière quasi continue.

5.6.2.1. Stratégie Batch : Attendre un Certain Nombre d'Événements avant un Recalcul Massif

La **stratégie batch**, également qualifiée de stratégie **périodique**, consiste à ne pas actualiser en continu le *Synergistic Connection Network* (SCN) au gré de chaque arrivée ou départ d'entité, ni dès la moindre fluctuation de flux. Il s'agit plutôt de définir des intervalles, exprimés en nombre d'événements ou en durée temporelle, au terme desquels on procède à une mise à jour globale et plus approfondie de la structure des liens ω . Cette approche présente l'intérêt de limiter les recomputations constantes, ce qui s'avère essentiel lorsque le nombre d'entités n peut devenir très grand et que la fonction de **synergie** $\{S(i,j)\}$ nécessite un calcul coûteux.

Dans le principe, la méthode batch repose sur l'idée qu'une suite de petits changements, intervenant de manière rapprochée, ne justifie pas forcément un recalcul complet des liaisons $\omega_{i,j}(t)$. On se contente alors de maintenir les règles d'**update** (cf. [Section 5.5](#)) sans intégrer immédiatement les nouvelles entités ou les entités supprimées. Les modifications sont simplement enregistrées dans une liste d'attente ou dans un buffer d'événements. Dès lors qu'un certain nombre de changements a été atteint, ou que l'on arrive à un instant prédefini Δt dans l'horloge système, on exécute le "batch" : on ajoute ou on retire toutes les entités différées, on met à jour la matrice ω ou la structure sparse correspondante, et l'on recalcule si nécessaire le *score* de synergie pour ces nouveaux nœuds.

Sur le plan **mathématique**, on peut formaliser la stratégie batch par un compteur d'événements κ . À chaque ajout ou suppression d'entité, on incrémente κ . Tant que $\kappa < \kappa_{\max}$, on n'altère pas la topologie du réseau ; on laisse la dynamique $\{\omega_{i,j}(t)\}$ se dérouler comme si rien n'avait changé, selon la règle habituelle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

ou selon l'une de ses variantes (inhibition compétitive, mécanisme multiplicatif, présence de bruit). Lorsque $\kappa \geq \kappa_{\max}$, on applique en bloc les opérations différées et on remet κ à zéro. De façon similaire, dans un cadre temporel, on peut définir un pas de temps Δt et n'autoriser le recalcul qu'aux instants multiples de Δt .

Ce procédé présente un avantage d'**économie de calcul** : pour un système volumineux ou hautement dynamique, il est souvent plus efficient d'effectuer une *mise à jour massive* de temps à autre plutôt qu'un ajustement permanent à chaque échantillon. Il offre également une forme de **stabilité** temporelle, dans la mesure où la structure du SCN ne se transforme pas incessamment et reste intelligible entre deux temps de recalcul. En revanche, cette approche implique une **perte de réactivité** : si un événement critique se produit, il ne sera intégré qu'au prochain lot, laissant un décalage entre la réalité et l'état du réseau.

Un exemple concret se rencontre dans un système de **recommandation** en ligne, où des utilisateurs et des produits apparaissent constamment. Une intégration *événement par événement* exigerait de réapprendre la topologie $\{\omega_{i,j}\}$ à chaque nouvel utilisateur, ce qui se révélerait onéreux. En adoptant une stratégie batch, on peut décider de réactualiser le SCN toutes les 12 heures ou lorsque 100 nouveaux comptes sont créés. Entre ces deux recalculs, l'infrastructure de recommandation applique la dynamique existante pour affiner ω , sans devoir gérer en permanence l'arrivée de nouvelles entités.

Sur le plan de l'**implémentation** logicielle, on maintient généralement un **registre** des modifications. Chaque fois qu'une entité est ajoutée ou supprimée (cf. [Sections 5.6.1.1](#) et [5.6.1.2](#)), on ne touche pas immédiatement à la matrice ω . On incrémente un compteur ou on note la date de la dernière modification. Puis, lorsque le quota est atteint ou que l'horloge a franchi un seuil, on applique de façon *synchronisée* la mise à jour : on relance éventuellement un calcul de synergie $\{S(i,j)\}$ pour les nouveaux nœuds, on retire ceux qui sont devenus obsolètes, et on solde toutes les modifications avant de reprendre le cours normal de la dynamique locale.

En matière de **comparaison** avec la stratégie *événemtuelle* (voir [Section 5.6.2.2](#)), le choix dépend largement de la nature du flux de données et du coût de recalcul. La stratégie batch favorise la **robustesse** et la réduction du **bruit** dans le SCN, au détriment de la *fraîcheur* de l'information. Elle prévient aussi de possibles oscillations liées à des micro-événements successifs. La stratégie événementielle, en revanche, intègre chaque changement quasiment en temps réel,

ce qui convient mieux à des applications de haute réactivité, mais peut alourdir la charge de calcul si les insertions et suppressions sont très fréquentes.

En conclusion, la **stratégie batch** offre une solution pragmatique pour les grandes architectures : plutôt que de mobiliser des ressources de calcul constamment, on regroupe les évolutions sur une fenêtre donnée, on les applique simultanément, puis on laisse la *dynamique* du SCN s'ajuster paisiblement jusqu'à la prochaine vague de modifications. Cette gestion par lots est par conséquent courante dans les systèmes qui tolèrent un léger retard entre la détection d'un événement et sa répercussion effective sur la structure du SCN, mais qui souhaitent maintenir un fonctionnement maîtrisé, à faible bruit, et bien échelonné dans le temps.

5.6.2.2. Stratégie Streaming : Mise à Jour Locale Itérative

La stratégie dite “streaming” se propose de mettre à jour le *Synergistic Connection Network* (SCN) de manière continue et événementielle, plutôt que de recourir à des “lots” ou *batches* espacés dans le temps. Cette approche se caractérise par une réactivité immédiate face à chaque événement (arrivée ou départ d’entité, flux de données, etc.), et privilégie un ajustement *incrémental* plutôt qu’un recalcul massif et périodique. Elle se distingue ainsi de la logique présentée en **Section 5.6.2.1**, où les modifications sont accumulées puis appliquées en bloc.

Dans la stratégie streaming, on considère que toute perturbation du système doit être répercutée aussitôt sur la structure de poids $\{\omega_{i,j}\}$. Cela signifie qu’à chaque fois qu’une entité \mathcal{E}_{n+1} est ajoutée, le SCN se dote immédiatement de la ligne et de la colonne correspondantes. Dans une implémentation dense, la matrice ω est agrandie de $(n+1) \times (n+1)$. Dans un format plus économique, on crée la structure sparse nécessaire pour référencer $\omega_{n+1,j}$ et $\omega_{j,n+1}$. À l’inverse, si une entité \mathcal{E}_i disparaît, alors les lignes et colonnes associées sont supprimées (voir **Sections 5.6.1.1** et **5.6.1.2**).

L’un des atouts majeurs de ce mode “événemementiel” tient à sa **réactivité**. Dans un contexte où les données évoluent rapidement, comme dans un flux sensoriel temps réel ou un réseau d’agents qui se connectent et se déconnectent sans cesse, une latence trop élevée peut nuire aux performances globales. La mise à jour locale itérative assure que le SCN reflète toujours l’état courant du système, ce qui peut être déterminant pour les applications critiques (surveillance, détection de fraude, coordination robotique). Mathématiquement, au moment où se produit l’ajout de \mathcal{E}_{n+1} , on initialise $\omega_{n+1,j} \approx 0$ (ou à une valeur proportionnelle à $S(n+1,j)$) puis on laisse la règle DSL classique

$$\omega_{n+1,j}(t+1) = \omega_{n+1,j}(t) + \eta [S(n+1,j) - \tau \omega_{n+1,j}(t)]$$

prendre le relai lors des itérations suivantes, sans attendre le déclenchement d’un gros *batch*.

Cependant, cette extrême proximité avec l’évolution du système n’est pas dénuée de difficultés. Dès lors que la fréquence des événements devient très élevée (centaines ou milliers de changements par minute), chaque mise à jour locale représente un travail algorithmique pouvant se cumuler de façon importante. Il faut donc concevoir une mécanique de mise à jour suffisamment efficace pour qu’à chaque arrivée (ou suppression) d’entité, on ne se retrouve pas à parcourir l’entièreté du réseau $\{\omega_{i,j}\}$. L’idée est généralement de s’en tenir à un *voisinage local*, par exemple les entités présentant une synergie initiale importante, afin d’éviter de manipuler systématiquement les $O(n^2)$ liaisons.

La stratégie streaming exige également un certain soin pour préserver la **stabilité** du SCN dans la durée. Si le flux d’événements est permanent, le système ne bénéficie plus vraiment de phases de repos où les liens $\omega_{i,j}(t)$ peuvent converger. Les règles présentées en **Sections 5.5** (mécanismes additifs, multiplicatifs, inhibition compétitive, etc.) doivent donc être calibrées pour limiter les oscillations. Sur le plan mathématique, on se situe dans un cadre dynamique *non autonome* où la taille et le contenu du réseau changent en continu, ce qui peut rendre la démonstration de convergence plus délicate.

Enfin, la distribution du SCN sur un ensemble de nœuds physiques (voir **Section 5.2.3.3**) accroît la complexité de cette approche, dans la mesure où chaque arrivée d’entité nécessite une propagation asynchrone de l’événement à tous les sous-ensembles où la synergie ou la mise à jour locale doit être prise en compte. Dans un environnement hautement distribué, la charge de synchronisation peut s’avérer très lourde si les événements affluent. Certaines implémentations

optent pour des techniques asynchrones plus avancées, éventuellement conjuguées à des mécanismes de réPLICATION ou de cohérence partielle, de façon à gérer cet afflux sans provoquer un embouteillage sur le réseau de communication.

La stratégie streaming peut être modérée par des approches hybrides, parfois qualifiées de “micro-batch” ou “fenêtre glissante”, qui visent à limiter l’application de la mise à jour à quelques entités à la fois. Par exemple, l’intégration d’une nouvelle entité \mathcal{E}_{n+1} peut être prise en compte immédiatement, mais la réallocation de synergies pour la globalité du SCN peut être différée si l’élément ajouté n’a qu’un impact minime. On se situe alors dans un compromis entre la mise à jour événementielle pure et la mise à jour par lots (batch), laissant la possibilité d’adapter le niveau de réactivité en fonction du *taux de modification*.

En conclusion, la stratégie streaming correspond à un paradigme d'**adaptation continue**, idéal pour les cas où la fraîcheur des informations est prioritaire et où la dynamique doit coller au plus près à un flux évoluant sans relâche. Elle suppose toutefois une implantation soignée, apte à traiter les événements en temps réel sans excéder les capacités de calcul, et un paramétrage approprié (coefficients η , τ , éventuel bruit ou inhibition) pour éviter de perpétuelles oscillations. Elle se révèle ainsi complémentaire à la stratégie batch, présentée en **Section 5.6.2.1**, dans le vaste domaine des SCN dynamiques, où chaque modèle d’actualisation répond à des besoins et des contraintes différents.

5.6.3. Extraction de Clusters / Sous-Réseaux

Même lorsqu’un **SCN** (Synergistic Connection Network) opère en temps réel (sections 5.6.1 et 5.6.2), il demeure essentiel de pouvoir **extraire** ou **observer** en continu la structure émergente : quels liens sont réellement “forts” ? Quels **clusters** ou sous-réseaux se constituent ? Cette section (5.6.3) aborde donc les opérations par lesquelles on récupère des informations sur le SCN pour les exploiter ou les visualiser. Nous nous concentrerons d’abord sur deux fonctions couramment rencontrées : $\text{getTopLinks}(i, k)$ et $\text{getAllClusters}(\theta)$ (5.6.3.1), puis sur la manière de mettre ces informations à disposition d’autres systèmes (5.6.3.2).

5.6.3.1. $\text{getTopLinks}(i, k)$, $\text{getAllClusters}(\theta)$

Le **Deep Synergy Learning** amène souvent à gérer un *Synergistic Connection Network* (SCN) où les entités $\{\mathcal{E}_i\}$ sont reliées par une matrice de liens ω . L’**objectif** de ce réseau est la mise en évidence de *clusters* ou sous-réseaux fortement cohérents, toutefois la matrice ω , qu’elle soit dense ou présentée en format sparse, ne fournit pas automatiquement l’information structurée sous forme de communautés. Dans cette optique, les opérations internes $\text{getTopLinks}(i, k)$ et $\text{getAllClusters}(\theta)$ permettent une extraction commode des connexions les plus significatives, qu’il s’agisse d’une analyse locale (pour un nœud donné) ou d’une identification globale de clusters par seuil.

A. Motivation Générale pour l’Extraction de Clusters

L’évolution d’un SCN sous les règles d’auto-organisation (décrites en **Section 5.5**) vise à établir, renforcer ou inhiber des liaisons $\{\omega_{i,j}\}$ en fonction de la **synergie** $S(i, j)$. À l’issue de cette dynamique, on espère discerner des *communautés* ou *composantes* où les liens internes conservent une grande valeur, traduisant un haut degré de similarité ou d’interaction entre les entités. Néanmoins, la matrice ω peut atteindre une taille considérable, et son contenu n’est pas directement lisible si l’on se contente de parcourir l’ensemble $\{\omega_{i,j}\}$. Les fonctions getTopLinks et getAllClusters s’inscrivent donc dans un registre **opérationnel**, conçu pour extraire les liens les plus pertinents et révéler la structure de *clusters* à un instant donné de l’itération.

B. $\text{getTopLinks}(i, k)$: Extraction Locale des Liens Principaux

L’opération $\text{getTopLinks}(i, k)$ fournit, pour une entité \mathcal{E}_i , les k liens $\omega_{i,j}$ de plus forte valeur. L’idée est de rapidement identifier le **voisinage principal** de \mathcal{E}_i , c’est-à-dire les entités \mathcal{E}_j vers lesquelles \mathcal{E}_i entretient la relation la plus intense. Cette notion rappelle la pratique du “ k plus proches voisins” (k-NN), à la différence que la mesure de proximité ou d’intérêt est ici fixée par la valeur courante de $\omega_{i,j}$.

Sur le plan **mathématique**, la requête $\text{getTopLinks}(i, k)$ se fonde sur un tri (ou un ordre partiel) des valeurs $\{\omega_{i,1}, \omega_{i,2}, \dots, \omega_{i,n}\}$. Pour $\omega_{i,j}$ susceptible de prendre des valeurs faibles ou nulles, une structure de données adaptée

(section 5.3.2.2 sur les indexations et accès rapides) permet de retrouver les k premières liaisons en un temps réduit, souvent $O(k\log n)$ ou $O(n\log k)$. La liste rentrée reflète l'état du SCN à l'itération en cours, et peut aisément changer dans le temps au fur et à mesure que la synergie entre \mathcal{E}_i et les autres entités \mathcal{E}_j évolue.

En **ingénierie logicielle**, `getTopLinks(i, k)` est très utile pour la **visualisation locale** : on peut afficher l'entité \mathcal{E}_i et ses k connexions dominantes, et ainsi naviguer dans le SCN de proche en proche. Ce mécanisme convient aussi à des tâches de **recommandation**, où l'on cherche les “voisins” d'un utilisateur \mathcal{E}_i dans un espace de préférences, ou à des algorithmes incrémentaux qui ne souhaitent pas analyser l'intégralité de la matrice ω .

C. `getAllClusters(θ)` : Vue Globale par Seuil

L'opération `getAllClusters(θ)` vise à révéler la structure des *clusters* en appliquant un **seuil** θ sur les liaisons $\{\omega_{i,j}\}$. Plus précisément, on considère le sous-graphe dont les arêtes satisfont

$$\omega_{i,j} \geq \theta,$$

tandis que celles situées en dessous de θ sont ignorées. Il en résulte un graphe binaire, où les liens jugés “forts” sont seuls préservés. Sur ce graphe, il devient aisément d'identifier des **composantes connexes** ou des **communautés**, par exemple au moyen d'un parcours DFS/BFS ou d'algorithmes de détection de clusters (type Louvain ou analyse de modularité). Ainsi, l'utilisateur choisit θ pour ajuster la granularité : un seuil proche de 0 rend le graphe très dense, tandis qu'un seuil élevé ne conserve que quelques connexions majeures, formant des *clusters* réduits mais très cohésifs.

Le choix de θ se rattache à la fois à la **distribution** des valeurs $\omega_{i,j}$ et aux intentions d'exploration. On peut également faire varier θ par paliers pour observer les transitions dans la structure de clusterisation (cf. “coupes” successives dans la matrice ω). Sur le plan **mathématique**, `getAllClusters(θ)` décrit un instantané : il se peut que deux appels consécutifs à des itérations différentes t et $t + 10$ n'aboutissent pas au même regroupement, si les liaisons ont significativement changé entre-temps.

D. Intégration dans l'Interface SCN et Avantages

Le fait de **fournir** `getTopLinks(i, k)` et `getAllClusters(θ)` de manière native dans un *SCN* présente plusieurs avantages. Tout d'abord, l'utilisateur ou les modules applicatifs n'ont pas besoin de manipuler la matrice ω dans sa totalité, laquelle peut être massive pour un grand n . Ensuite, si l'implémentation de ω emploie des structures de données **sparse** ou exploite des mécanismes de **parsimonie** (voir Section 5.5.3), la fonction “top- k ” et le filtrage par θ peuvent être réalisés efficacement, sans avoir à traiter toutes les paires (i, j) . Par ailleurs, cette capacité d'extraction **internationale** (en lien direct avec l'état du réseau) permet des mises à jour rapides et cohérentes en fin d'itération, ou même pendant un flux streaming (voir Sections 5.6.2.1 et 5.6.2.2).

Les usages pratiques sont nombreux. Dans des systèmes de **recommandation**, `getTopLinks(i, k)` sert à déterminer quelles entités, produits ou contenus sont les plus proches d'un utilisateur \mathcal{E}_i . Dans un **dashboard** de supervision, on peut régulièrement appeler `getAllClusters(θ)` pour dévoiler l'évolution des communautés en temps réel. Dans un cadre purement analytique, ces fonctions constituent la base de l'examen de la structure *multi-échelle* du SCN : si θ décroît, on obtient des regroupements plus vastes ; s'il croît, on isole les noyaux les plus cohésifs.

E. Exemple d'Utilisation : Visualisation Locale et Regroupement Global

Dans la pratique, on peut imaginer un scénario où la commande `getTopLinks($i, 5$)` est utilisée pour naviguer dans le SCN, à la manière d'un **explorateur** : partant d'une entité \mathcal{E}_i , on obtient ses 5 liens les plus forts, puis on se déplace vers l'un de ces voisins et réitère, formant ainsi un chemin au sein du réseau. Cette visualisation locale évite de charger toutes les relations ω .

Pour la **vue d'ensemble**, on appelle `getAllClusters($\theta = 0.2$)` : on filtre le réseau sur les liens supérieurs ou égaux à 0.2, et l'on détecte ensuite les sous-ensembles fortement connectés (composantes connexes si on considère le graphe non orienté, communautés plus élaborées si on applique un algorithme idoine). On obtient ainsi un partitionnement en *clusters*, qui reflète la structure issue de l'auto-organisation : à bas seuil, les clusters sont gros et moins nets ; à haut seuil, on ne garde que des groupements extrêmement soudés.

Conclusion

Les fonctions `getTopLinks(i, k)` et `getAllClusters(θ)` constituent des pièces maîtresses de l'**interface SCN** pour l'extraction de structure. Elles offrent une **passerelle** entre la représentation interne (pondérations $\{\omega_{i,j}\}$) et les besoins concrets de **visualisation**, de **clustering** ou d'**analyse**. Le **k plus forts liens** d'une entité \mathcal{E}_i permet un accès local et ciblé, tandis que la construction d'un graphe seuil θ et l'identification de composantes ou communautés procure une vue globale et hiérarchique des clusters. Par leur intégration directe dans l'API SCN, ces fonctions s'inscrivent dans la logique d'un système de *Deep Synergy Learning* modulable, au service d'usages aussi divers que la recommandation, la classification ou la supervision temps réel.

5.6.3.2. Usage Externe (Visualisation, Classification), Design des Formats de Sortie (JSON, CSV, etc.)

Une fois la structure d'un Synergistic Connection Network (SCN) extraite — par exemple via des outils internes tels que `getTopLinks` ou `getAllClusters` présentés en **Section 5.6.3.1** — se pose la question de la mise à disposition de ces informations pour d'autres modules ou d'autres analyses. Dans la plupart des applications, le **réseau** n'existe pas pour lui-même : on souhaite en effet **exploiter** les pondérations ω , les clusters ou la topologie qui en résulte pour des finalités concrètes, comme la **visualisation** en temps réel, la **classification** dans un autre système d'apprentissage machine, ou encore la **détection** de motifs ou d'anomalies. Cette section détaille donc les aspects relatifs au **design** d'interfaces et de formats de sortie, afin de diffuser la structure de clusterisation et les liens du SCN vers l'extérieur.

A. Visualisation et Classification : Motivations et Approches

Dans un grand nombre de cas d'usage, les données structurées par le SCN — qu'il s'agisse de regroupements (clusters) ou de liaisons pondérées en cours d'évolution — doivent être **communiquées** à des composants externes. Il peut s'agir d'une bibliothèque de visualisation, typiquement pour représenter un graphe dynamique, ou d'un pipeline d'apprentissage souhaitant intégrer les *communautés* ou la *mesure de synergie* comme caractéristiques supplémentaires.

273. Visualisation interactive

En ingénierie de **dashboard** ou en contextes de **monitoring**, il est fréquent qu'on veuille afficher la disposition des entités, leurs liens forts et les clusters qu'ils forment. Des bibliothèques comme D3.js, Sigma.js ou Cytoscape nécessitent des **formats** standards (JSON, GraphML, etc.) pour représenter la liste des nœuds et des arêtes. Un composant "SCN" doit donc être en mesure de fournir ces informations, idéalement à la demande (API REST) ou via une mise à jour continue (WebSocket). Cela permet de voir "en direct" quels liens $\omega_{i,j}$ sont hauts et comment les regroupements évoluent à mesure que la dynamique s'applique.

274. Classification et analyse en aval

De plus, le SCN peut servir de **préalable** à la classification d'entités, par exemple en exportant la **structure de cluster** comme une étiquette de "classe" ou de "communauté". Un autre module de type "machine learning supervisé" récupère alors ces *labels* et les exploite comme feature ou comme partition de référence. De même, des algorithmes de détection d'anomalies peuvent comparer les comportements d'entités supposées être dans le même cluster et repérer d'éventuelles divergences. Les services externes ont donc besoin d'accéder :

- Soit aux **clusters** (chaque entité \mathcal{E}_i étant associée à un cluster \mathcal{C}_i),
- Soit aux **scores** $\omega_{i,j}$ directement, afin de calculer leur propre fonction.

Dans un système évolutif (voir **Section 5.6.2**), il est d'autant plus important d'actualiser ces informations dès que le réseau subit une transformation (arrivées, départs, renforcement de liens). Les modules externes peuvent alors "pull" (requête périodique) ou se faire "push" (notification en temps réel) d'un snapshot mis à jour du SCN.

B. Formats de Sortie : JSON, CSV, GraphML, etc.

L'**export** des données du SCN peut prendre plusieurs formes, en fonction de la taille, de l'usage et des standards adoptés dans l'environnement applicatif. Les formats textuels comme JSON et CSV dominent largement, mais des formats plus spécifiques à la représentation de graphes (GraphML, GEXF) ou des formats binaires peuvent également être requis.

275.JSON

JSON (JavaScript Object Notation) est le plus fréquemment employé, aussi bien pour une exposition **web** (via REST) que pour un échange **machine-to-machine**. On peut par exemple renvoyer un objet JSON structuré comme suit :

```
{  
  "nodes": [  
    { "id": i, "label": ..., "attributes": {...} },  
    ...  
  ],  
  "links": [  
    { "source": i, "target": j, "weight": \omega_{i,j} },  
    ...  
  ],  
  "clusters": [  
    { "clusterId": c_1, "nodes": [i_1, i_2, ...] },  
    ...  
  ]  
}
```

Les bibliothèques de visualisation comme D3.js, Cytoscape.js, etc. comprennent aisément ce type de structure, et il est simple d'y associer des informations graphiques (couleurs, positions, etc.). JSON est aussi facile à générer ou à parser dans la plupart des langages.

276.CSV (Comma-Separated Values)

CSV demeure très populaire pour un usage **off-line** ou pour un import/export rapide vers Excel, pandas ou R. On peut créer :

- Un **fichier d'arêtes** (*edges.csv*) :

```
source,target,weight  
i,j,\omega  
i',j',\omega  
...  
...
```

- Un **fichier de clusters** (*clusters.csv*) :

```
node,clusterId  
i,c  
i',c  
...  
...
```

Bien que cette forme tabulaire soit succincte, elle est facilement analysable par une large gamme d'outils. En revanche, elle n'est pas idéale pour coder des structures plus complexes (attributs multiples par lien ou par noeud).

277.GraphML, GEXF

Dans le champ de la **visualisation de graphes** et de la recherche en réseau, des formats comme GraphML ou GEXF (Graph Exchange XML Format) sont couramment utilisés, notamment pour la compatibilité avec des outils comme Gephi, Tulip ou NetworkX. Le SCN peut être converti dans l'un de ces formats afin de bénéficier des algorithmes et interfaces de ces logiciels (layout, calcul de modularité, etc.).

278.Autres possibilités

Pour des **SCN** de très grande envergure (millions de noeuds, millions d'arêtes), le volume de données devient prohibitif dans un format JSON ou CSV. On recourt alors à des exports **binaires** ou à des protocoles de sérialisation plus

compacts (Protobuf, Cap'n Proto). L'essentiel est de réduire la taille des liens enregistrés, éventuellement en exploitant la **parcimonie** et en ne stockant que les arêtes supérieures à un certain seuil.

C. Mécanismes de Production et d'Accès

Pour offrir de tels formats, un composant interne peut réaliser :

279.Une **API** ou un **service** spécialisé dans l'export. Celui-ci reçoit une requête, par exemple `GET /scn/clusters?threshold=0.3`, exécute `getAllClusters(0.3)`, puis sérialise le résultat en JSON ou CSV.

280.Une **fonction** locale, telle que `exportLinks(format="CSV")`, qui génère un fichier ou un flux textuel contenant la liste des liaisons.

281.Des **instantanés** programmés : à chaque itération ou toutes les N itérations, on produit un “snapshot” de l'état actuel du SCN (avec labels de cluster), que l'on stocke dans un répertoire ou que l'on envoie à un pipeline de data-mining.

Le **temps réel** (visualisation en continu) peut être assuré par des technologies comme WebSocket ou SSE (Server-Sent Events), où le SCN pousse automatiquement des mises à jour sous forme de JSON chaque fois qu'un changement notable (évolution de la matrice ω , arrivée/suppression d'entité) se produit.

D. Exemples d'Usage Externe

282.Visualisation d'un Graphe Dynamique

Un tableau de bord (exemple avec D3.js) récupère l'état courant via `getAllClusters(θ)`, puis construit un JSON “nodes/links”. L'interface affiche un graphe évolutif où les nœuds sont colorés par *cluster*. Si un nœud E_i change de cluster suite à un réajustement de ω , le prochain appel reflète cette mutation, donnant un effet d'animation ou de transition dans la vue utilisateur.

283.Réintégration dans une Chaîne de Classification

Un module aval, chargé de classifier les entités (ex. détection de fraude dans des transactions), demande régulièrement la partition $\{\mathcal{C}_i\}$ ou la liste des top- k liens de chaque entité. Il intègre ces données comme *features*, afin de compléter l'information sur chaque transaction ou utilisateur. Par exemple, un utilisateur dont le cluster diffère subitement de son historique peut être marqué comme suspect.

284.Diagnostic et Archivage

Dans des systèmes complexes, on peut consigner — à intervalles réguliers — un export CSV ou JSON de la matrice ω seillée. Ces archives constituent une base d'analyse a posteriori pour valider la stabilité de l'auto-organisation, expliquer des événements remarquables (consolidation ou éclatement de clusters), ou confronter le réseau à un “oracle” de vérité terrain.

Conclusion

Les informations sur la structure d'un SCN (top-links, clusters, communautés) ne demeurent pas confinées au module d'auto-organisation. Elles doivent être **exposées** sous des formats standards (JSON, CSV, GraphML...) à d'autres composantes, qu'il s'agisse de bibliothèques de visualisation, d'outils de classification ou de frameworks d'exploration de graphes. Sur le plan **ingénierie**, on mettra ainsi en place :

- Des **endpoints** ou **fonctions** d'export (ex. `exportClusters(theta, format="json")`),
- Un **protocole** de diffusion (API REST, WebSocket, fichiers),
- Une **intégration** dans des bibliothèques ou plateformes de data science.

Ce design veille à équilibrer la **lisibilité** (formats standard), la **performance** (possibilité d'extraire seulement un sous-ensemble ou de filtrer par seuil) et la **dimension** du SCN (certaines structures massives imposant des stratégies d'export parcimonieux). L'essentiel demeure de faire en sorte que le SCN, en tant que *cœur adaptatif*, puisse rayonner ses

informations de clusterisation au **reste** de l'architecture logicielle, facilitant à la fois la **compréhension** humaine et le **traitement** automatique par les algorithmes externes.

5.7. Distribution, Scalabilité et Architecture Modulaire

5.7.1. SCN Distribué sur Plusieurs Nœuds

- 5.7.1.1. Motifs : si n est grand, on peut diviser en sous-SCN (chacun gère un sous-ensemble d'entités).
- 5.7.1.2. Communication et protocole d'échange ω inter-sous-SCN.
- 5.7.1.3. Risque d'incohérence et solutions (synchronisation épisodique, etc.).

5.7.2. Mise en Place d'un “meta-SCN”

- 5.7.2.1. Idée : un SCN global reliant plusieurs sous-SCN, usage d'un “graphon” ou d'un “meta-nœud”.
- 5.7.2.2. Périodicité de la mise à jour, agrégation de clusters émergents.
- 5.7.2.3. Cas d'une architecture microservices : chaque service = sous-SCN.

5.7.3. Ingénierie Logicielle

- 5.7.3.1. Patterns de conception (Observer, Strategy, etc.) pour modular l'implémentation.
- 5.7.3.2. Approches NoSQL, BDD orientée graphe (Neo4j, Titan) pour stocker ω .
- 5.7.3.3. Équilibrage de charge (sharding de la matrice ω).

5.7. Distribution, Scalabilité et Architecture Modulaire

Dans l'optique d'un **SCN** (Synergistic Connection Network) de très grande taille (nombre d'entités n élevé) ou réparti entre différents sites, la question de la **distribution** et de la **scalabilité** se pose inévitablement. Le chapitre 5.7 se concentre sur cette problématique : comment structurer un SCN en plusieurs sous-parties (sous-SCN), chacune gérant une portion du réseau ou un sous-ensemble d'entités, et comment organiser la communication et la synchronisation inter-blocs. Même si cette approche s'éloigne d'une vision purement locale ou centralisée (chapitres précédents), elle s'avère indispensable pour une **architecture** modulaire et apte à traiter des volumes importants de données ou des entités éparpillées sur plusieurs nœuds.

5.7.1. SCN Distribué sur Plusieurs Nœuds

Lorsque la dimension n d'un SCN devient considérable ou que les entités se trouvent déjà distribuées (ex. multiples robots, serveurs distants, etc.), on procède à un **partitionnement** du réseau en "sous-SCN". Cette section (5.7.1) en présente les justifications (5.7.1.1), puis explore les questions de communication (5.7.1.2) et de synchronisation (5.7.1.3).

5.7.1.1. Motifs : si n est grand, on peut diviser en sous-SCN (chacun gère un sous-ensemble d'entités)

Lorsqu'un **Synergistic Connection Network (SCN)** doit gérer un nombre très élevé d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$, les défis se multiplient tant du point de vue **mathématique** qu'**algorithmique** et **matériel**. En effet, si n atteint des valeurs de l'ordre de 10^5 à plusieurs millions, la gestion centralisée d'une matrice de pondérations Ω de dimensions $n \times n$ devient rapidement problématique. D'un point de vue **mémoire**, le coût de stockage de Ω croît en $\mathcal{O}(n^2)$, ce qui, pour de grandes valeurs de n , peut représenter plusieurs dizaines, voire centaines de gigaoctets, même en utilisant des structures de données optimisées telles que des matrices **sparse**. Du point de vue **algorithmique**, la mise à jour de toutes les pondérations nécessite la réalisation d'un nombre quadratique d'opérations par itération, ce qui rend la boucle de mise à jour extrêmement coûteuse en temps de calcul sur une seule machine. Enfin, sur le plan **matériel**, l'accès à la bande passante mémoire et le parallélisme deviennent des goulots d'étranglement majeurs lorsqu'on traite simultanément des millions de liaisons.

Afin de surmonter ces limitations, il est judicieux de segmenter le SCN en plusieurs **sous-SCN**. Chaque sous-SCN est responsable de la gestion d'un sous-ensemble d'entités, ce qui permet de partitionner la matrice globale Ω en blocs plus petits, notés par $\Omega^{(p)}$ pour $p = 1, \dots, m$. Mathématiquement, si l'ensemble des entités est divisé en m groupes $\mathcal{V}_1, \dots, \mathcal{V}_m$ avec $|\mathcal{V}_p| \approx \frac{n}{m}$, chaque sous-SCN gère une matrice de taille approximative $\left(\frac{n}{m}\right)^2$. Ainsi, le coût de stockage et de mise à jour de chaque bloc est réduit en $\mathcal{O}\left(\frac{n^2}{m^2}\right)$ par rapport au système centralisé, et les interactions inter-blocs peuvent être traitées séparément, voire de manière moins fréquente.

Du point de vue **théorique**, le partitionnement en sous-SCN permet également de mieux capturer des **structures naturelles** ou des **séparations géographiques** dans le réseau. Par exemple, dans un réseau de robots répartis sur plusieurs sites physiques ou dans un système multi-agent où les entités présentent des caractéristiques distinctes (capteurs, actuateurs, utilisateurs, etc.), il est cohérent de regrouper les entités par affinité ou proximité. Ce découpage favorise non seulement une **réduction des coûts** (tant en mémoire qu'en temps de calcul) mais aussi une **meilleure résilience**. En cas de défaillance d'un sous-SCN, les autres continuent de fonctionner de manière autonome, assurant une tolérance aux pannes au niveau du système global.

Du point de vue **ingénierie**, l'implémentation d'un SCN partitionné est facilitée par une architecture logicielle modulaire. Chaque sous-SCN peut être déployé sur un serveur ou un nœud de calcul différent, avec des interfaces standardisées pour la communication inter-blocs. Cette approche modulaire simplifie la gestion de la parallélisation et permet d'exploiter des stratégies de calcul distribué, où la mise à jour locale se fait indépendamment dans chaque sous-SCN, et où les interactions inter-blocs sont traitées par des protocoles d'échange de messages ou des agrégations périodiques.

Implémentation Python avec Graphiques

Le code Python suivant simule un SCN avec un grand nombre d'entités que nous partitionnons en plusieurs sous-SCN. Chaque sous-SCN gère la mise à jour de ses propres pondérations $\omega_{i,j}$ en utilisant la règle additive classique, et nous visualisons ensuite la structure finale des matrices pour observer l'effet de la segmentation.

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Paramètres de base
n = 200      # Nombre total d'entités
m = 4        # Nombre de sous-SCN (groupes)
eta = 0.05    # Taux d'apprentissage
tau = 1.0     # Coefficient de décroissance
iterations = 100  # Nombre d'itérations pour chaque sous-SCN

# Création d'une matrice de synergie S globale pour toutes les entités
# Pour simplifier, on fixe S(i,j) = 0.8 pour i != j et 0 sur la diagonale
S_global = np.full((n, n), 0.8)
np.fill_diagonal(S_global, 0)

# Partitionner les entités en m groupes égaux
group_size = n // m
subSCN_indices = [range(i * group_size, (i + 1) * group_size) for i in range(m)]

# Initialisation des matrices de pondérations pour chaque sous-SCN
omega_sub = {}
for idx, indices in enumerate(subSCN_indices):
    size = len(indices)
    # Initialisation aléatoire faible pour éviter l'absence de signal
    omega_sub[idx] = np.random.uniform(0, 0.05, (size, size))
    np.fill_diagonal(omega_sub[idx], 0)

# Fonction de mise à jour déterministe additive pour un sous-SCN
def update_omega(omega, S, eta, tau):
    new_omega = omega.copy()
    size = omega.shape[0]
    for i in range(size):
        for j in range(size):
            if i != j:
                delta = eta * (S[i, j] - tau * omega[i, j])
                new_omega[i, j] = omega[i, j] + delta
                # Éviter les valeurs négatives par clipping
                new_omega[i, j] = max(new_omega[i, j], 0)
    return new_omega

# Stockage de l'évolution pour visualisation
omega_history = {idx: [] for idx in range(m)}

# Simulation de mise à jour pour chaque sous-SCN
for idx, indices in enumerate(subSCN_indices):
    omega = omega_sub[idx]
    # On suppose que la matrice de synergie S pour chaque sous-SCN est extraite de S_global
    S_local = S_global[np.ix_(list(indices), list(indices))]
    # Stockage initial
    omega_history[idx].append(omega)

```

```

omega_history[idx].append(omega.copy())
for t in range(iterations):
    omega = update_omega(omega, S_local, eta, tau)
    omega_history[idx].append(omega.copy())
omega_sub[idx] = omega # Sauvegarde finale dans le dictionnaire

# Visualisation : Heatmap finale de chaque sous-SCN
plt.figure(figsize=(14, 10))
for idx in range(m):
    plt.subplot(2, m // 2, idx + 1)
    sns.heatmap(omega_sub[idx], cmap="viridis", cbar=True)
    plt.title(f"Sous-SCN {idx+1} (taille = {group_size}x{group_size})")
plt.tight_layout()
plt.show()

# Visualisation : Évolution de la moyenne des pondérations dans chaque sous-SCN
plt.figure(figsize=(10, 6))
for idx in range(m):
    mean_values = [np.mean(omega_history[idx][t]) for t in range(iterations + 1)]
    plt.plot(mean_values, label=f"Sous-SCN {idx+1}")
plt.xlabel("Itérations")
plt.ylabel("Pondération moyenne")
plt.title("Évolution de la Pondération Moyenne dans Chaque Sous-SCN")
plt.legend()
plt.grid(True)
plt.show()

```

Analyse et Discussion

Principes et Formulation Mathématique

La mise à jour de chaque pondération dans un SCN est déterminée par la formule additive

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Dans le contexte d'un grand nombre d'entités, la division en **sous-SCN** permet de limiter les coûts de stockage et de calcul en traitant séparément des blocs de la matrice Ω . Chaque sous-SCN est associé à une sous-matrice $\omega^{(p)}$ de taille réduite, ce qui rend l'algorithme de mise à jour plus gérable, tant du point de vue algorithmique que matériel.

Implémentation et Visualisation

L'implémentation Python proposée simule le processus de mise à jour pour chaque sous-SCN. Le code partitionne les entités en m groupes égaux, extrait la matrice de synergie locale correspondante, et applique itérativement la règle de mise à jour additive. Les visualisations générées – des heatmaps finales pour chaque sous-SCN et des courbes montrant l'évolution de la pondération moyenne – illustrent clairement l'auto-organisation locale dans chaque partition. Ces graphiques permettent d'observer la convergence progressive des pondérations, tout en montrant que la division en sous-SCN réduit la complexité du problème global.

Avantages du Partitionnement

En divisant le réseau en sous-SCN, on réduit le coût en mémoire de la matrice Ω et on limite le nombre d'opérations par itération. Ce découpage favorise également la parallélisation, car chaque sous-SCN peut être traité indépendamment, voire sur des machines différentes. De plus, dans des scénarios où les entités présentent des caractéristiques géographiques ou typologiques distinctes, le partitionnement permet d'adapter la dynamique locale aux spécificités de chaque sous-ensemble.

Conclusion

La stratégie de partitionnement d'un grand SCN en sous-SCN permet de surmonter les limitations inhérentes à un réseau centralisé de grande dimension. D'un point de vue mathématique, la segmentation réduit la complexité en divisant une matrice Ω de taille $\mathcal{O}(n^2)$ en plusieurs sous-matrices de taille réduite, chacune traitée de manière autonome. Du point de vue algorithmique et matériel, cette approche favorise la scalabilité et la résilience du système, tout en facilitant l'implémentation de mises à jour parallèles et de stratégies de contrôle local. L'implémentation Python présentée, assortie de graphiques, démontre concrètement comment la dynamique d'auto-organisation se déploie dans chaque sous-SCN, offrant ainsi un outil pédagogique et pratique pour l'étude des SCN à grande échelle.

5.7.1.2. Communication et Protocole d'Échange ω Inter-sous-SCN

Dans un **Synergistic Connection Network** (SCN) segmenté en plusieurs sous-réseaux distincts, la gestion des interactions entre entités réparties dans différents sous-SCN devient une question cruciale. Lorsqu'un ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ est divisé en plusieurs blocs $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$, chaque sous-SCN gère localement sa propre matrice de pondérations, c'est-à-dire les valeurs $\omega_{i,j}$ pour $i, j \in \mathcal{V}_p$. Cependant, les liaisons inter-blocs, c'est-à-dire celles pour lesquelles $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$ avec $p \neq q$, nécessitent une coordination particulière. La présente section expose en détail les principes et enjeux de la **communication** et des **protocoles d'échange** permettant de synchroniser les pondérations inter-sous-SCN.

A. Notion de Liens Inter-Blocs et Rôle du Protocole

Lorsque le SCN est segmenté, il ne s'agit pas de traiter chaque sous-SCN comme un système isolé ; en effet, des **synergies** $S(i,j)$ importantes peuvent exister entre des entités appartenant à des blocs différents. Mathématiquement, pour des entités $\mathcal{E}_i \in \mathcal{V}_p$ et $\mathcal{E}_j \in \mathcal{V}_q$ avec $p \neq q$, la pondération $\omega_{i,j}$ est une variable d'importance qui doit être intégrée dans la dynamique globale du SCN. Dans un tel cadre, le protocole de communication a pour objectif de garantir que ces valeurs inter-blocs soient correctement mises à jour et que les informations échangées entre les sous-SCN reflètent la synergie réelle entre les entités, même si elles résident sur des nœuds ou des serveurs différents.

D'un point de vue mathématique, la dynamique globale peut être vue comme un système couplé, dans lequel chaque sous-SCN contribue à l'évolution de l'ensemble des pondérations. Ainsi, les mises à jour locales dans un sous-SCN, notées par exemple $\omega^{(p)}(t)$ pour le bloc \mathcal{V}_p , doivent être complétées par des échanges d'informations pour les pondérations $\omega_{i,j}$ lorsque $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$. Le protocole doit donc assurer la **transmission** et la **synchronisation** de ces données inter-blocs afin de préserver la cohérence globale.

B. Protocoles de Communication et de Synchronisation

Pour assurer un échange efficace et cohérent des pondérations inter-sous-SCN, plusieurs stratégies peuvent être envisagées. Parmi elles, deux approches principales se dégagent :

- **Responsabilité Unique** : Dans ce mode, une sous-SCN est désignée comme responsable de la gestion d'un lien inter-blocs particulier. Par exemple, si $\omega_{i,j}$ relie une entité de \mathcal{V}_p à une entité de \mathcal{V}_q , alors le sous-SCN de \mathcal{V}_p (ou un module central dédié) stocke et met à jour cette pondération. Les autres blocs qui doivent accéder à cette information effectuent des requêtes périodiques pour obtenir la valeur actualisée. Cette méthode centralise la mise à jour d'un lien spécifique, ce qui facilite la garantie d'une cohérence temporelle, mais nécessite une définition claire de la responsabilité de chaque lien.
- **Copie Locale et Synchronisation** : Alternativement, chaque sous-SCN peut maintenir sa propre copie locale des pondérations inter-blocs. Dans ce cas, lorsqu'un sous-SCN modifie une pondération $\omega_{i,j}$ pour $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$, il doit diffuser cette modification aux autres sous-SCN concernés. Pour gérer ce processus, un mécanisme de synchronisation est indispensable, souvent via l'attribution de **timestamps** ou de **numéros de version** pour chaque mise à jour. La synchronisation peut se faire de manière asynchrone, avec une stratégie de réconciliation en cas de divergence entre copies, ou par des rounds synchrones qui imposent un "barrier" d'échange à la fin de chaque itération.

Sur le plan algorithmique, ces stratégies se traduisent par des protocoles de type **round-based synchronization** dans lesquels, à chaque cycle d’itération, les sous-SCN effectuent d’abord leurs mises à jour locales, puis entrent dans une phase de communication où ils échangent les valeurs des pondérations inter-blocs. Ce mécanisme garantit que, au début de chaque itération, toutes les parties du système disposent d’une version cohérente de la matrice ω .

C. Gestion des Conflits et Garanties de Cohérence

Dans un système distribué, l’actualisation concurrente des liens inter-blocs peut générer des **conflits** si plusieurs sous-SCN tentent de modifier la même pondération simultanément. Pour éviter ce genre de problèmes, plusieurs méthodes peuvent être employées :

- L’usage de **verrous** (locks) ou de mécanismes de synchronisation fine qui assurent qu’une seule mise à jour d’une pondération donnée se réalise à la fois. Bien que cette approche garantisse une cohérence forte, elle peut ralentir la dynamique si le nombre de conflits est important.
- La technique du **double-buffer**, dans laquelle la matrice $\omega(t)$ est lue en lecture seule pendant que les mises à jour sont écrites dans une matrice distincte ω_{next} . À la fin de l’itération, un **barrier** de synchronisation permet d’effectuer un swap complet entre ω_{next} et $\omega(t + 1)$, assurant ainsi la cohérence sans verrouillage fin.
- Des **algorithmes lock-free** ou de type **Gossip**, qui permettent une mise à jour asynchrone tout en utilisant des numéros de version ou des timestamps pour réconcilier les divergences. Ces algorithmes, bien qu’efficaces en termes de performances, requièrent une analyse rigoureuse pour garantir la convergence du système.

Le choix entre ces approches dépend fortement de la taille du SCN et de la vitesse de communication entre les sous-SCN. Pour des systèmes de grande envergure, où la latence et la bande passante peuvent devenir critiques, une approche hybride combinant rounds synchrones périodiques et mises à jour asynchrones intermédiaires peut s’avérer optimale.

D. Conclusion

La **communication et le protocole d’échange** des pondérations inter-sous-SCN représentent une composante essentielle pour maintenir la cohérence globale dans un SCN distribué. En segmentant le réseau en sous-SCN, il est indispensable de définir un protocole précis pour l’échange des informations entre les blocs afin que les synergies entre entités situées dans des sous-ensembles différents soient prises en compte correctement. Les solutions vont de la responsabilité unique, où un seul bloc est responsable d’un lien donné, à la copie locale synchronisée via des mécanismes de versioning. Le choix du protocole doit être guidé par les exigences en termes de performance, de latence et de robustesse de convergence. Ainsi, cette communication inter-blocs permet à l’ensemble du système de s’auto-organiser de manière cohérente, tout en assurant une évolutivité et une résilience accrues dans des environnements distribués complexes.

Ces mécanismes de synchronisation et d’échange garantissent que, malgré la dispersion physique ou logique des sous-SCN, le SCN global conserve une dynamique intégrée qui favorise l’émergence de clusters pertinents et la formation d’une organisation réseau robuste.

5.7.1.3. Risque d’Incohérence et Solutions (Synchronisation Épisodique, etc.)

Dans le cadre d’un **Synergistic Connection Network** (SCN) divisé en plusieurs sous-réseaux, la répartition des entités en blocs distincts pose inévitablement le problème de la cohérence des mises à jour des liens inter-blocs. En effet, alors que chaque sous-SCN gère localement la dynamique de ses propres pondérations $\omega_{i,j}$ pour i, j appartenant à un même ensemble \mathcal{V}_p , les connexions reliant des entités de blocs différents, c’est-à-dire des pondérations pour lesquelles $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$ avec $p \neq q$, ne sont plus mises à jour de manière centralisée. Cette situation engendre plusieurs défis, notamment des retards dans la diffusion des mises à jour et des déphasages pouvant conduire à une incohérence globale du SCN.

A. Nature du Risque d'Incohérence

Dans un SCN centralisé, la mise à jour de la matrice $\omega(t)$ est effectuée de façon homogène à l'aide d'une fonction d'itération unique, ce qui permet d'assurer que chaque composante $\omega_{i,j}$ évolue en fonction des mêmes informations à chaque itération. Cependant, dans un SCN distribué, chaque sous-SCN maintient une partie de la matrice, et les liens inter-blocs $\omega_{i,j}$ sont actualisés localement par différents agents ou processus de calcul. Par conséquent, deux sous-SCN peuvent ne pas disposer simultanément de la même version de $\omega_{i,j}$ lorsque des mises à jour sont effectuées. Par exemple, le sous-SCN SCN_p pourrait mettre à jour $\omega_{i,j}$ et obtenir une valeur de 0.65, tandis que le sous-SCN SCN_q n'a pas encore reçu cette mise à jour et maintient une valeur de 0.50. Ce décalage temporel engendre une incohérence, car des corrections contradictoires peuvent être appliquées ensuite par les deux sous-réseaux, chacun se basant sur une information obsolète. Mathématiquement, on peut considérer que l'ensemble des pondérations inter-blocs obéit à une dynamique couplée asynchrone : si la fonction de mise à jour locale dans le sous-SCN p est donnée par

$$\omega_{i,j}^{(p)}(t+1) = F\left(\omega_{i,j}^{(p)}(t), S_{i,j}\right),$$

alors l'absence de synchronisation précise avec $\omega_{i,j}^{(q)}(t)$ peut faire en sorte que, sur le long terme, les versions locales ne convergent pas vers un unique point fixe commun. Un tel phénomène se traduit par des oscillations ou par une divergence progressive des valeurs d'une même liaison au sein des différents blocs.

B. Solutions pour Limiter l'Incohérence

Afin de pallier ces problèmes, plusieurs stratégies peuvent être mises en œuvre, dont les principales sont les synchronisations épisodiques et l'attribution d'une responsabilité unique pour la mise à jour de chaque lien inter-blocs.

Synchronisation Épisodique :

Une approche efficace consiste à introduire des phases de synchronisation périodique. Pendant une phase d'itération locale, chaque sous-SCN effectue ses mises à jour indépendamment, en utilisant les données disponibles à l'instant t . Puis, après un certain nombre d'itérations, tous les sous-SCN se synchronisent en échangeant leurs versions des pondérations inter-blocs. Cette synchronisation peut se faire par le biais d'une barrière (barrier synchronization) qui bloque temporairement l'évolution jusqu'à ce que chaque sous-SCN ait communiqué sa mise à jour. Mathématiquement, si chaque sous-SCN p possède une version $\omega_{i,j}^{(p)}(t)$, alors à la fin d'un cycle de synchronisation, on peut définir la version globale par une moyenne ou par un mécanisme de sélection, par exemple :

$$\omega_{i,j}^{\text{global}}(t) = \frac{1}{M} \sum_{p=1}^M \omega_{i,j}^{(p)}(t),$$

où M est le nombre de sous-SCN impliqués. Ce procédé permet de réduire les écarts et de réaligner les mises à jour avant de reprendre le cycle d'auto-organisation. Le principal avantage de cette méthode est la garantie d'une cohérence périodique, même si elle introduit une latence dans la propagation des mises à jour.

Responsabilité Unique (Bloc Maître) :

Une autre solution consiste à désigner, pour chaque lien inter-blocs $\omega_{i,j}$, un sous-SCN responsable de sa mise à jour. Par exemple, si $\omega_{i,j}$ relie une entité appartenant à \mathcal{V}_p à une entité dans \mathcal{V}_q , on peut assigner la responsabilité de la mise à jour à SCN_p (ou à un module central dédié). Dans ce cas, tous les sous-SCN concernés se réfèrent à la même source de vérité pour la valeur de $\omega_{i,j}$. Cette méthode élimine la redondance des mises à jour et assure que la dynamique s'appuie sur une valeur unique pour chaque lien inter-blocs. Cependant, elle peut créer un goulet d'étranglement si un nombre important de liens est géré par un seul bloc, et nécessite la mise en place de protocoles de consultation et de notification efficaces.

Approches Hybrides et Paramétrage :

Pour limiter les effets des retards et des divergences, il est également possible de moduler les paramètres de la mise à jour locale, tels que le taux d'apprentissage η , pour atténuer les fluctuations. En combinant des méthodes de

synchronisation épisodique avec un ajustement fin des paramètres (par exemple, en introduisant des coefficients d'inertie qui lissent la transition entre les mises à jour locales et globales), on peut obtenir une dynamique plus robuste. Ces ajustements permettent d'atténuer les oscillations dues aux déphasages entre sous-SCN et de favoriser une convergence harmonieuse vers des configurations cohérentes.

Conclusion

Le risque d'incohérence dans un SCN segmenté découle de la nature distribuée des mises à jour inter-blocs, où des décalages temporels ou des différences dans la réception des informations peuvent engendrer des divergences dans les valeurs de $\omega_{i,j}$. Pour pallier ce problème, plusieurs solutions sont envisageables, telles que la synchronisation épisodique par rounds, la désignation d'un bloc maître pour la mise à jour de chaque lien, ou encore des mécanismes hybrides combinant ajustement paramétrique et synchronisation partielle. Ces approches, en garantissant une cohérence périodique ou continue des mises à jour, permettent au SCN global de maintenir une dynamique intégrée et de favoriser l'émergence de clusters cohérents malgré la dispersion des sous-SCN. D'un point de vue mathématique, ces dispositifs s'intègrent dans un cadre de systèmes dynamiques couplés avec décalages, tandis que d'un point de vue ingénierie, ils constituent le compromis nécessaire pour allier scalabilité, performance et stabilité dans un environnement distribué.

5.7.2. Mise en Place d'un “meta-SCN”

Lorsque l'on segmente un SCN en plusieurs sous-SCN (5.7.1), il peut devenir utile, voire essentiel, de **coordonner** ces blocs par un **réseau de plus haut niveau** : un “meta-SCN”. L'idée générale consiste à créer, au-dessus des sous-SCN, une **couche** ou un **nœud intermédiaire** capable d'agréger l'information inter-blocs, de manière à préserver une forme de cohésion globale. Cette section (5.7.2) présente le concept (5.7.2.1), puis aborde la périodicité de mise à jour (5.7.2.2) et, enfin, l'illustration d'une architecture microservices (5.7.2.3).

5.7.2.1. Idée : un SCN global reliant plusieurs sous-SCN, usage d'un “graphon” ou d'un “meta-nœud”

Dans le cadre d'un **Synergistic Connection Network** (SCN) de grande envergure, où le nombre d'entités n est si élevé qu'il devient inenvisageable de gérer une matrice de pondérations ω de taille $n \times n$ dans un système monolithique, il apparaît judicieux de segmenter le réseau en plusieurs **sous-SCN**. Chaque sous-SCN, noté SCN_p pour $p = 1, \dots, m$, gère un sous-ensemble d'entités $\mathcal{V}_p \subset \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Toutefois, pour que le réseau dans son ensemble conserve une cohérence globale, il est indispensable d'introduire une couche d'agrégation supérieure qui relie ces sous-SCN entre elles. Cette couche peut être conceptualisée sous la forme d'un **SCN global** basé sur un **graphon** ou, de façon équivalente, sur un système de **meta-nœuds**.

A. Motivation pour un SCN global

La nécessité de diviser le SCN en sous-ensembles découle principalement de contraintes de **scalabilité** et de **complexité computationnelle**. En effet, lorsque n devient très grand, la complexité mémoire et temporelle d'un traitement centralisé (avec une complexité en $O(n^2)$) devient prohibitive. Par ailleurs, dans des contextes distribués ou hétérogènes (par exemple, des réseaux de robots dispersés géographiquement ou des systèmes multi-agents fonctionnant sur des infrastructures distinctes), il existe des **séparations naturelles** dans les interactions. Ces considérations incitent à organiser le réseau en blocs autonomes. Toutefois, isoler complètement ces blocs conduirait à une perte d'information sur la **synergie** entre des entités appartenant à des sous-ensembles différents. Il est donc souhaitable d'introduire un mécanisme d'agrégation, permettant de condenser l'information relative aux connexions inter-blocs en un ensemble de **pondérations agrégées**. Ce mécanisme est comparable à la notion de **graphon** en théorie des graphes, qui sert d'approximation continue à la matrice d'adjacence d'un grand réseau, ou encore à l'idée de **meta-nœud** dans une approche hiérarchique, où chaque sous-SCN est représenté par un nœud unique dans un graphe de niveau supérieur.

B. Construction d'un Meta-SCN et Modélisation par Graphon

Le **meta-SCN** repose sur l'agrégation des liaisons inter-blocs. Soit $\mathcal{V}_1, \dots, \mathcal{V}_m$ la partition de l'ensemble des entités. Pour chaque paire de blocs (p, q) avec $p \neq q$, on définit une pondération agrégée

$$\hat{\omega}_{p,q} = \Psi(\{\omega_{i,j} : i \in \mathcal{V}_p, j \in \mathcal{V}_q\}),$$

où Ψ représente une **fonction d'agrégation** choisie selon les besoins du système (par exemple, la moyenne, la somme pondérée, ou même le maximum des $\omega_{i,j}$ entre les deux blocs). Cette approche permet de réduire la complexité du SCN global, en passant d'une matrice de dimension $n \times n$ à une matrice $\hat{\omega}$ de dimension $m \times m$, avec $m \ll n$. Dans certains contextes, l'objet continu appelé **graphon** est utilisé pour décrire la limite d'un graphe dense lorsque $n \rightarrow \infty$, fournissant ainsi une représentation continue de la structure d'interaction. Ici, le concept de graphon se traduit par une fonction $W : [0,1]^2 \rightarrow \mathbb{R}^+$ qui, par un processus de rééchelonnement, permet de représenter les interactions entre blocs de manière continue.

C. Avantages et Enjeux de l'Agrégation Inter-blocs

L'utilisation d'un meta-SCN offre plusieurs avantages d'un point de vue à la fois théorique et pratique. D'un point de vue **mathématique**, l'agrégation permet de traiter la dynamique des interconnexions à un niveau supérieur, ce qui simplifie l'analyse de convergence et de stabilité du système global. En effet, l'étude de la dynamique dans un espace de dimension réduite facilite l'identification des attracteurs et des transitions de phase. Du point de vue **ingénierie**, cette approche permet de réduire la charge de communication entre les différents blocs, puisque les sous-SCN n'ont plus à échanger les valeurs de chaque lien inter-blocs individuellement, mais peuvent communiquer des **agrégats** (moyennes, totaux, etc.), ce qui réduit la surcharge en bande passante et accélère les phases de synchronisation.

Toutefois, cette centralisation partielle par un meta-SCN présente également des limites. La perte de granularité dans l'agrégation peut masquer certaines structures fines qui se développent au niveau microscopique des liaisons individuelles. De plus, l'introduction d'un noeud central ou d'un ensemble de meta-noeuds crée potentiellement des points de congestion ou des risques de panne unique, si ces éléments ne sont pas correctement redondants ou distribués.

D. Conclusion sur l'Approche du Meta-SCN

En résumé, la segmentation d'un SCN en sous-ensembles offre une solution élégante aux problèmes de scalabilité et de complexité lorsque le nombre d'entités est extrêmement grand. Le recours à un **meta-SCN** ou à un **graphon** permet d'agrérer l'information des liaisons inter-blocs en une structure de niveau supérieur, facilitant ainsi la coordination globale, la synchronisation et l'analyse des clusters émergents. Cette approche hiérarchique représente un compromis essentiel entre la gestion locale fine des interactions et la nécessité d'une vision globale cohérente du système.

5.7.2.2. Périodicité de la Mise à Jour, Agrégation de Clusters Émergents

Lorsque l'on subdivise un Synergistic Connection Network (SCN) en plusieurs sous-réseaux locaux, il devient indispensable d'établir un mécanisme pour « réunir » ces blocs dans une vision globale cohérente. En effet, chaque sous-SCN, en évoluant selon ses propres dynamiques locales, tend à affiner ses connexions internes. Toutefois, pour qu'un réseau global homogène émerge – où des clusters transcendent les frontières locales se forment – il faut périodiquement synchroniser et agréger les informations relatives aux liens inter-blocs. Ce processus de mise à jour périodique intervient à un niveau mété, permettant de reconstituer la structure globale à partir des agrégats de chaque sous-SCN.

A. Principe d'une Mise à Jour Périodique du Méta-SCN

La démarche consiste à laisser chaque sous-SCN évoluer de manière autonome pendant un certain nombre d'itérations locales, noté T (ou sur un intervalle de temps Δt), pendant lesquelles la règle de mise à jour des pondérations

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

est appliquée de façon répétée pour les liens internes ou déjà établis entre les sous-SCN. À l'issue de cette phase, chaque sous-SCN compile un résumé de ses liaisons inter-blocs, par exemple en calculant une agrégation telle que la

moyenne ou la somme pondérée des pondérations reliant ses entités aux entités d'un autre sous-SCN. Pour deux sous-SCN, \mathcal{V}_p et \mathcal{V}_q , on peut définir la pondération agrégée

$$\widehat{\omega}_{p,q} = \Psi(\{\omega_{i,j} \mid i \in \mathcal{V}_p, j \in \mathcal{V}_q\}),$$

où Ψ désigne une fonction d'agrégation adaptée (par exemple, la moyenne ou la somme). Ainsi, le méta-SCN se construit en tant que graphe de m noeuds (correspondant aux m sous-SCN) et de $\widehat{\omega}_{p,q}$ définissant les liaisons entre ces noeuds. Ce schéma permet de limiter le trafic et la charge de calcul en regroupant les mises à jour inter-blocs de manière périodique plutôt qu'en continu, et de préserver l'autonomie locale des sous-SCN entre ces phases de synchronisation.

B. Méthodologie d'Agrégation et Implications sur la Cohérence Globale

Sur le plan mathématique, l'ensemble du système peut être vu comme évoluant à deux échelles distinctes. Pendant les intervalles entre deux synchronisations globales, la dynamique locale est gouvernée par la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

pour tous les i,j appartenant à un même sous-SCN ou déjà associés par des liaisons inter-blocs précédemment synchronisées. Puis, à chaque période T , une fonction d'agrégation globale, notée G , est appliquée. Formellement, si l'on note $\boldsymbol{\omega}^{\text{global}}(T_k)$ l'ensemble des pondérations inter-blocs au bout du k -ème round global, alors la transition vers le round suivant s'exprime par

$$\boldsymbol{\omega}^{\text{global}}(T_{k+1}) = G(\boldsymbol{\omega}^{\text{global}}(T_k), \{\omega^{(p)}(T_k)\}_{p=1}^m).$$

Cette opération d'agrégation permet de « réinitialiser » la cohérence des valeurs inter-blocs, en compensant les éventuels retards ou divergences accumulées au cours des mises à jour locales. L'équilibre entre la fréquence de synchronisation et la liberté d'évolution locale est déterminant : une synchronisation trop fréquente risque de perturber les dynamiques internes en imposant une correction trop rigide, alors qu'une synchronisation trop espacée peut conduire à des divergences importantes entre les sous-SCN, rendant difficile l'émergence de clusters cohérents au niveau global.

C. Impact sur la Formation des Clusters

Le mécanisme d'agrégation périodique joue un rôle essentiel dans la formation des clusters à l'échelle globale. Chaque sous-SCN, en s'auto-organisant localement, forme des clusters internes qui reflètent la cohésion des entités au sein du bloc. Cependant, la détection de clusters globaux nécessite d'examiner les liaisons inter-blocs : lorsque les agrégats $\widehat{\omega}_{p,q}$ indiquent une forte synergie entre des entités de deux sous-SCN distincts, il devient possible de fusionner les clusters locaux correspondants en un cluster global. Ce processus de fusion, réalisé lors des phases de synchronisation globale, permet d'obtenir une vision hiérarchique du réseau. En d'autres termes, la dynamique du SCN est ainsi organisée en deux niveaux : un niveau local, où les sous-SCN optimisent leurs connexions, et un niveau global, où un méta-SCN agrège ces connexions pour détecter des structures de clusters transcendant les frontières locales. Ce couplage multi-échelle favorise une convergence plus robuste et une meilleure adaptation aux variations de la synergie inter-blocs.

D. Conclusion

La périodicité de la mise à jour et l'agrégation des clusters émergents constituent des éléments fondamentaux pour maintenir la cohérence d'un SCN distribué. En permettant aux sous-SCN de se stabiliser localement pendant une période déterminée avant d'intégrer leurs informations au niveau global, on obtient une structure hiérarchique capable de concilier l'efficacité des dynamiques locales avec une vision globale cohérente du réseau. Cette approche, articulée autour d'une synchronisation périodique, assure que les modifications des liaisons inter-blocs soient regroupées et traitées de manière contrôlée, réduisant ainsi les risques d'oscillations et d'incohérences, tout en favorisant l'émergence de clusters significatifs à l'échelle macro.

```
import numpy as np
import matplotlib.pyplot as plt
```

```

import seaborn as sns

# Paramètres de la dynamique locale
eta = 0.05      # Taux d'apprentissage
tau = 1.0        # Coefficient de décroissance
n_total = 200    # Nombre total d'entités
m = 4            # Nombre de sous-SCN (blocs)
T_max = 300      # Nombre total d'itérations locales
T_period = 20    # Période de synchronisation globale (en itérations)

# Distribution homogène des entités entre les blocs
n_block = n_total // m

# Initialisation de la matrice globale omega (dense)
omega = np.random.uniform(0.0, 0.01, (n_total, n_total))
np.fill_diagonal(omega, 0.0) # Pas de liaison auto-connectée

# Initialisation d'une matrice S de synergie aléatoire (pour simplifier)
S = np.random.uniform(0.2, 0.8, (n_total, n_total))
S = (S + S.T) / 2 # Rendre la matrice symétrique
np.fill_diagonal(S, 0.0)

# Stockage des agrégats inter-blocs pour visualisation
meta_history = []

def update_local(omega, S, eta, tau):
    """Mise à jour additive locale pour l'ensemble du réseau."""
    return omega + eta * (S - tau * omega)

def aggregate_inter_block(omega, m, n_block):
    """Agrège les valeurs inter-blocs en calculant la moyenne pour chaque paire de blocs."""
    meta_matrix = np.zeros((m, m))
    for p in range(m):
        for q in range(m):
            # Indices des entités dans le bloc p et q
            idx_p = slice(p * n_block, (p + 1) * n_block)
            idx_q = slice(q * n_block, (q + 1) * n_block)
            # Si p == q, on peut ignorer ou mettre NaN car ce sont des liens internes
            if p == q:
                meta_matrix[p, q] = np.nan
            else:
                # Calcul de la moyenne des omega inter-blocs
                meta_matrix[p, q] = np.mean(omega[idx_p, idx_q])
    return meta_matrix

# Pour visualiser la dynamique globale, on va stocker la matrice meta à chaque période
meta_list = []

# Simulation de la dynamique locale avec synchronisation périodique
for t in range(T_max):
    # Mise à jour locale
    omega = update_local(omega, S, eta, tau)

    # À chaque période T, réaliser l'agrégation inter-blocs
    if (t + 1) % T_period == 0:
        meta = aggregate_inter_block(omega, m, n_block)
        meta_list.append(meta)

```

```

meta_list.append(meta)

# Affichage graphique de la dynamique du meta-SCN
plt.figure(figsize=(8, 6))
# On affiche la dernière matrice meta agrégée
sns.heatmap(meta_list[-1], annot=True, fmt=".2f", cmap="viridis",
            cbar_kws={'label': 'Valeur moyenne des liens inter-blocs'})
plt.title("Matrice Agrégée des Liaisons Inter-blocs après Synchronisation Globale")
plt.xlabel("Bloc q")
plt.ylabel("Bloc p")
plt.show()

# Affichage de l'évolution d'un exemple de lien inter-blocs
# On choisit, par exemple, la liaison entre le bloc 1 et le bloc 2
link_history = [meta[0, 1] for meta in meta_list]

plt.figure(figsize=(8, 4))
plt.plot(np.arange(len(link_history)) * T_period, link_history, marker='o')
plt.title("Évolution de la Valeur Moyenne des Liens Inter-blocs (Bloc 1 à Bloc 2)")
plt.xlabel("Itérations globales")
plt.ylabel("Valeur moyenne de  $\hat{\omega}_{1,2}$ ")
plt.grid(True)
plt.show()

```

Dans cette implémentation, le **SCN** global est simulé par une matrice ω qui évolue selon la règle additive locale, avec une mise à jour effectuée à chaque itération. Le réseau est divisé en m blocs égaux, chacun correspondant à un sous-SCN. Toutes les TTT itérations, la fonction d'agrégation calcule la moyenne des pondérations inter-blocs pour chaque paire de blocs, constituant ainsi une matrice de niveau supérieur $\hat{\omega}$. Cette matrice agrégée fournit une vue globale sur la dynamique des liens inter-blocs et permet d'identifier visuellement, via une heatmap, les zones où la synergie collective est particulièrement forte ou faible. Le graphe de l'évolution d'un lien inter-blocs (entre le bloc 1 et le bloc 2 dans l'exemple) illustre la manière dont la synchronisation périodique permet de stabiliser et de monitorer la formation de clusters à un niveau macro.

En résumé, la **périodicité** de mise à jour et l'agrégation des clusters émergents jouent un rôle essentiel dans la coordination des sous-SCN. Cette approche hiérarchique assure que la dynamique locale des sous-réseaux se consolide régulièrement dans une structure globale cohérente, facilitant ainsi l'émergence de clusters transcendant les frontières initiales et permettant une meilleure adaptabilité et résilience du SCN global.

5.7.2.3. Cas d'une Architecture Microservices : Chaque Service = Sous-SCN

Dans le cadre d'un **Synergistic Connection Network** (SCN) déployé à grande échelle, la gestion centralisée d'une matrice de pondérations $\{\omega_{i,j}\}$ peut rapidement devenir un goulet d'étranglement, tant au niveau de la **mémoire** que du **calcul**. Pour surmonter ces limitations, il est judicieux de segmenter le SCN en plusieurs sous-réseaux autonomes, chacun étant hébergé dans un **microservice** dédié. Chaque microservice, ou **sous-SCN**, gère un sous-ensemble \mathcal{V}_p d'entités et exécute localement sa propre dynamique de mise à jour des poids. Par ailleurs, une coordination hiérarchique est instaurée pour synchroniser les informations entre les différents microservices, permettant ainsi d'obtenir une vue globale cohérente du réseau. Cette approche modulaire offre de nombreux avantages en termes de **scalabilité**, **résilience** et **flexibilité** architecturale.

A. Conception d'une Architecture Microservices pour un SCN

L'idée fondamentale est de découper l'ensemble des entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ en plusieurs groupes disjoints $\mathcal{V}_1, \dots, \mathcal{V}_m$ de taille plus ou moins homogène, de sorte que chaque groupe soit géré par un microservice dédié, désigné par SCN_p pour

$p = 1, \dots, m$. Chaque microservice se charge alors de stocker la matrice locale $\omega^{(p)}$, qui représente les pondérations internes entre les entités appartenant à \mathcal{V}_p . Cette segmentation permet d'éviter la complexité $\mathcal{O}(n^2)$ d'une unique matrice globale, en la divisant en plusieurs blocs de taille réduite, ce qui facilite leur stockage (potentiellement en format *sparse*) et accélère la mise à jour des poids par itération.

L'architecture microservices repose sur le principe de **découplage fonctionnel** : chaque microservice est un module indépendant responsable d'un sous-SCN, et il expose une **API** permettant la communication avec les autres microservices. Par exemple, chaque service peut publier périodiquement les agrégats des liaisons inter-blocs, ou répondre à des requêtes concernant la configuration de ses clusters locaux. Ce mécanisme permet également de gérer la **latence** et d'allouer les ressources de calcul de manière optimale, grâce à une répartition horizontale de la charge. En effet, si le nombre total d'entités n est trop élevé, chaque microservice peut être dimensionné indépendamment ou même subdivisé en services plus petits, assurant ainsi une **scalabilité horizontale** aisée.

B. Considérations Mathématiques et Modélisation

Mathématiquement, si l'on note $\omega_{i,j}^{(p)}(t)$ la pondération entre deux entités i et j au sein du sous-SCN SCN_p , la dynamique locale suit typiquement une équation de mise à jour du type

$$\omega_{i,j}^{(p)}(t+1) = \omega_{i,j}^{(p)}(t) + \eta [S^{(p)}(i,j) - \tau \omega_{i,j}^{(p)}(t)],$$

où $S^{(p)}(i,j)$ représente la synergie locale entre les entités de \mathcal{V}_p . Pour les liaisons inter-blocs, c'est-à-dire pour $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$ avec $p \neq q$, on définit un agrégat global qui peut être noté

$$\hat{\omega}_{p,q} = \Psi(\{\omega_{i,j}\}_{i \in \mathcal{V}_p, j \in \mathcal{V}_q}),$$

où Ψ est une fonction d'agrégation (par exemple, la moyenne ou la somme pondérée). Ce sur-graphe, ou **meta-SCN**, permet de représenter la **cohésion** inter-blocs et de prendre des décisions au niveau macro, telles que la fusion de clusters ou la réorganisation des microservices en cas de forte synergie entre eux. Ce modèle mathématique traduit ainsi la hiérarchie du système : la dynamique locale à l'intérieur de chaque microservice et la dynamique globale résultant de l'agrégation des informations inter-blocs.

C. Avantages et Limites de l'Approche Microservices

Du point de vue de l'**ingénierie logicielle**, l'architecture microservices offre plusieurs avantages notables. Premièrement, elle permet un **déploiement indépendant** : chaque sous-SCN peut être développé, testé, mis à jour et déployé sans perturber l'ensemble du réseau. Deuxièmement, cette approche favorise une **modularité** accrue, puisque chaque service est spécialisé dans la gestion d'un sous-ensemble d'entités, ce qui facilite la maintenance et l'extension du système. Troisièmement, la **scalabilité horizontale** est grandement améliorée, car il est possible d'allouer des ressources supplémentaires à un microservice surchargé ou de le subdiviser si nécessaire. Toutefois, cette approche présente également des défis, notamment en ce qui concerne la **coordination** entre les microservices et la gestion des liaisons inter-blocs. La communication entre services doit être soigneusement orchestrée pour éviter les incohérences et assurer la convergence globale du SCN.

Conclusion

L'**approche microservices** pour la gestion d'un SCN distribué se révèle être une solution robuste pour faire face à la complexité induite par un grand nombre d'entités. En assignant à chaque microservice la responsabilité de gérer un sous-SCN local, le système bénéficie d'une meilleure **scalabilité**, d'une plus grande **résilience** en cas de défaillance d'un service individuel, et d'une **modularité** facilitant le développement et la maintenance. Le recours à un meta-SCN pour l'agrégation des synergies inter-blocs permet de conserver une vue globale cohérente du réseau, essentielle pour la formation de clusters significatifs et pour la prise de décisions stratégiques à l'échelle du système. Cette structure hiérarchique, appuyée par des protocoles de communication adaptés, constitue un compromis efficace entre la gestion locale détaillée et la coordination globale nécessaire à un SCN de grande envergure.

5.7.3. Ingénierie Logicielle

Même s'il s'agit d'un ouvrage à dominante mathématique ou théorique, un **SCN** (Synergistic Connection Network) de grande envergure, distribué en plusieurs sous-SCN (5.7.1) et éventuellement chapeauté par un meta-SCN (5.7.2), requiert une **organisation logicielle** suffisamment robuste et modulaire. Le chapitre 5.7.3 vise à souligner les principales approches d'ingénierie employées pour structurer et maintenir un système de ce type, dans la perspective d'une mise en œuvre pérenne ou d'un cadre de recherche-application.

5.7.3.1. Patterns de Conception (Observer, Strategy, etc.) pour Moduler l'Implémentation

La mise en œuvre d'un **Synergistic Connection Network** (SCN), tel que défini par ses équations d'évolution de pondération $\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j), \dots)$, nécessite non seulement une compréhension rigoureuse de sa dynamique mathématique, mais également une traduction concrète dans un environnement logiciel modulaire et évolutif. Dans ce contexte, l'utilisation de **patterns de conception** (design patterns) issus du génie logiciel orienté objet – notamment les patterns **Strategy** et **Observer** – permet de découpler la logique de mise à jour, la gestion des synergies, et le suivi des événements, tout en facilitant la maintenance et l'extension du système.

A. Contexte et Problématique

Le cœur mathématique d'un SCN se résume souvent à une équation générée telle que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

où :

- $\omega_{i,j}(t)$ représente la pondération entre les entités \mathcal{E}_i et \mathcal{E}_j à l'itération t ,
- $S(i,j)$ est la **synergie** entre ces entités,
- η est le **taux d'apprentissage** et τ le coefficient de décroissance.

Cependant, dans une implémentation réelle, cette formule n'est qu'un composant d'un système beaucoup plus vaste qui doit aussi :

- **Calculer** la synergie $S(i,j)$ de manière adaptée aux types d'entités,
- **Adapter** la règle de mise à jour à différentes variantes (par exemple, additive, multiplicative, avec termes stochastiques ou d'inhibition),
- **Observer** et **notifier** les changements de l'état du réseau pour permettre une analyse en temps réel ou une adaptation dynamique.

Pour répondre à ces enjeux, des **patterns de conception** offrent un cadre permettant de modulariser ces préoccupations. Ils permettent d'isoler le « **quoi** » (la formule de mise à jour et le calcul de S) du « **comment** » (la manière dont le code est organisé, exécuté et surveillé). Dans ce cadre, deux patterns se distinguent particulièrement :

- **Strategy** : Pour encapsuler la variation des algorithmes de mise à jour dans des classes interchangeables.
- **Observer** : Pour mettre en place un système de notifications qui informe d'un changement dans l'état des pondérations, sans alourdir le module central.

B. Pattern Strategy : Flexibilité dans la Mise à Jour

Le pattern **Strategy** consiste à définir une **interface** abstraite pour un algorithme, permettant ainsi d'encapsuler différentes implémentations concrètes qui pourront être interchangées au moment de l'exécution. Dans le cas du SCN, cela signifie créer une interface pour la mise à jour de ω qui se présente, par exemple, sous la forme :

$$\text{applyUpdate}(\omega_{i,j}(t), S(i,j), \text{params}) \rightarrow \omega_{i,j}(t + 1).$$

La fonction F qui intervient dans l'équation de mise à jour peut ainsi être décomposée en modules distincts, chacun implémentant une stratégie particulière. Par exemple :

- **AdditiveUpdateRule** :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

- **MultiplicativeUpdateRule** :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) \times [1 + \eta (S(i,j) - \tau \omega_{i,j}(t))].$$

En définissant une interface **UpdateRule** (par exemple, en Python ou en Java), le module central du SCN peut appeler une méthode *applyUpdate* sans se préoccuper des détails de l'algorithme utilisé. Ce découplage facilite l'expérimentation et la comparaison entre différentes dynamiques d'apprentissage.

C. Pattern Observer : Suivi et Notification des Changements

Le pattern **Observer** permet d'établir un mécanisme de **notification** dans lequel un objet (le **sujet**) informe un ensemble d'**observateurs** lorsqu'un événement pertinent se produit. Dans le cadre d'un SCN, le sujet peut être le module qui gère la matrice ω et qui, à chaque itération ou à chaque changement significatif, émet une notification indiquant l'évolution des poids.

Cette approche est utile pour plusieurs raisons :

- Elle permet de **déclencher** des actions supplémentaires, comme l'actualisation de visualisations ou l'extraction de clusters, sans alourdir le code de la mise à jour.
- Elle offre une **décorrélation** entre la logique de mise à jour et les modules d'analyse, ce qui facilite la maintenance.
- Elle permet d'implémenter des **règles adaptatives**, par exemple pour ajuster dynamiquement les paramètres η ou τ en fonction des fluctuations observées.

En pratique, le module central du SCN peut inclure une méthode pour **enregistrer** des observateurs, qui seront appelés par exemple via une méthode *notifyObservers()* dès que l'état de ω est mis à jour.

D. Intégration des Patterns dans une Architecture Modulaire

L'architecture globale du SCN se structure autour d'un **noyau** central qui gère la matrice ω et orchestre la boucle d'auto-organisation. Ce noyau fait appel aux modules suivants :

- Un **UpdateModule** qui, via le pattern Strategy, applique la règle de mise à jour appropriée.
- Un **SynergyModule** qui fournit les valeurs de $S(i,j)$ à partir des représentations des entités.
- Un **ObserverModule** qui, via le pattern Observer, capte les changements dans ω et déclenche des actions (visualisation, logging, ajustement des paramètres).

Mathématiquement, la mise à jour d'un poids peut être réécrite comme :

$$\omega_{i,j}(t + 1) = F(\omega_{i,j}(t), S(i,j), \eta, \tau),$$

où F est encapsulée dans la *Strategy* utilisée, et où le noyau se contente de parcourir la matrice et d'appliquer cette fonction à chaque composante. Les observateurs, quant à eux, reçoivent la notification du changement et effectuent des traitements complémentaires.

E. Conclusion

L'adoption des patterns **Strategy** et **Observer** permet de modulariser la mise en œuvre d'un SCN en séparant la logique de mise à jour des poids et le suivi des événements. Cette modularisation confère une grande flexibilité au système, permettant d'expérimenter différentes règles de mise à jour, d'intégrer facilement des fonctionnalités supplémentaires (comme l'inhibition, la saturation ou le recuit stochastique) et d'assurer une surveillance continue de l'évolution de la dynamique auto-organisée. Du point de vue mathématique, le modèle reste inchangé – la fonction F détermine toujours l'évolution des $\omega_{i,j}$ – tandis que l'architecture logicielle garantit que cette dynamique peut être adaptée, étendue et maintenue dans des environnements complexes.

Exemple Pseudo-implémentation en Python

Voici une implémentation simplifiée en Python qui illustre l'utilisation des patterns **Strategy** et **Observer** dans le cadre de la mise à jour des poids d'un SCN :

```
import numpy as np
import matplotlib.pyplot as plt

# Définition des paramètres généraux
params = {'eta': 0.05, 'tau': 1.0}
T_max = 100 # Nombre d'itérations
n = 10      # Nombre d'entités

# Création d'une matrice de synergie S (symétrique, sans auto-connexion)
np.random.seed(42)
S = np.random.uniform(0.5, 1.0, (n, n))
for i in range(n):
    S[i, i] = 0.0
    for j in range(i + 1, n):
        S[j, i] = S[i, j]
        S[j, j] = S[i, j]

# Initialisation de la matrice de poids w
w = np.zeros((n, n))

# -----
# Pattern Strategy: Interface de mise à jour
# -----
class UpdateRule:
    def apply(self, old_weight, synergy, params):
        raise NotImplementedError

class AdditiveUpdateRule(UpdateRule):
    def apply(self, old_weight, synergy, params):
        return old_weight + params['eta'] * (synergy - params['tau'] * old_weight)

class MultiplicativeUpdateRule(UpdateRule):
    def apply(self, old_weight, synergy, params):
        return old_weight * (1 + params['eta'] * (synergy - params['tau'] * old_weight))

# Choix de la stratégie de mise à jour
update_rule = AdditiveUpdateRule() # On peut changer pour MultiplicativeUpdateRule()

# -----
# Pattern Observer: Système de notification
# -----
class Observer:
    def update(self, w, t):
```

```

    raise NotImplementedError

class WeightObserver(Observer):
    def __init__(self):
        self.mean_history = []
        self.max_history = []

    def update(self, w, t):
        self.mean_history.append(np.mean(w))
        self.max_history.append(np.max(w))

# Instanciation de l'observer
observer = WeightObserver()

# Historique pour visualisation
w_history = [w.copy()]

# Boucle d'itération principale du SCN
for t in range(T_max):
    w_next = np.copy(w)
    for i in range(n):
        for j in range(n):
            if i != j:
                w_next[i, j] = update_rule.apply(w[i, j], S[i, j], params)
    w = w_next
    w_history.append(w.copy())
    observer.update(w, t)

# Visualisation des statistiques de poids au fil des itérations
iterations = range(T_max)
plt.figure(figsize=(10, 6))
plt.plot(iterations, observer.mean_history, label="Moyenne des poids")
plt.plot(iterations, observer.max_history, label="Maximum des poids")
plt.xlabel("Itérations")
plt.ylabel("Valeur des poids")
plt.title("Évolution des poids dans le SCN avec mise à jour additive")
plt.legend()
plt.grid(True)
plt.show()

# Visualisation finale de la matrice de poids sous forme de heatmap
plt.figure(figsize=(8, 6))
plt.imshow(w, cmap="viridis", interpolation="nearest")
plt.colorbar(label="Valeur de  $\omega$ ")
plt.title("Heatmap finale de la matrice de poids  $\omega$ ")
plt.xlabel("Index j")
plt.ylabel("Index i")
plt.show()

```

Explications Complémentaires

Dans cette implémentation, le pattern **Strategy** est incarné par l'interface *UpdateRule* et ses implémentations concrètes, ici *AdditiveUpdateRule* (et éventuellement *MultiplicativeUpdateRule*). Le module central de mise à jour du SCN se contente d'appeler la méthode *apply* pour chaque paire (i, j) , ce qui permet d'abstraire le choix de la dynamique

de mise à jour. Cela facilite grandement l’expérimentation, car il suffit de changer l’instance de la stratégie pour tester différentes formules.

Le pattern **Observer** est utilisé via la classe *WeightObserver* qui, à chaque itération, enregistre des statistiques globales (moyenne et maximum des poids). Ces données sont ensuite exploitées pour visualiser l’évolution de la dynamique, offrant ainsi une **vision** en temps réel des comportements émergents.

L’approche adoptée garantit que la **logique mathématique** – la formule de mise à jour – reste distincte de l’**orchestration** logicielle et de la surveillance, assurant ainsi une architecture modulaire et facilement extensible. Les visualisations générées par les graphiques fournissent une illustration claire de la convergence (ou des oscillations éventuelles) des pondérations, permettant ainsi de valider l’auto-organisation du SCN.

En résumé, l’utilisation conjointe des patterns **Strategy** et **Observer** permet de modulariser l’implémentation d’un SCN de manière flexible, tout en assurant que la dynamique auto-organisée reste fidèle aux principes mathématiques sous-jacents, et que le système peut être facilement monitoré et étendu dans un cadre logiciel moderne.

5.7.3.2. Approches NoSQL, BDD Orientée Graphe (Neo4j, Titan) pour Stocker ω

La gestion de la **persistante** des pondérations $\omega_{i,j}$ dans un Synergistic Connection Network (SCN) à grande échelle pose des défis majeurs tant sur le plan **mémoire** que sur celui des **requêtes complexes**. Lorsque le nombre d’entités n devient très important, la matrice $\Omega \in \mathbb{R}^{n \times n}$ contenant l’ensemble des $\omega_{i,j}$ peut atteindre une taille de l’ordre de $O(n^2)$. Dans ces conditions, un simple stockage en mémoire vive ou sous forme de fichiers plats ne suffit pas pour assurer des mises à jour efficaces, l’extraction rapide de sous-ensembles pertinents ou l’analyse ultérieure de la structure globale du réseau.

A. Choix d’une Base de Données NoSQL ou Graph

Les **bases de données relationnelles** traditionnelles, malgré leur maturité, peinent à gérer de très grandes quantités de données lorsqu’il s’agit de modéliser des graphes denses. Par exemple, le stockage d’une table *Edges(source, target, weight)* dans une base SQL génère rapidement des contraintes en termes de bande passante et de temps de réponse pour des requêtes complexes telles que la recherche des top- k liens sortants ou la détection de communautés.

Les **bases NoSQL** offrent une solution alternative en se focalisant sur la scalabilité horizontale et la distribution des données. Parmi celles-ci, les bases **orientées graphe** — telles que **Neo4j** et **JanusGraph/Titan** — se distinguent par leur capacité native à représenter directement les **nœuds** et les **arêtes**. Dans ce paradigme, chaque entité \mathcal{E}_i est modélisée comme un nœud, et chaque pondération $\omega_{i,j}$ apparaît comme une arête dotée d’un attribut « weight ». Mathématiquement, on peut assimiler cette représentation à une fonction

$$\omega: V \times V \rightarrow \mathbb{R}^+,$$

où V désigne l’ensemble des nœuds. Les langages de requête dédiés, tels que **Cypher** pour Neo4j ou **Gremlin** pour JanusGraph, permettent d’extraire aisément des sous-graphes, de calculer des agrégats ou d’appliquer des algorithmes de détection de communautés sur le graphe persistant.

B. Intégration dans la Dynamique du SCN

Du point de vue de l’**auto-organisation**, le SCN évolue selon une dynamique décrite par la relation générale

$$\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j), \text{paramètres}),$$

où F représente la fonction de mise à jour (qui peut être additive, multiplicative ou comporter des termes stochastiques). Dans un contexte de grande échelle, il devient impératif de persister les valeurs de $\omega_{i,j}$ afin de :

- **Conserver** un historique des évolutions pour des analyses ultérieures (audit, vérification de convergence ou détection d’événements anormaux),
- **Exécuter** des requêtes en temps réel pour extraire les **clusters** ou les **top- k** liens, et

- **Distribuer** la charge en exploitant les capacités de partitionnement et de réPLICATION offertes par les bases NoSQL.

L'intégration de la dynamique dans une base orientée graphe consiste donc à synchroniser, de manière périodique ou en temps réel, la matrice Ω avec la structure persistée. Pour cela, des mécanismes de *batch update* ou de mise à jour épisodique sont mis en place afin de limiter le nombre d'opérations d'écriture, souvent regroupées par des algorithmes d'agrégation sur les liens inter-blocs.

C. Approches de Stockage et de Requête

L'utilisation d'une base **orientée graphe** présente plusieurs avantages du point de vue algorithmique et mathématique. Premièrement, la représentation d'un SCN dans un graphe $G = (V, E)$ se traduit naturellement par :

- Chaque nœud $v \in V$ représente une entité \mathcal{E}_i ,
- Chaque arête $e = (u, v) \in E$ est associée à une pondération $\omega_{u,v}$.

Cette correspondance directe facilite l'implémentation de **requêtes complexes**, telles que la recherche des voisins d'un nœud ayant un poids supérieur à un seuil donné, ou l'extraction des chemins les plus courts pondérés par ω . Par exemple, une requête Cypher dans Neo4j pour récupérer les liens les plus forts pourrait être formulée ainsi :

```
MATCH (i)-[r]->(j) WHERE r.weight > θ RETURN i, j, r.weight ORDER BY r.weight DESC LIMIT k,
```

ce qui permet de visualiser la structure des **clusters** émergents et d'identifier les **communautés** présentes dans le réseau.

De plus, l'aspect distribué des bases telles que **JanusGraph** (souvent couplée à des backends comme Cassandra ou HBase) permet de gérer efficacement des graphes contenant des millions de nœuds et d'arêtes, en assurant une **scalabilité horizontale** ainsi qu'une haute disponibilité.

D. Avantages et Contraintes des Approches NoSQL/Graph

D'un point de vue **mathématique**, l'utilisation d'une base orientée graphe permet de traiter les pondérations $\omega_{i,j}$ comme des composantes d'un opérateur linéaire défini sur l'espace des nœuds, facilitant l'application d'algorithmes de partitionnement ou de clustering. Les **algorithmes de détection de communautés** (par exemple, Louvain ou Label Propagation) peuvent directement exploiter la structure du graphe pour déterminer des clusters à différents niveaux de granularité.

Sur le plan **ingénierie**, l'utilisation d'une BDD NoSQL/Graph offre la possibilité de distribuer le stockage et le calcul, en permettant un **sharding** du graphe sur plusieurs machines. Toutefois, cette approche nécessite une phase de conception soignée pour définir les schémas d'indexation, les stratégies de réPLICATION et les protocoles de mise à jour afin d'éviter des incohérences ou des retards importants dans les opérations d'écriture.

Par ailleurs, la nature même d'un graphe persistant implique que l'on puisse réaliser des **snapshots** ou des **logs** d'évolution, ce qui est particulièrement utile pour l'analyse rétroactive de la dynamique du SCN et la validation des hypothèses théoriques sur la convergence.

Conclusion

L'adoption d'approches NoSQL, et plus particulièrement l'utilisation de bases de données orientées graphe telles que **Neo4j** ou **JanusGraph/Titan**, représente une solution adaptée et pragmatique pour le stockage et la consultation des pondérations $\omega_{i,j}$ dans un SCN à grande échelle. Cette stratégie permet non seulement de modéliser naturellement le réseau sous forme de graphe (où chaque entité est un nœud et chaque connexion une arête), mais aussi de bénéficier d'algorithmes de requête et de détection de communautés optimisés pour ce type de données. Par ailleurs, ces bases offrent une scalabilité horizontale et une flexibilité d'intégration, ce qui est essentiel pour la gestion d'un SCN réparti sur plusieurs machines ou clusters. Sur le plan mathématique, la structure du graphe permet d'appliquer des méthodes analytiques et d'optimisation directement sur la matrice de pondérations, tout en assurant la cohérence et la persistance des données. Ainsi, ces approches facilitent l'extraction de clusters, la surveillance de l'évolution des liens et l'analyse

globale de la dynamique auto-organisée du SCN, en répondant aux exigences de performance et de flexibilité d'un système distribué moderne.

5.7.3.3. Équilibrage de Charge (Sharding de la Matrice ω)

Dans un **Synergistic Connection Network** (SCN) à très grande échelle, la matrice de pondérations Ω qui regroupe les connexions $\omega_{i,j}$ entre toutes les entités peut rapidement devenir volumineuse, avec une complexité mémoire de l'ordre de $O(n^2)$ lorsque le nombre d'entités n augmente. Pour pallier ce problème, il est souvent nécessaire de recourir à des techniques d'**équilibrage de charge** par le *sharding*, c'est-à-dire en partitionnant la matrice en fragments plus petits répartis sur plusieurs nœuds ou serveurs. Cette approche permet non seulement de réduire la charge de calcul et la pression sur la mémoire, mais également de réduire la latence lors de l'accès aux données et de faciliter la mise à l'échelle horizontale du système.

Sur le plan **mathématique**, on peut formaliser le sharding comme suit. Soit $\Omega \in \mathbb{R}^{n \times n}$ la matrice complète des pondérations, et supposons que nous partitionnons l'ensemble des entités $V = \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ en m sous-ensembles disjoints, notés $\mathcal{V}_1, \mathcal{V}_2, \dots, \mathcal{V}_m$, de sorte que chaque sous-SCN SCN_p gère la matrice locale $\Omega^{(p)}$ de dimensions $|\mathcal{V}_p| \times |\mathcal{V}_p|$. Dans le cas idéal d'un partitionnement homogène, chaque bloc contient environ $\frac{n}{m}$ entités, et la taille de chaque matrice locale est alors de l'ordre de $O\left(\left(\frac{n}{m}\right)^2\right)$. Ainsi, le coût global de stockage pour les données internes aux shards est réduit à

$$m \times O\left(\left(\frac{n}{m}\right)^2\right) = O\left(\frac{n^2}{m}\right),$$

ce qui, pour un m suffisamment grand, constitue une amélioration significative par rapport à une gestion centralisée.

Le partitionnement par sharding améliore également l'**efficacité des requêtes**. Par exemple, lorsque l'on cherche à extraire les k plus fortes connexions pour une entité donnée, il est beaucoup plus rapide de consulter la ligne correspondante dans le shard local que d'effectuer un parcours exhaustif sur une matrice globale gigantesque. De plus, l'**accès local** aux données réduit la latence réseau et permet une meilleure exploitation du cache dans les environnements distribués.

Toutefois, le sharding introduit la problématique des **liaisons inter-shards**. Les connexions $\omega_{i,j}$ reliant des entités situées dans des shards différents (par exemple, $i \in \mathcal{V}_p$ et $j \in \mathcal{V}_q$ avec $p \neq q$) nécessitent une gestion particulière. La communication entre shards peut être réalisée de deux manières principales :

- Dans une première approche, chaque shard est responsable de stocker et de mettre à jour les liens pour lesquels il détient la ligne correspondante. Par exemple, on peut définir que le shard auquel appartient l'entité \mathcal{E}_i est le maître de toutes les pondérations $\omega_{i,j}$ pour tout j . Les autres shards doivent alors interroger ce maître pour obtenir la version la plus récente de ces liens.
- Dans une seconde approche, chaque shard maintient une **copie locale** des liens inter-shards, et un mécanisme de synchronisation périodique (par exemple, via des barrières synchrones ou des protocoles de type *gossip*) est mis en œuvre afin d'harmoniser les divergences éventuelles.

Sur le plan **ingénierie**, la méthode de partitionnement peut être réalisée via un **partitionnement par hachage** ou par des algorithmes de **graph partitioning** tels que METIS ou Kernighan–Lin. Le partitionnement par hachage répartit les entités de manière uniforme en utilisant, par exemple, l'opération $p = \text{hash}(i) \bmod m$. Le partitionnement par graphes, quant à lui, tente de minimiser le nombre de liens inter-shards en regroupant les entités fortement connectées dans le même fragment.

Les avantages du sharding sont multiples : il permet une **scalabilité horizontale** en répartissant la charge sur plusieurs serveurs, améliore la **localité** des données pour des requêtes plus rapides, et augmente la **résilience** du système, puisque la défaillance d'un shard n'entraîne pas nécessairement la panne de l'ensemble du réseau. Néanmoins, il faut prendre

garde aux risques d'**incohérence** entre les shards, ainsi qu'à la complexité supplémentaire induite par la gestion des mises à jour inter-shards.

En conclusion, l'équilibrage de charge par sharding de la matrice ω est une solution incontournable pour gérer des SCN de très grande échelle. La fragmentation de la matrice en blocs plus petits permet non seulement de réduire le coût mémoire et le temps de calcul par itération, mais aussi d'améliorer la réactivité du système en facilitant l'accès aux données locales. Cependant, cette approche nécessite une attention particulière à la synchronisation des mises à jour inter-shards et à la gestion de la communication entre les différentes partitions, afin de garantir la cohérence globale du SCN.

Exemple de mise en œuvre en Python (sans visualisation graphique)

Bien que nous nous concentrions ici sur l'explication théorique, il est pertinent d'illustrer brièvement comment l'on pourrait organiser le sharding de la matrice ω dans un environnement Python. Supposons que nous partitionnions nos entités en m shards et que nous stockions chaque shard sous forme d'un tableau NumPy. Une implémentation simple pourrait ressembler à :

```
import numpy as np

def initialize_shard(n, m, shard_index):
    """Initialise le shard pour le sous-ensemble d'entités du shard_index.
    n : nombre total d'entités
    m : nombre de shards
    shard_index : index du shard courant (0 <= shard_index < m)
    """
    # On suppose un partitionnement uniforme : chaque shard gère n/m entités
    entities_per_shard = n // m
    # Dimensions du shard: (entities_per_shard x n)
    # On stocke uniquement les lignes correspondantes à ce shard
    shard = np.zeros((entities_per_shard, n))
    return shard

def update_shard(shard, S, eta, tau, gamma):
    """Applique la règle de mise à jour sur le shard.
    S : matrice complète de synergie (n x n)
    shard : matrice du shard courant (entities_per_shard x n)
    Cette fonction met à jour chaque poids selon la formule additive avec inhibition.
    """
    entities_per_shard, n = shard.shape
    new_shard = np.copy(shard)
    for i in range(entities_per_shard):
        global_i = i # Dans un partitionnement uniforme, global_i = shard_index * entities_per_shard + i
        for j in range(n):
            # Calcul de la mise à jour additive
            delta = eta * (S[global_i, j] - tau * shard[i, j])
            # Ajout du terme d'inhibition : somme sur k != j
            inhibition = gamma * (np.sum(shard[i, :] - shard[i, j]))
            new_shard[i, j] = shard[i, j] + delta - inhibition
    return new_shard

# Paramètres
n = 1000      # nombre total d'entités
m = 10        # nombre de shards
eta = 0.05
tau = 1.0
gamma = 0.01
```

```

# Initialisation de la matrice de synergie S (pour l'exemple, valeurs aléatoires)
np.random.seed(42)
S = np.random.uniform(0, 1, (n, n))

# Initialisation des shards
shards = [initialize_shard(n, m, p) for p in range(m)]

# Mise à jour sur plusieurs itérations (par exemple, 100 itérations)
iterations = 100
for t in range(iterations):
    for p in range(m):
        shards[p] = update_shard(shards[p], S, eta, tau, gamma)
    # Ici, on pourrait synchroniser les mises à jour inter-shards si nécessaire (non implémenté)

# À ce stade, chaque shard contient sa partie mise à jour de la matrice ω.

```

Ce code illustre l'approche de **sharding** où chaque shard gère un sous-ensemble des lignes de la matrice ω . Bien que cet exemple ne couvre pas la synchronisation des liaisons inter-shards, il démontre la manière dont la mise à jour locale est effectuée sur chaque fragment, réduisant ainsi la charge de calcul par rapport à une matrice centralisée. Dans une application réelle, des mécanismes supplémentaires seraient intégrés pour synchroniser les valeurs des liens inter-blocs et pour assurer une cohérence globale.

En synthèse, l'équilibrage de charge par sharding permet de gérer efficacement des SCN à grande échelle en divisant la matrice ω en blocs plus petits, améliorant ainsi la scalabilité, la localité des accès et la résilience du système tout en maintenant la dynamique mathématique fondamentale du réseau.

5.8. Sécurité, Fiabilité et Vérifications

5.8.1. Vulnérabilités au Bruit ou à l'Attaque

- 5.8.1.1. Si un agent malveillant manipule $\omega_{i,j}$, risque de clusters artificiels ou sabotage de la dynamique.
- 5.8.1.2. Solutions : logs, watchers, checks d'anomalies.

5.8.2. Robustesse Face aux Pannes

- 5.8.2.1. Quid si un module SCN tombe ? le réseau peut-il se reconfigurer ?
- 5.8.2.2. Basculement automatique ou redondance des données ω .

5.8.3. Contrôle d'Intégrité

- 5.8.3.1. Détection de liens aberrants (trop forts trop vite).
- 5.8.3.2. Calcul d'indicateurs de cohérence globale (ex. distribution statistique des ω).

5.8. Sécurité, Fiabilité et Vérifications

Dans une optique de **grande échelle** et de **distribution** (cf. sections précédentes), la sécurité et la fiabilité d'un **SCN** (Synergistic Connection Network) ne sauraient être laissées de côté, en particulier lorsqu'on envisage des scénarios collaboratifs ou ouverts (accès par divers agents, partage de données, etc.). Le chapitre 5.8 vise à souligner les problématiques de **vulnérabilité** face à des attaques ou manipulations (5.8.1), la robustesse en cas de pannes (5.8.2) et la question du contrôle d'intégrité (5.8.3). Même si l'on se concentre principalement sur l'aspect mathématique et théorique, il est crucial de prévoir des mécanismes de sécurité et de vérification pour un SCN qui puisse fonctionner de façon fiable dans des environnements potentiellement hostiles ou imprévisibles.

5.8.1. Vulnérabilités au Bruit ou à l'Attaque

Dans un SCN largement ouvert — et plus encore s'il est distribué (5.7) —, les entités ou les liens peuvent être soumis à des perturbations extérieures : bruit injecté par le système lui-même, agents malveillants, données corrompues. La sous-section (5.8.1.1) évoque les risques d'attaques et de sabotage, puis (5.8.1.2) propose quelques contre-mesures (logs, watchers, checks d'anomalies, etc.).

5.8.1.1. Si un agent malveillant manipule $\omega_{i,j}$, risque de clusters artificiels ou sabotage de la dynamique

Dans le cadre d'un **Synergistic Connection Network** (SCN), la robustesse de la formation des *clusters* repose sur la mise à jour itérative et locale des pondérations $\omega_{i,j}$ par l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage** et τ le **coefficent de décroissance**. Ce mécanisme vise à renforcer les liaisons correspondant à une **synergie** élevée entre entités tout en assurant une régulation préventive d'une croissance non contrôlée. Cependant, si un **agent malveillant** parvient à intervenir sur $\omega_{i,j}$ ou sur le calcul de $S(i,j)$, il est possible d'injecter des perturbations qui peuvent induire la formation de *clusters artificiels* ou, inversement, empêcher l'émergence des structures attendues, compromettant ainsi la **stabilité** et la **légitimité** du SCN.

A. Enjeu Mathématique et Opérationnel

La dynamique de mise à jour, symbolisée par la fonction F définie par

$$F(\omega_{i,j}(t), S(i,j)) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

est au cœur de la formation des clusters. La **convergence** vers un état stable se traduit par l'atteinte d'un équilibre approximatif $\omega_{i,j}^*$ tel que

$$\omega_{i,j}^* \approx \frac{S(i,j)}{\tau}.$$

L'**intégrité** de cette dynamique dépend de la fiabilité des valeurs de $S(i,j)$ et de $\omega_{i,j}$. Un attaquant qui intervient sur ces paramètres peut, par exemple, **injecter** des valeurs anormalement élevées dans certains éléments de la matrice ω ou altérer le module de calcul de $S(i,j)$ afin de renvoyer des valeurs décalées. En conséquence, l'agent malveillant peut fausser la perception de la *cohérence* entre entités, menant à la formation de **clusters fictifs** – où certaines entités paraissent artificiellement surconnectées – ou, au contraire, paralyser la capacité d'auto-organisation du réseau en annulant les liens essentiels.

Sur le plan **opérationnel**, de telles manipulations peuvent compromettre des systèmes critiques, tels que des algorithmes de recommandation, la coordination de robots ou encore des systèmes de détection d'anomalies, dans

lesquels la configuration de $\omega_{i,j}$ reflète la structure réelle et utile des données. Une altération des pondérations conduit ainsi à une perte de **fiabilité** et de **prédictibilité** dans l'évolution du réseau.

B. Scénarios d'Attaques Possibles

Plusieurs vecteurs d'attaque peuvent être envisagés pour manipuler les valeurs de $\omega_{i,j}$ ou de $S(i,j)$:

Dans ce scénario, un **agent interne** ou une entité ayant acquis un accès privilégié au système modifie directement la matrice des pondérations. Les actions possibles incluent :

- **Gonflement artificiel** : en assignant par exemple $\omega_{i,j} = 100$ pour certaines paires (i,j) , l'attaquant force la dynamique à renforcer de manière excessive des liaisons qui ne reflètent pas la réalité de la synergie entre entités. Ce gonflement crée des *clusters* artificiels, induisant une surconnexion entre des entités qui, en conditions normales, ne devraient pas être regroupées.
- **Affaiblissement ciblé** : en réduisant à zéro ou en diminuant fortement des liens cruciaux pour la cohésion du réseau, l'attaquant peut empêcher la formation de clusters réellement cohésifs, sabotant ainsi le processus d'auto-organisation.

Une autre approche consiste à intercepter ou à altérer le module de calcul de la **synergie**. Par exemple, si le composant $S(i,j)$ est implémenté dans un module nommé *synergyCalculator*, une corruption de ce module peut renvoyer des valeurs exagérément élevées (par exemple $S(i,j) = 10^5$) ou des valeurs très faibles, indépendamment de la relation réelle entre \mathcal{E}_i et \mathcal{E}_j . Cette falsification trompe la mise à jour locale de $\omega_{i,j}$ et conduit le réseau à renforcer des connexions basées sur des indicateurs erronés.

Dans des architectures distribuées, où plusieurs sous-SCN communiquent pour mettre à jour la matrice globale, l'attaquant peut interférer avec la transmission des messages. Les stratégies comprennent :

- **Spoofing des messages** : l'envoi de fausses informations sur les valeurs de $\omega_{i,j}$ ou la synergie calculée, induisant une mise à jour erronée dans les différents nœuds du réseau.
- **Retard ou suppression** : en retardant la transmission ou en supprimant certains messages inter-blocs, l'attaquant crée des incohérences temporelles qui perturbent la convergence du système.

C. Impact sur la Dynamique

Lorsque des valeurs de $\omega_{i,j}$ sont artificiellement gonflées, la dynamique du SCN tend à regrouper les entités associées dans un même cluster, même si elles n'ont pas de synergie naturelle élevée. Le mécanisme de mise à jour renforce ces liaisons aberrantes, menant à l'émergence de *clusters fictifs* qui ne reflètent pas la structure intrinsèque des données. Par exemple, dans un système de recommandation, un groupe d'utilisateurs pourrait être rassemblé artificiellement, faussant ainsi les résultats et la pertinence des recommandations.

Inversement, la réduction ou l'annulation de certains liens cruciaux par l'attaquant empêche la consolidation de clusters légitimes. Le résultat peut être une oscillation continue des valeurs de $\omega_{i,j}$ ou une dispersion générale qui rend impossible l'atteinte d'un état d'équilibre. Dans un contexte de robotique ou de systèmes d'intelligence collective, une telle perturbation conduit à un manque de coordination et à une incapacité d'atteindre une **stabilité opérationnelle**.

Les preuves mathématiques assurant la convergence de la dynamique du SCN reposent sur l'hypothèse d'un comportement aléatoire (et non adversarial) dans la perturbation des pondérations. En présence d'une **perturbation dirigée**, la fonction de mise à jour

$$\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j))$$

se voit remplacée par

$$\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j)) + \text{Perturbation}_{\text{adv}}(i,j,t),$$

où $\text{Perturbation}_{\text{adv}}(i, j, t)$ est choisie par l'attaquant pour maximiser son effet destructeur. Ainsi, les garanties de stabilité, de convergence et de formation de clusters sont compromises, rendant l'analyse théorique du SCN caduque en conditions adversariales.

D. Dimension Mathématique : Perturbation Adversariale de F

D'un point de vue mathématique, l'attaque peut être formalisée par l'introduction d'un **terme perturbateur** dans l'opérateur de mise à jour. Ainsi, au lieu d'avoir

$$\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i, j)),$$

on considère que

$$\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i, j)) + \Delta_{\text{adv}}(i, j, t),$$

où

$$\Delta_{\text{adv}}(i, j, t) = \text{Perturbation}_{\text{adv}}(i, j, t)$$

représente l'action malveillante. Si cette perturbation est suffisamment grande et choisie de manière stratégique, elle peut modifier la trajectoire dynamique de $\omega_{i,j}(t)$ au point de rendre la convergence vers $\omega_{i,j}^* \approx \frac{S(i, j)}{\tau}$ impossible. En d'autres termes, la présence de Δ_{adv} introduit une **incertitude** et une **instabilité** qui empêchent la formation d'un état stable, rendant ainsi le SCN vulnérable à des modifications arbitraires de sa topologie.

Conclusion

En résumé, un **agent malveillant** qui manipule les pondérations $\omega_{i,j}$ ou falsifie les valeurs de synergie $S(i, j)$ peut gravement compromettre la dynamique d'un SCN. Une injection artificielle de valeurs élevées entraîne la création de *clusters artificiels* dans lesquels les entités se retrouvent surconnectées sans réelle justification, tandis que l'annulation ou l'affaiblissement de liens essentiels sabote la formation de clusters cohésifs, conduisant à une instabilité générale du réseau. Mathématiquement, cela se traduit par l'introduction d'un terme perturbateur dans l'opérateur de mise à jour, annulant ainsi les garanties théoriques de convergence et de stabilité sur lesquelles repose le modèle.

Sur le plan opérationnel, de telles attaques représentent une menace majeure pour les systèmes critiques reposant sur le SCN (voir également les sections **5.7.1** et **5.7.2**), car elles compromettent la **fiabilité** et l'**intégrité** du processus d'auto-organisation. Les contre-mesures, qui seront détaillées dans les sections ultérieures (**5.8.1.2**, **5.8.2**, etc.), incluent l'implémentation de protocoles cryptographiques, la vérification de la cohérence des logs et des mécanismes de redondance pour limiter l'impact d'une perturbation adversariale.

En définitive, la robustesse du SCN face à des attaques ciblées sur $\omega_{i,j}$ constitue un enjeu crucial tant du point de vue théorique que pour la mise en œuvre pratique de systèmes d'**intelligence collective** et d'**auto-organisation**.

5.8.1.2. Solutions : Logs, Watchers, Checks d'Anomalies

La robustesse d'un **Synergistic Connection Network** (SCN) face aux attaques malveillantes, telles que celles décrites en **Section 5.8.1.1**, repose sur la mise en place de mécanismes complémentaires destinés à surveiller, tracer et valider en temps réel la dynamique de mise à jour des pondérations $\omega_{i,j}$. Bien que ces solutions ne modifient pas le **coeur mathématique** du modèle – défini par l'équation de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

– elles jouent un rôle essentiel pour **fiabiliser** l'auto-organisation en introduisant une surcouche de sécurité et de contrôle. Nous détaillons ci-après trois axes complémentaires de défense : la **traçabilité** via les journaux (logs), la surveillance en temps réel par des **watchers** et l'utilisation d'algorithmes de **détection d'anomalies**.

A. Journaux (Logs) et Traçabilité

La **tracabilité** est une condition indispensable pour détecter toute modification suspecte dans la dynamique du SCN. En enregistrant chaque événement de mise à jour des pondérations, il est possible de reconstituer l'historique complet de l'évolution de $\omega_{i,j}$. Pour ce faire, chaque mise à jour est loggée avec des métadonnées détaillées, telles que :

- **Identifiants** des entités concernées, (i, j) ,
- La **variation** appliquée, notée

$$\Delta\omega_{i,j}(t) = \omega_{i,j}(t+1) - \omega_{i,j}(t),$$

- Le **timestamp** de la mise à jour,
- L'**ID de l'agent** ou du composant initiateur de la mise à jour,
- Des **paramètres contextuels** (valeur calculée de la synergie $S(i, j)$, paramètres η et τ , etc.),
- Une **signature cryptographique** ou un *hash* du message, par exemple

$$H(i, j, t) = \text{Hash}(i, j, \omega_{i,j}(t), \Delta\omega_{i,j}(t), \text{timestamp}),$$

Ces **logs** constituent un registre immuable qui peut être exploité postérieurement pour une **analyse forensique**. Leur utilité est double : ils permettent d'identifier en temps différé des anomalies (par exemple, une variation exceptionnelle telle que $\Delta\omega_{i,j}(t)$ passant de 0.2 à 100 en une seule itération) et offrent une base pour **corrélér** les événements avec les actions d'un potentiel attaquant. Dans un contexte distribué, la centralisation ou la synchronisation des logs via des solutions de stockage sécurisées (bases NoSQL, systèmes de fichiers distribués avec intégrité garantie) assure une **tracabilité** à l'échelle du réseau.

B. Watchers et Monitoring en Temps Réel

Les **watchers** sont des modules de surveillance intégrés dans la boucle de mise à jour des pondérations. Leur rôle est de **filtrer** en temps réel les variations de $\omega_{i,j}$ afin de détecter immédiatement toute anomalie qui pourrait résulter d'une action malveillante. Mathématiquement, on peut considérer un Watcher comme une fonction de contrôle W appliquée à chaque mise à jour :

$$W(\Delta\omega_{i,j}(t)) = \begin{cases} 1, & \text{si } |\Delta\omega_{i,j}(t)| > \kappa, \\ 0, & \text{sinon,} \end{cases}$$

où κ est un **seuil critique** défini en fonction de la dynamique attendue du SCN. Lorsqu'une condition d'alerte est satisfaite – par exemple, si

$$|\omega_{i,j}(t+1) - \omega_{i,j}(t)| > \kappa \quad \text{ou} \quad \omega_{i,j}(t+1) > \omega_{\max},$$

– le Watcher déclenche une **alerte en temps réel**. Cette alerte peut conduire à diverses actions immédiates :

- **Blocage temporaire** de la mise à jour pour le lien concerné,
- **Demande de validation manuelle** ou automatisée via des protocoles d'authentification renforcée,
- **Transmission** d'un signal de sécurité à un module central de surveillance ou à d'autres nœuds du réseau pour une **réaction collective**.

En contexte distribué, plusieurs Watchers peuvent coopérer et partager leurs observations via un protocole de consensus, assurant ainsi une **vérification mutuelle** des mises à jour anormales avant qu'elles ne perturbent la dynamique globale.

C. Checks d'Anomalies et Méthodes Avancées

Au-delà des contrôles immédiats et locaux effectués par les Watchers, il est possible d'implémenter des algorithmes d'**anomaly detection** qui s'intéressent à la **distribution** globale des pondérations et à l'évolution des **clusters**. Ces méthodes avancées incluent :

Une approche consiste à surveiller des métriques statistiques sur l'ensemble des pondérations. Par exemple, en calculant la **variance** σ^2 et l'**entropie** H de la distribution des ω_{ij} :

$$\sigma^2 = \frac{1}{N} \sum_{i < j} (\omega_{i,j} - \bar{\omega})^2, \quad H = - \sum_{i < j} p(\omega_{i,j}) \ln p(\omega_{i,j}),$$

où $\bar{\omega}$ est la moyenne des $\omega_{i,j}$ et $p(\omega_{i,j})$ la distribution empirique des poids. Une variation brutale de σ^2 ou une chute/incrément inattendu de H par rapport aux valeurs historiques peut être le signe d'une **injection malveillante**.

En complément de l'analyse statistique, l'observation de la **topologie** des clusters peut révéler des configurations anormales. Par exemple, l'apparition soudaine d'un cluster isolé dont la cohésion interne serait extrêmement élevée, ou au contraire, la disparition d'un cluster connu, peut être détectée par des techniques de **clustering supervisé ou non supervisé**. On peut alors définir un indice de **cohérence cluster** C tel que :

$$C = \frac{1}{|G|} \sum_{(i,j) \in G} \omega_{i,j},$$

où G désigne l'ensemble des paires au sein d'un cluster donné. Une valeur de C anormalement haute ou basse par rapport aux attentes théoriques indique un dysfonctionnement, pouvant être imputé à une attaque.

Lorsque des anomalies sont détectées, il est impératif de disposer de **mécanismes de réaction** qui permettent de limiter l'impact d'une attaque. Parmi ceux-ci, on peut citer :

- Le **rollback**, consistant à rétablir les valeurs antérieures de $\omega_{i,j}$ issues des logs jugées fiables,
- La mise en quarantaine des noeuds ou des liens suspectés d'avoir été altérés,
- La déclinaison de procédures de **vérification renforcée** (p.ex. via des signatures numériques ou des protocoles de consensus) pour valider les mises à jour futures.

Ces politiques de réaction reposent sur une modélisation mathématique du problème sous la forme d'un opérateur de perturbation, où l'on cherche à minimiser l'écart entre la trajectoire observée et la trajectoire attendue définie par $F(\omega(t))$.

Conclusion

Les mesures de **sécurité** déployées autour du SCN – incluant la **consignation détaillée des logs**, la **surveillance en temps réel** via des Watchers et l'implémentation d'algorithmes de **détection d'anomalies** – constituent une couche de défense essentielle pour garantir la **fiabilité** et la **stabilité** de la dynamique d'auto-organisation. Du point de vue **mathématique**, ces solutions n'altèrent pas la fonction de mise à jour fondamentale $\omega(t+1) = F(\omega(t))$ mais introduisent un mécanisme de contrôle externe qui valide ou rejette les évolutions des pondérations en fonction de critères prédéfinis. Du point de vue **opérationnel**, elles permettent de détecter et d'intervenir rapidement en cas de comportement aberrant, réduisant ainsi le risque de formation de clusters artificiels ou de sabotage de la convergence du SCN. Ces solutions, intégrées de manière cohérente dans une architecture distribuée (cf. Sections 5.7 et 5.8), renforcent la **confiance** dans le système et permettent d'en assurer la résilience face à des attaques potentielles dans des environnements critiques.

5.8.2. Robustesse Face aux Pannes

Même lorsque le **SCN** (Synergistic Connection Network) est doté de mécanismes de sécurité (5.8.1) et d'une architecture soigneusement construite (5.7), il subsiste toujours la possibilité de **pannes** : un sous-SCN peut cesser de fonctionner, un serveur peut tomber en panne, ou un module logiciel critique peut se bloquer. La robustesse d'un système à ces défaillances est alors un point essentiel : comment assurer que le **réseau** global ne s'écroule pas et qu'il puisse, si nécessaire, se **reconfigurer** pour continuer de fonctionner ? Dans cette section (5.8.2), nous examinons d'abord (5.8.2.1) les scénarios de panne d'un module SCN et la possibilité de reconfiguration, puis (5.8.2.2) discute les approches de basculement automatique et de redondance.

5.8.2.1. Quid si un module SCN tombe ? Le réseau peut-il se reconfigurer ?

Dans un **Synergistic Connection Network** (SCN) distribué, la robustesse et la résilience du système dépendent de sa capacité à absorber les perturbations, notamment lorsque l'un des modules ou sous-réseaux, noté SCN_p , tombe en panne. Un tel module gère un sous-ensemble d'entités, \mathcal{V}_p , ainsi que les liaisons internes associées, c'est-à-dire l'ensemble $\{\omega_{i,j} \mid i, j \in \mathcal{V}_p\}$. La question se pose alors de savoir si le réseau global peut se reconfigurer pour continuer à fonctionner malgré la perte partielle ou totale d'un module, et quelles stratégies – tant du point de vue opérationnel que mathématique – permettent d'assurer une telle résilience.

A. Scénarios de Panne dans un SCN Distribué

Dans un système décomposé en plusieurs modules $\text{SCN}_1, \dots, \text{SCN}_m$, les pannes peuvent se manifester de diverses manières, affectant différemment la dynamique globale. Nous distinguons principalement trois scénarios :

Tout d'abord, dans le **cas mineur**, la panne d'un module SCN_p est de courte durée. Par exemple, un crash temporaire dû à une défaillance matérielle ou à un incident réseau provoque l'arrêt de la mise à jour des liaisons concernant les entités \mathcal{V}_p pendant une période limitée. Pendant ce temps, le reste du SCN poursuit son évolution selon la dynamique standard

$$\omega(t+1) = F(\omega(t)) \quad \text{où} \quad F(\omega(t)) = \omega(t) + \eta [S - \tau \omega(t)].$$

Les liens impliquant \mathcal{V}_p peuvent être alors mis en veille ou temporairement ignorés. Une fois le module rétabli, une phase de **re-synchronisation** est nécessaire pour réintégrer les entités affectées dans le calcul global, en récupérant les valeurs sauvegardées (logs, snapshots) afin de minimiser la rupture de la dynamique.

Ensuite, dans le **cas majeur**, la panne est définitive. On peut alors envisager deux issues complémentaires : soit les données associées à \mathcal{V}_p sont irrémédiablement perdues, soit il est nécessaire de migrer ces entités vers un autre module SCN_q . Dans ce dernier cas, la réaffectation implique une **réorganisation de la partition** initiale $\{\mathcal{V}_1, \dots, \mathcal{V}_m\}$ de l'ensemble des entités. Concrètement, les lignes et colonnes de la matrice ω correspondant aux indices de \mathcal{V}_p sont soit supprimées, soit transférées dans un nouvel espace de calcul, ce qui se traduit par une modification structurelle de la dynamique.

Enfin, un **cas hybride** peut survenir lorsqu'une panne initialement mineure se prolonge, rendant la réintégration problématique. Dans ce scénario, la solution de reconfiguration doit prendre en compte la durée d'inactivité et les conséquences sur la **topologie** du réseau, en adoptant par exemple une approche hybride qui combine le gel temporaire des liaisons avec une éventuelle réallocation progressive des entités affectées.

B. Reconfiguration ou Dégradation du SCN

La résilience du réseau repose sur sa capacité à se reconfigurer en réponse à la perte d'un module. Deux approches principales peuvent être envisagées :

Dans le cas d'une panne définitive ou prolongée, la stratégie la plus souhaitable est la **migration** des entités \mathcal{V}_p vers un autre module SCN_q . Cette procédure de réaffectation s'effectue en plusieurs étapes :

- **Sauvegarde et restauration** : Si des mécanismes de backup ou des logs détaillés existent, ils permettent de reconstruire l'historique des mises à jour des liaisons $\omega_{i,j}$ pour $i,j \in \mathcal{V}_p$. On dispose alors d'une base de données pour réintégrer ces valeurs dans le nouveau module.
- **Réaffectation de la partition** : La partition initiale $\{\mathcal{V}_1, \dots, \mathcal{V}_m\}$ est redéfinie en intégrant les entités de \mathcal{V}_p dans un autre sous-ensemble, par exemple en formant un nouvel ensemble $\mathcal{V}_q' = \mathcal{V}_q \cup \mathcal{V}_p$. La mise à jour de la dynamique se fait alors sur une matrice ω' de dimension réduite ou réallouée, avec

$$\omega': \mathcal{V}_{\text{nouvelle}} \times \mathcal{V}_{\text{nouvelle}} \rightarrow \mathbb{R}^+.$$

- **Re-synchronisation inter-blocs** : Une fois les entités migrées, il est crucial que les liaisons inter-blocs soient actualisées afin de restaurer la cohérence globale. Ceci peut être réalisé par un protocole de synchronisation qui met à jour les valeurs de $S(i,j)$ pour les nouveaux liens entre \mathcal{V}_q' et les autres sous-ensembles.

Si la panne est de courte durée, il est souvent préférable d'adopter une **stratégie de dégradation** temporaire. Dans ce cas, les entités appartenant à \mathcal{V}_p sont mises hors circuit, et le SCN global continue à fonctionner en excluant les mises à jour associées à ce bloc. Mathématiquement, cela revient à définir une fonction indicatrice I_p telle que :

$$I_p(i) = \begin{cases} 0, & \text{si } i \in \mathcal{V}_p, \\ 1, & \text{sinon.} \end{cases}$$

La dynamique de mise à jour est alors adaptée de la forme

$$\omega_{i,j}(t+1) = I_p(i)I_p(j) \left[\omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] \right],$$

ce qui garantit que les liens impliquant des entités hors ligne ne perturbent pas la convergence du réseau. Lorsque le module défaillant revient en ligne, un protocole de **reconnexion** permet de réintégrer progressivement les entités concernées dans la dynamique globale.

C. Analyse Mathématique de la Panne

D'un point de vue formel, la panne d'un module SCN se traduit par la suppression des lignes et colonnes correspondant aux entités \mathcal{V}_p dans la matrice des pondérations ω . Soit $\omega(t)$ une matrice $N \times N$ décrivant la dynamique initiale, et soit $P \subset \{1, \dots, N\}$ l'ensemble des indices associés à \mathcal{V}_p . La panne entraîne la considération d'une nouvelle matrice réduite

$$\omega_{\text{réduit}}(t) = (\omega_{i,j}(t))_{i,j \notin P}.$$

La dynamique de mise à jour s'exprime alors par

$$\omega_{\text{réduit}}(t+1) = \omega_{\text{réduit}}(t) + \eta [S_{\text{réduit}} - \tau \omega_{\text{réduit}}(t)],$$

où $S_{\text{réduit}}$ désigne la restriction de la fonction de synergie aux entités encore actives. Plusieurs questions se posent à ce stade :

- **Robustesse de la convergence** : Les théorèmes de convergence établissant que $\omega(t)$ tend vers un équilibre de la forme $\omega^* \approx S/\tau$ reposent sur la structure complète du réseau. La suppression d'un sous-ensemble de nœuds représente une perturbation du système dynamique qui doit être analysée à l'aide des théories de la robustesse des systèmes dynamiques et de la résilience des graphes.
- **Impact sur la connectivité** : La disparition d'un bloc peut modifier la **connectivité** globale du SCN, affectant ainsi la capacité du réseau à former des clusters cohérents. Des résultats de théorie des graphes indiquent que la suppression d'un sous-ensemble de nœuds (ou de liens) peut, dans le pire des cas, décomposer le graphe en plusieurs composantes, mais si le réseau initial possède une **redondance** suffisante, la connectivité résiduelle permettra au système de continuer à converger de manière partielle.
- **Temps de re-synchronisation** : La migration des entités vers un autre module ou le retour d'un module en panne induit un délai pendant lequel la matrice ω évolue sous une topologie modifiée. Il est alors nécessaire

d'étudier le temps de convergence de la dynamique modifiée, qui peut être exprimé en fonction des paramètres η et τ , ainsi que de la proportion d'entités affectées.

Ces considérations mathématiques traduisent le fait que, bien que la disparition d'un module perturbe la dynamique initiale, si la structure du réseau est suffisamment redondante et si des mécanismes de reconfiguration sont en place, le système peut retrouver un nouvel état d'équilibre qui, bien que différent de celui initial, reste fonctionnel.

Conclusion

Lorsqu'un module SCN, SCN_p , tombe en panne dans un environnement distribué, le système global se trouve confronté à une **perturbation structurelle** importante. Trois scénarios principaux se dégagent : la dégradation temporaire du réseau (où les entités de \mathcal{V}_p sont mises hors ligne), la reconfiguration par migration des entités vers d'autres modules, ou la perte définitive des entités affectées. D'un point de vue mathématique, cela correspond à une réduction de dimension dans la matrice ω et à une réinitialisation partielle de la dynamique $\omega(t+1) = F(\omega(t))$.

Les stratégies de réallocation – telles que la migration vers un autre bloc, la réorganisation de la partition ou le maintien d'un mode dégradé – permettent de préserver, dans la mesure du possible, la capacité du réseau à converger et à former des clusters cohérents malgré la disparition d'un sous-ensemble d'entités. La clé réside dans la redondance et dans la rapidité d'intervention pour rétablir la connectivité inter-blocs. Ainsi, le SCN, conçu pour être résilient, peut se reconfigurer et continuer à évoluer, assurant ainsi la continuité des services même en présence de défaillances matérielles ou logicielles.

En définitive, la gestion des pannes dans un SCN distribué repose sur un équilibre délicat entre la **reconfiguration dynamique** des sous-ensembles d'entités et la **préservation** des propriétés de convergence et de stabilité du système global. Les mécanismes de sauvegarde, de migration et de synchronisation constituent autant d'outils indispensables pour garantir la résilience d'un réseau de grande envergure dans des environnements potentiellement hostiles.

5.8.2.2. Basculement Automatique ou Redondance des Données ω

Dans un **Synergistic Connection Network** (SCN) distribué, la dynamique des pondérations $\omega(t)$ – régie par la mise à jour itérative

$$\omega(t+1) = F(\omega(t)) = \omega(t) + \eta [S - \tau \omega(t)]$$

– doit continuer à évoluer de manière cohérente même en cas de défaillance d'un sous-réseau ou d'un microservice. Pour ce faire, il est essentiel d'implémenter des mécanismes de **basculement automatique** (failover) et de **redondance** (réplication) des données ω . Ces dispositifs, couramment utilisés en ingénierie des systèmes distribués, permettent d'assurer la continuité de l'auto-organisation et la résilience du SCN, en minimisant l'impact d'une panne prolongée ou définitive d'un module.

A. Basculement Automatique (Failover)

Le basculement automatique, ou failover, se définit comme le processus par lequel la responsabilité d'un module défaillant est transférée à un nœud opérationnel afin de garantir la disponibilité du service. Dans le contexte d'un SCN composé de plusieurs sous-réseaux SCN_1, \dots, SCN_m , chaque sous-ensemble d'entités \mathcal{V}_p est géré par son bloc dédié. Lorsque le module SCN_p devient injoignable en raison d'une panne (causée par exemple par un crash matériel, un problème de réseau ou une défaillance logicielle), un orchestrateur de surveillance, qui utilise des signaux de type « heartbeat », détecte l'absence de réponse et déclenche un mécanisme de failover.

Ce mécanisme repose sur la re-partition de l'ensemble des entités initialement réparties selon la collection $\{\mathcal{V}_1, \dots, \mathcal{V}_m\}$. Dès lors, les entités de \mathcal{V}_p sont transférées vers un autre bloc opérationnel, par exemple SCN_q . Mathématiquement, cela revient à redéfinir la partition du système en posant, pour un basculement réussi,

$$\mathcal{V}'_q = \mathcal{V}_q \cup \mathcal{V}_p,$$

et à mettre à jour la dynamique locale en appliquant la fonction F sur la nouvelle matrice de pondérations $\omega'(t)$ qui est reconstruite à partir de la réunion des données des blocs concernés. En pratique, cette opération nécessite que les informations relatives aux pondérations $\omega_{i,j}$ pour $i,j \in \mathcal{V}_p$ soient immédiatement accessibles via un système de stockage redondant. La transparence de ce mécanisme repose sur la rapidité avec laquelle l'orchestrateur détecte la panne et redirige les flux de données, de manière à ce que la continuité de la dynamique globale ne soit que temporairement perturbée.

B. Redondance (RéPLICATION) des Données ω

La redondance des données constitue le second pilier de la résilience d'un SCN. Sans réPLICATION, la panne d'un module entraînerait la perte définitive des pondérations associées aux entités de ce module, rendant impossible toute opération de failover. La réPLICATION consiste à stocker simultanément plusieurs copies des données critiques, en l'occurrence les valeurs de ω , sur des nœuds ou dans des clusters de stockage distribués.

Plusieurs stratégies de réPLICATION peuvent être mises en œuvre :

D'abord, la stratégie **N-Way** consiste à répliquer chaque portion de la matrice ω sur N nœuds distincts. Formellement, pour chaque lien $\omega_{i,j}$ lié aux entités d'un sous-ensemble \mathcal{V}_p , on maintient N copies, notées $\omega_{i,j}^{(1)}, \omega_{i,j}^{(2)}, \dots, \omega_{i,j}^{(N)}$, telles que la mise à jour de l'une ou plusieurs d'entre elles garantit la disponibilité de l'information en cas de défaillance d'un nœud.

Une autre approche courante est le modèle **Master-Slave**, dans lequel un nœud maître détient la version actuelle et définitive de $\omega_{i,j}$ pour un sous-ensemble donné, et plusieurs nœuds esclaves reçoivent les mises à jour en quasi temps réel. En cas de défaillance du maître, l'un des esclaves peut prendre le relais, en assurant la continuité du service. On peut représenter ce mécanisme par l'équation de synchronisation suivante : si $\omega_{i,j}^{(M)}(t)$ désigne la version maître et $\omega_{i,j}^{(S)}(t)$ la version esclave, alors

$$\omega_{i,j}^{(S)}(t) = \omega_{i,j}^{(M)}(t) + \delta_{i,j}(t),$$

où $\delta_{i,j}(t)$ est un terme de décalage que l'on s'efforce de maintenir très faible grâce à des protocoles de consensus (par exemple, via le protocole Paxos ou Raft).

Enfin, des méthodes telles que **Erasure Coding** permettent d'optimiser l'espace de stockage tout en garantissant la possibilité de reconstruire la matrice ω en cas de perte de certaines portions de données. Dans ce cadre, ω est codée en plusieurs fragments, dont un nombre suffisant permet de reconstituer l'information d'origine, suivant un schéma de correction d'erreurs.

C. Conséquences sur la Continuité de la Dynamique

L'intégration de mécanismes de basculement automatique et de redondance a pour objectif fondamental de permettre au SCN de poursuivre sa dynamique d'auto-organisation malgré la perte ou l'inaccessibilité temporaire d'un module. Lorsqu'un basculement est déclenché, la reconfiguration du réseau se traduit par une mise à jour de la partition des entités et par la restauration des pondérations via les copies redondantes, de sorte que la dynamique

$$\omega(t+1) = F(\omega(t))$$

peut reprendre son évolution sans perte significative de la structure initialement acquise.

Le basculement automatique assure que, dès qu'un module défaillant est détecté, un nouveau nœud héberge les entités concernées et continue la mise à jour. La redondance des données ω garantit quant à elle que l'historique des liaisons, et par extension la « mémoire » du réseau, est préservée. Même si une légère asynchronie peut être introduite – en raison de délais dans la synchronisation des copies – des protocoles d'agrégation et de réconciliation (par exemple, l'utilisation de vecteurs d'horodatage ou de mécanismes de fusion d'updates) permettent de minimiser ces écarts et d'assurer la cohérence de la dynamique.

L'architecture résiliente ainsi obtenue permet au SCN de s'adapter à des environnements imprévisibles en maintenant une **convergence partielle** ou, idéalement, une **reconvergence** vers un nouvel équilibre qui préserve les propriétés essentielles de l'auto-organisation, notamment la formation de clusters et la stabilité de la dynamique.

Conclusion

Le basculement automatique et la redondance des données ω constituent des mécanismes indispensables pour assurer la résilience d'un SCN distribué. Grâce au basculement (failover), le système peut rediriger instantanément les entités d'un module défaillant vers un autre bloc opérationnel, modifiant ainsi dynamiquement la partition des entités tout en préservant la continuité des mises à jour. Parallèlement, la réPLICATION ou la redondance des pondérations garantit que, même en cas de panne, l'information essentielle sur la dynamique des liaisons est conservée et peut être rapidement restaurée. Du point de vue mathématique, ces mécanismes n'altèrent pas la forme fondamentale de la mise à jour

$$\omega(t+1) = F(\omega(t)),$$

mais assurent que cette dynamique puisse se poursuivre de manière fiable malgré des défaillances ponctuelles ou prolongées. En somme, ces stratégies de basculement et de redondance sont cruciales pour déployer un SCN de grande échelle, capable de résister aux aléas d'une infrastructure distribuée, tout en maintenant l'intégrité et la continuité de son processus d'auto-organisation.

5.8.3. Contrôle d'Intégrité

Alors même que l'on déploie des mécanismes de sécurité (5.8.1) et assure la robustesse face aux pannes (5.8.2), il reste un volet essentiel : **vérifier l'intégrité** de la dynamique ω en continu, de manière à détecter à la fois des anomalies locales (liens "invraisemblables") et d'éventuelles incohérences globales (perte de cohérence statistique du réseau). Dans un **SCN** à grande échelle ou évolutif, un tel **contrôle d'intégrité** garantit que la formation des clusters ou la stabilisation des pondérations n'est pas subrepticement dévoyée par des artefacts (erreurs, manipulations, accidents). La section 5.8.3 détaille d'abord (5.8.3.1) la détection de liens aberrants, puis (5.8.3.2) évoque des indicateurs de cohérence globale.

5.8.3.1. Détection de Liens Aberrants (Trop Forts Trop Vite)

Dans le cadre d'un **Synergistic Connection Network** (SCN), la stabilité de la dynamique d'auto-organisation repose sur des mises à jour progressives et contrôlées des pondérations $\omega_{i,j}$. En effet, la règle de mise à jour classique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

est conçue pour induire une évolution graduelle, dans laquelle les valeurs des liens se renforcent ou décroissent de façon cohérente avec les paramètres du système, notamment le taux η et le coefficient de décroissance τ . Cependant, il peut survenir des situations où un lien évolue de manière anormale, c'est-à-dire où $\omega_{i,j}$ « explose » ou augmente de façon disproportionnée d'une itération à l'autre. Ce phénomène, désigné par l'expression « trop forts trop vite », constitue un indice de comportement aberrant, qu'il soit provoqué par des erreurs de calcul, des dysfonctionnements logiciels ou des attaques malveillantes. La détection de ces liens anormaux est essentielle afin de préserver la cohérence du réseau et de déclencher, le cas échéant, des mesures correctives.

A. Motivation et Contexte

Le principe fondamental de l'auto-organisation dans un SCN repose sur la mise à jour itérative des pondérations. En temps normal, cette mise à jour induit une progression contrôlée, telle que décrite par l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où la valeur de $S(i,j)$ – la synergie entre les entités i et j – détermine l'orientation du renforcement ou de l'affaiblissement du lien, et où le facteur η module l'amplitude des modifications. Dans un système bien calibré, on

s'attend à ce que les variations $\Delta\omega_{i,j} = |\omega_{i,j}(t+1) - \omega_{i,j}(t)|$ soient relativement modestes, se conformant à une distribution normale ou à un intervalle prédéfini de variations.

Pourtant, lorsqu'un lien subit une croissance brutale – par exemple, une augmentation de la valeur par un facteur de 100 en une seule itération – il devient impératif de qualifier ce comportement d'**aberrant**. Ce type d'évolution peut être le résultat d'une erreur logicielle (panne dans le module de calcul de la synergie ou bug dans l'implémentation de la règle de mise à jour), d'un dysfonctionnement numérique (problèmes d'overflow ou d'arrondis en environnement GPU/HPC), ou encore d'une manipulation malveillante (voir section 5.8.1 pour le contexte d'attaques).

B. Définition et Caractéristiques d'un Lien Aberrant

Dans un SCN mature, la majorité des pondérations $\omega_{i,j}$ se situent dans un intervalle relativement restreint, par exemple entre 0 et 1, voire un peu au-dessus en fonction de la normalisation appliquée. Pour quantifier ce qui constitue une valeur « hors norme », on peut définir des indicateurs statistiques tels que la moyenne μ et l'écart-type σ de la distribution des ω . Un lien est considéré comme aberrant s'il se trouve en dehors de l'intervalle

$$\mu \pm k \sigma,$$

où k est un paramètre choisi en fonction du degré de tolérance souhaité (par exemple, $k = 3$ pour une détection de valeurs extrêmes dans une distribution gaussienne). De même, la définition d'un seuil statique ω_{limit} – par exemple, une valeur de 2.0 ou 10.0 – permet d'identifier rapidement des liens dont la valeur excède largement la plage normale.

Au-delà de la valeur absolue d'un lien, l'analyse de la vitesse d'évolution, c'est-à-dire du saut

$$\Delta\omega_{i,j} = |\omega_{i,j}(t+1) - \omega_{i,j}(t)|,$$

est tout aussi cruciale. Dans un contexte de mise à jour graduelle, le paramètre η limite généralement l'amplitude des incrément. Ainsi, si dans des conditions standards on observe des variations d'ordre $O(10^{-2})$ ou $O(10^{-1})$, une variation de $O(1)$ ou supérieure en une seule itération suggère une anomalie. On peut introduire un test simple sous la forme

$$|\omega_{i,j}(t+1) - \omega_{i,j}(t)| > \delta,$$

où δ est un seuil défini en fonction de la dynamique attendue. Le franchissement de ce seuil constitue un signal d'alerte indiquant qu'un lien évolue de manière trop rapide.

C. Mécanismes de Détection

La détection des liens aberrants dans un SCN peut s'appuyer sur plusieurs mécanismes complémentaires qui, ensemble, forment une couche de sécurité supplémentaire dans la boucle de mise à jour des pondérations.

La méthode la plus directe consiste à imposer un seuil statique, noté ω_{limit} , au-delà duquel un lien est automatiquement marqué comme suspect. Parallèlement, on peut effectuer un contrôle sur la variation entre deux itérations. Concrètement, si l'on observe

$$\omega_{i,j}(t+1) > \omega_{\text{limit}} \quad \text{ou} \quad |\omega_{i,j}(t+1) - \omega_{i,j}(t)| > \delta,$$

alors le lien concerné est considéré comme aberrant. Ces seuils peuvent être définis de manière statique pour un SCN donné ou ajustés dynamiquement en fonction de la phase de convergence du réseau.

Une approche plus globale consiste à analyser la distribution des valeurs de ω sur l'ensemble du réseau. En estimant la moyenne μ et l'écart-type σ des pondérations, il est possible d'identifier les valeurs qui se trouvent en dehors d'un intervalle de confiance, par exemple $\mu \pm 3\sigma$. Cette approche permet de repérer non seulement des liens isolés, mais aussi des motifs d'anomalies lorsque plusieurs liens s'écartent simultanément des valeurs attendues. De plus, l'analyse des quantiles (par exemple, considérer que 99 % des liens se trouvent en dessous d'une valeur donnée) peut servir d'indicateur pour fixer dynamiquement un seuil.

Les techniques de **machine learning** appliquées à la détection d'anomalies offrent une méthode avancée pour identifier les comportements aberrants dans le réseau. Ces algorithmes, entraînés sur des données historiques ou simulées du

SCN, permettent de reconnaître des patterns anormaux, tels que des sauts brusques de plusieurs liens simultanément ou l'apparition soudaine d'un cluster entier présentant des valeurs extrêmes. L'intégration de ces méthodes peut se faire en parallèle de la mise à jour des pondérations et fournir des alertes en temps réel sur la base d'un score d'anomalie calculé pour chaque lien.

D. Réactions et Mesures Correctives

La détection d'un lien aberrant n'est que la première étape ; il est essentiel de mettre en place des mécanismes de réaction pour limiter l'impact de tels événements.

Lorsqu'un lien est détecté comme aberrant, un système de surveillance (souvent appelé « Watcher ») peut générer une alerte immédiate, notifiant un administrateur ou un module de contrôle central. Cette alerte peut inclure des informations détaillées telles que l'identifiant du lien, la valeur observée, le saut $\Delta\omega_{i,j}$, et l'instant de l'événement, facilitant ainsi une analyse ultérieure.

Dans des environnements critiques, la détection d'un comportement anormal peut déclencher la suspension temporaire du lien concerné. Le système peut alors appliquer un mécanisme de rollback, annulant la mise à jour problématique et rétablissant la valeur précédente de $\omega_{i,j}$ jusqu'à ce qu'un recalculation soit effectué en toute sécurité. Par exemple, on peut définir la mise à jour corrective suivante :

$$\omega_{i,j}(t+1) = \min\{\omega_{i,j}(t+1), \omega_{\text{limit}}\},$$

ce qui impose un clamp (ou clip) sur la valeur du lien afin d'empêcher toute croissance incontrôlée.

Enfin, en complément des réactions immédiates, le SCN peut être équipé d'un module de réévaluation qui, après détection d'un lien aberrant, effectue une nouvelle estimation de la synergie $S(i,j)$ ou applique une mise à jour plus prudente pour ce lien. Cette stratégie peut inclure la mise en œuvre d'un facteur de pondération adaptatif temporaire ou d'un filtre qui atténue la réponse de mise à jour dans la zone de détection de l'anomalie.

Conclusion

La détection de liens aberrants dans un SCN est une composante cruciale pour assurer la robustesse et la fiabilité de la dynamique d'auto-organisation. En se fondant sur l'observation des valeurs absolues et des variations de $\omega_{i,j}$, des techniques statistiques telles que l'analyse de la distribution (moyenne, écart-type, quantiles) permettent de qualifier un lien comme anormal lorsque sa valeur dépasse un seuil préétabli ou lorsqu'un saut brutal est constaté. Des mécanismes complémentaires, tels que l'implémentation de tests statiques et dynamiques (par exemple, la condition $|\Delta\omega_{i,j}| > \delta$), ainsi que le déploiement d'algorithme d'anomaly detection basés sur le machine learning, offrent une approche globale pour repérer et signaler les anomalies. Sur le plan opérationnel, la mise en place d'un module de surveillance (Watcher) intégré dans la boucle de mise à jour permet de déclencher des alertes, de suspendre ou de rétablir les mises à jour suspectes, et d'appliquer des mesures correctives telles que le rollback ou le clamp. Ces dispositifs, en collaboration avec des systèmes de logging détaillé et des contrôles de cohérence, garantissent que la dynamique du SCN reste fidèle aux principes d'auto-organisation, même face à des variations imprévues ou malveillantes.

285.5.8.3.2. Calcul d'Indicateurs de Cohérence Globale (ex. Distribution Statistique des ω)

Dans le contexte d'un **Synergistic Connection Network** (SCN), la dynamique d'auto-organisation repose sur l'évolution contrôlée des pondérations $\omega_{i,j}$ qui relient les entités. Alors que la détection locale de liens aberrants (voir Section 5.8.3.1) permet d'identifier des sauts brusques dans la mise à jour de $\omega_{i,j}$, il est tout aussi crucial d'assurer qu'à l'échelle globale la matrice ω conserve une structure cohérente et ne dérive pas vers un état artificiellement faussé. La surveillance globale repose sur le calcul et le suivi d'indicateurs statistiques qui quantifient la « bonne santé » de la distribution des pondérations, garantissant ainsi la pertinence des clusters formés par le SCN. Nous développerons ici en détail les motivations, les exemples d'indicateurs, les méthodes de surveillance et les considérations tant mathématiques que pratiques associées à cette démarche.

A. Motivation et Contexte de la Surveillance Globale

L'évolution normale d'un SCN repose sur une mise à jour graduelle de $\omega_{i,j}$ selon la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où les paramètres η et τ régulent respectivement le taux d'apprentissage et la décroissance. Dans un système bien calibré, la majorité des liens évolue dans une plage prédefinie et stable, souvent centrée autour d'une valeur moyenne avec une dispersion modérée. Toutefois, plusieurs risques peuvent compromettre cette cohérence globale :

- **Dérive systémique** : Même en l'absence d'erreurs locales, un mauvais paramétrage (valeurs inappropriées de η ou τ) peut entraîner une dérive collective des valeurs de ω . Par exemple, une croissance exponentielle de la somme totale des pondérations,

$$\Sigma(t) = \sum_{i,j} \omega_{i,j}(t),$$

ou un effondrement généralisé vers zéro, pourraient indiquer un problème de convergence global.

- **Manipulations malveillantes** : Comme évoqué dans la Section 5.8.1, un attaquant peut altérer certaines valeurs, créant des biais qui, même s'ils ne se manifestent pas immédiatement sur un lien isolé, perturbent l'ensemble de la distribution.
- **Effets numériques** : Des erreurs d'arrondi, des overflow ou des problèmes de précision dans des environnements de calcul intensif (GPU, HPC) peuvent provoquer des modifications subtiles mais cumulatives dans la répartition des ω .

Ainsi, il est essentiel de disposer d'indicateurs globaux qui, en agrégant les données sur l'ensemble du SCN, permettent de détecter des dérives ou des anomalies pouvant compromettre la formation et la stabilité des clusters.

B. Exemples d'Indicateurs de Cohérence Globale

Pour évaluer la « santé » globale de la matrice ω , plusieurs indicateurs peuvent être calculés et suivis dans le temps.

Une première approche consiste à analyser la distribution statistique des valeurs de ω en calculant des moments de la distribution. La **moyenne** $\bar{\omega}$ et la **variance** $\text{Var}(\omega)$ sont définies par :

$$\bar{\omega} = \frac{1}{N} \sum_{i,j} \omega_{i,j}, \quad \text{Var}(\omega) = \frac{1}{N} \sum_{i,j} (\omega_{i,j} - \bar{\omega})^2,$$

où N est le nombre total de liaisons. En outre, l'analyse des **quantiles** permet de connaître la répartition des valeurs : par exemple, on peut déterminer le 95^e percentile et vérifier que 95 % des liens se situent en dessous d'une certaine valeur. Des écarts brusques dans ces indicateurs, par exemple une augmentation soudaine de la moyenne ou un élargissement de la variance, peuvent signaler un déséquilibre dans la dynamique globale.

L'**entropie** $H(\omega)$ offre une mesure de la diversité des pondérations et se définit par :

$$H(\omega) = - \sum_b p(b) \log p(b),$$

où $p(b)$ est la probabilité qu'un lien $\omega_{i,j}$ se situe dans le bin b d'un histogramme construit sur les valeurs de ω . Une entropie élevée indique une distribution diversifiée, tandis qu'une entropie faible pourrait suggérer une homogénéisation anormale (par exemple, si presque tous les ω convergent vers une valeur unique). Un changement soudain de l'entropie au fil du temps peut donc être un indicateur puissant d'une dérive globale.

Le suivi de la **somme globale** des pondérations,

$$\Sigma(t) = \sum_{i,j} \omega_{i,j}(t),$$

permet de détecter des tendances globales d'emballement ou d'inhibition. Une croissance exponentielle de $\Sigma(t)$ ou, inversement, une diminution drastique vers zéro, signale une perturbation de la dynamique du réseau. Par ailleurs, le ratio entre la valeur maximale et la valeur minimale,

$$R(t) = \frac{\max_{i,j} \omega_{i,j}(t)}{\min_{i,j} \omega_{i,j}(t)},$$

peut servir d'indicateur de dispersion extrême. Si $R(t)$ augmente de manière anormale, cela suggère que certaines liaisons se renforcent de manière disproportionnée par rapport aux autres, ce qui peut être symptomatique d'une dérive globale.

Les propriétés topologiques du SCN, telles que la **modularité** et la **connectivité** des clusters, sont également des indicateurs pertinents. La modularité Q d'une partition du graphe, qui quantifie la densité des liens à l'intérieur des clusters par rapport aux liens entre les clusters, peut être calculée pour évaluer la qualité des communautés formées. Une fluctuation soudaine de Q pourrait indiquer que la structure interne du SCN se modifie de manière inattendue, potentiellement en raison d'un emballement des ω .

C. Méthodes de Surveillance et Seuils d'Alarme

Afin de garantir que le SCN reste dans un état de cohérence global acceptable, il est nécessaire de mettre en place des systèmes de surveillance qui calculent périodiquement ces indicateurs. Deux approches complémentaires peuvent être déployées :

- **Surveillance en continu** : À chaque itération (ou à intervalles réguliers), on collecte un échantillon représentatif de la matrice ω et on calcule les indicateurs (moyenne, variance, entropie, somme globale). Ces valeurs sont comparées à des plages acceptables, définies à partir d'un historique ou d'un modèle théorique. Un dépassement des seuils prédéfinis déclenche une alerte.
- **Analyses « offline »** : Des snapshots de la matrice ω sont stockés périodiquement pour permettre une analyse plus approfondie des tendances sur le long terme. Ces analyses rétrospectives offrent la possibilité de détecter des dérives lentes mais significatives et de reconfigurer le système en cas de besoin.

D. Considérations Mathématiques et Pratiques

D'un point de vue mathématique, l'étude de la distribution des pondérations peut être assimilée à une analyse de la stabilité d'un système dynamique. Le SCN, qui évolue selon

$$\omega(t+1) = F(\omega(t)),$$

doit converger vers un état stable ou présenter une distribution stationnaire. L'introduction de contrôles statistiques (moyenne, variance, entropie) constitue une forme de rétroaction qui permet de comparer l'état actuel à l'état attendu, en appliquant des tests d'hypothèse pour détecter des écarts significatifs. Par exemple, si l'on définit une fenêtre de confiance $\mu \pm k\sigma$ (avec k fixé par la théorie ou l'expérience), toute valeur dépassant cette fenêtre peut être considérée comme une anomalie.

Sur le plan pratique, ces calculs doivent être implémentés de manière efficace, notamment dans des environnements distribués où le nombre de liaisons peut être très élevé. L'utilisation de techniques d'échantillonnage et l'agrégation des statistiques sur des bases de données NoSQL ou des systèmes de traitement de graphes permettent de réduire le coût computationnel tout en assurant une surveillance robuste.

Conclusion

Le calcul d'indicateurs de cohérence globale est une étape essentielle pour assurer que la matrice ω d'un SCN conserve une distribution cohérente et conforme aux attentes théoriques, malgré les perturbations ou manipulations potentielles. En surveillant des indicateurs tels que la moyenne, la variance, l'entropie, la somme globale et la modularité, il est possible de détecter des dérives systémiques qui pourraient compromettre la formation des clusters et l'auto-organisation du réseau. Ces indicateurs, calculés en continu

ou de manière périodique, permettent de déclencher des alarmes et d'initier des actions correctives pour maintenir la qualité et la stabilité du SCN. Sur le plan mathématique, ces outils fournissent un regard global sur la distribution des pondérations et servent de base pour ajuster dynamiquement les paramètres du système (par exemple, η et τ) ou activer des mécanismes de contrôle supplémentaire pour prévenir toute dérive systématique. Ainsi, le suivi des indicateurs de cohérence globale est un levier crucial pour garantir la résilience et la fiabilité d'un SCN en fonctionnement.

5.9. Exemples d'Implémentations et Études de Cas

5.9.1. SCN Multimodal

- 5.9.1.1. Schéma modulaire : *ModuleSynergySub*, *ModuleSynergySym*, *SCNCore*, etc.
- 5.9.1.2. Expliquer comment se fait la boucle de mise à jour concrètement.

5.9.2. SCN Hybride (Symbolique + Sub-symbolique)

- 5.9.2.1. Entités logiques, vecteurs, gestion via la synergie multiple.
- 5.9.2.2. Rapide cas d'usage, mini-dataset.

5.9.3. Robotique ou Multi-Agent

- 5.9.3.1. Rappeler la distribution possible, la synchro inter-robots.
- 5.9.3.2. Visualisation d'un SCN en essaim robotique.

5.9. Exemples d'Implémentations et Études de Cas

Dans cette dernière partie du chapitre 5, nous proposons d'illustrer **concrètement** comment un **SCN** (Synergistic Connection Network) peut être **implémenté** ou employé dans des situations variées, montrant ainsi la **mise en œuvre** pratique des concepts théoriques et d'architecture présentés jusque-là. Les études de cas (5.9.1, 5.9.2, 5.9.3) visent à donner des **exemples concrets** — chacun abordant un type différent de données ou d'environnement — et à détailler la **structure logicielle** (modules, boucles de mise à jour, etc.) nécessaire pour un fonctionnement effectif. Même si l'on s'est concentré en amont sur la dimension mathématique et la distribution (chap. 5.7), ces exemples montrent comment passer du cadre théorique à l'expérimentation ou à l'application dans des domaines multimodaux, hybrides (symbolique/sub-symbolique) ou robotiques.

5.9.1. SCN Multimodal

Lorsqu'un SCN doit manipuler plusieurs **types** de données (images, sons, textes, etc.) au sein d'un même réseau, il doit non seulement organiser la distribution ou la gestion de la matrice ω , mais aussi prévoir des **modules** de synergie adaptés à chaque modalité. La sous-section (5.9.1.1) décrit un **schéma modulaire** commun, puis (5.9.1.2) s'attache à la **boucle de mise à jour** concrète.

Dans cette section, nous développons en détail le schéma modulaire appliqué à un **Synergistic Connection Network** (SCN) multimodal, en mettant en évidence les différents modules qui interviennent dans le calcul de la synergie entre entités hétérogènes, ainsi que leur intégration dans le cœur du SCN. Nous présenterons d'abord le cadre théorique et mathématique de cette approche, puis nous proposerons une implémentation en Python de l'ensemble des modules évoqués, à savoir : *ModuleSynergySub*, *ModuleSynergySym* et *SCNCore*. Cette démarche permet d'illustrer comment, en séparant la logique de calcul de similarité selon les modalités (sub-symbolique vs. symbolique), on obtient une architecture évolutive, lisible et extensible.

I. Schéma Modulaire : Cadre Théorique et Mathématique

A. Contexte Multimodal et Besoin de Modulation

Dans de nombreux systèmes d'auto-organisation, notamment dans le cadre du **Deep Synergy Learning (DSL)**, les entités $\{\mathcal{E}_i\}$ ne sont pas homogènes. Certaines représentent des caractéristiques d'images (extraits d'un CNN), d'autres des descripteurs audio (par exemple, MFCC, spectrogrammes), et d'autres encore des tokens ou des concepts issus du traitement du langage naturel. Chaque entité est ainsi associée à un espace particulier \mathcal{X}_i , ce qui impose de définir la fonction de synergie $S(i, j)$ de manière adaptée à la modalité considérée.

Pour gérer cette hétérogénéité, il est naturel de séparer le calcul de la synergie en plusieurs modules spécialisés :

- **ModuleSynergySub** s'occupe des représentations sub-symboliques (images, audio, signaux sensoriels) et implémente des méthodes de calcul de similarité adaptées à des vecteurs continus (par exemple, similarité cosinus pour les images, distance inversée pour l'audio).
- **ModuleSynergySym** prend en charge le calcul de la synergie pour des entités symboliques, par exemple en exploitant des méthodes issues de l'analyse sémantique ou de la co-occurrence dans des espaces discrets.
- Le **SCNCore** constitue le noyau du système. Il maintient la matrice de pondérations ω qui encode les liaisons entre entités et applique la règle de mise à jour typique, par exemple dans sa forme additive :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Pour calculer $S(i,j)$, le *SCNCore* délègue l'opération au module spécialisé approprié, selon la nature des entités i et j .

Cette factorisation, qui revient mathématiquement à écrire

$$S(i,j) = f_{m(i,j)}(\mathcal{E}_i, \mathcal{E}_j),$$

où $m(i,j)$ désigne la modalité (ou la combinaison de modalités) concernée, permet non seulement d'alléger la complexité logicielle (en évitant un “god object” regroupant toutes les conditions) mais aussi d'assurer une extensibilité du système, chaque module pouvant être mis à jour ou remplacé indépendamment.

B. Bénéfices du Schéma Modulaire

- **Séparation des responsabilités** : chaque module est responsable d'un type spécifique de calcul de similarité. Cela permet de garantir la clarté du code, de faciliter le débogage et la maintenance, ainsi que l'ajout de nouvelles modalités.
- **Extensibilité** : l'architecture modulaire autorise l'intégration de nouvelles sources de données sans modifier le cœur de la dynamique $\omega(t)$. Par exemple, l'ajout d'une nouvelle modalité (données Lidar, signaux physiologiques, etc.) ne nécessite que l'implémentation d'un nouveau module de calcul de synergie.
- **Réutilisabilité et Déploiement Distribué** : chaque module peut être déployé sous forme de microservice indépendant, permettant une architecture distribuée où chaque sous-SCN conserve une structure modulaire similaire.

II. Implémentation en Python

Nous proposons ci-dessous une implémentation en Python de l'architecture modulaire décrite. Le code comporte trois classes principales :

- **ModuleSynergy** (classe de base abstraite)
- **ModuleSynergySub** et **ModuleSynergySym**, qui dérivent de la classe de base et implémentent chacune une méthode de calcul de similarité adaptée aux représentations sub-symboliques et symboliques, respectivement.
- **SCNCore** qui gère la matrice de pondérations ω et orchestre les mises à jour en appelant le module de synergie approprié selon la modalité des entités.

Pour illustrer cette implémentation, nous définirons également une classe **Entity** permettant de représenter les entités multimodales.

Voici le code complet :

```
import numpy as np
from abc import ABC, abstractmethod
from typing import Any, Dict, List, Tuple

# Définition d'une classe Entity pour encapsuler les entités multimodales
class Entity:
    """
    Représente une entité dans le SCN.

    Attributes:
    id (str): Identifiant unique de l'entité.
    modality (str): Type de modalité ('sub' pour sub-symbolique, 'sym' pour symbolique).
    data (Any): Données associées à l'entité.
```

```

"""
def __init__(self, id: str, modality: str, data: Any):
    self.id = id
    self.modality = modality
    self.data = data

# Classe de base abstraite pour les modules de calcul de synergie
class ModuleSynergy(ABC):
    """
Classe abstraite définissant l'interface pour le calcul de la synergie entre deux entités.
    """

    @abstractmethod
    def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
        """
Calcule et renvoie un score de similarité entre entity1 et entity2.
        """
        pass

# ModuleSynergySub : pour les représentations sub-symboliques (images, audio, etc.)
class ModuleSynergySub(ModuleSynergy):
    """
Module pour le calcul de la synergie entre entités sub-symboliques.
On suppose que les données sont des vecteurs numpy.
    """

    def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
        # Vérifier que les deux entités appartiennent à la modalité sub-symbolique
        if entity1.modality != 'sub' or entity2.modality != 'sub':
            raise ValueError("ModuleSynergySub ne peut traiter que des entités sub-symboliques.")

# Par exemple, on peut utiliser la similarité cosinus
vec1 = np.array(entity1.data)
vec2 = np.array(entity2.data)
# Calcul de la similarité cosinus
norm1 = np.linalg.norm(vec1)
norm2 = np.linalg.norm(vec2)
if norm1 == 0 or norm2 == 0:
    return 0.0
similarity = np.dot(vec1, vec2) / (norm1 * norm2)
# On peut normaliser pour qu'elle soit dans [0, 1] (si besoin)
return max(0.0, similarity)

# ModuleSynergySym : pour les entités symboliques (textes, concepts, etc.)
class ModuleSynergySym(ModuleSynergy):
    """
Module pour le calcul de la synergie entre entités symboliques.
Ici, on utilise une méthode simplifiée basée sur la correspondance exacte ou une mesure de similarité simple.
    """

    def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
        if entity1.modality != 'sym' or entity2.modality != 'sym':
            raise ValueError("ModuleSynergySym ne peut traiter que des entités symboliques.")

# Par exemple, si les données sont des chaînes de caractères représentant des concepts,
# on renvoie 1.0 si elles sont identiques, 0.0 sinon, ou on pourrait utiliser une mesure plus fine.
if entity1.data == entity2.data:
    return 1.0
else:

```

```

return 0.0

# SCNCore : le cœur du SCN qui maintient la matrice des pondérations et orchestre la mise à jour.
class SCNCORE:
    """
        SCNCORE gère la dynamique du réseau en conservant la matrice des pondérations  $\omega$ 
        et en appliquant la règle de mise à jour sur la base des scores de synergie.
    """
    def __init__(self, entities: List[Entity], eta: float = 0.1, tau: float = 0.2):
        """
            Initialise le SCNCORE avec la liste d'entités, le taux d'apprentissage (eta) et le coefficient de décroissance (tau).
            La matrice  $\omega$  est initialisée aléatoirement pour toutes les paires d'entités.
        """
        self.entities = entities
        self.N = len(entities)
        self.eta = eta
        self.tau = tau
        # Initialisation de la matrice  $\omega$  avec de petites valeurs aléatoires
        self.omega = np.random.uniform(0.01, 0.05, size=(self.N, self.N))
        # Pour garantir la symétrie, on peut symétriser la matrice
        self.omega = (self.omega + self.omega.T) / 2.0

        # Instanciation des modules de synergie
        self.module_synergy_sub = ModuleSynergySub()
        self.module_synergy_sym = ModuleSynergySym()

    def compute_synergy(self, i: int, j: int) -> float:
        """
            Calcule le score de synergie entre les entités d'indices  $i$  et  $j$ 
            en fonction de leur modalité.
        """
        entity1 = self.entities[i]
        entity2 = self.entities[j]
        # Si les deux entités ont la modalité sub-symbolique, utiliser ModuleSynergySub
        if entity1.modality == 'sub' and entity2.modality == 'sub':
            return self.module_synergy_sub.compute_similarity(entity1, entity2)
        # Si les deux entités ont la modalité symbolique, utiliser ModuleSynergySym
        elif entity1.modality == 'sym' and entity2.modality == 'sym':
            return self.module_synergy_sym.compute_similarity(entity1, entity2)
        # Pour des paires de modalités différentes, on définit une stratégie de fusion (ici, on prend la moyenne)
        else:
            score_sub = 0.0
            score_sym = 0.0
            count = 0
            if entity1.modality == 'sub' or entity2.modality == 'sub':
                try:
                    score_sub = self.module_synergy_sub.compute_similarity(entity1, entity2)
                    count += 1
                except ValueError:
                    pass
            if entity1.modality == 'sym' or entity2.modality == 'sym':
                try:
                    score_sym = self.module_synergy_sym.compute_similarity(entity1, entity2)
                    count += 1
                except ValueError:
                    pass

```

```

# Si aucun module n'est applicable, renvoyer 0
if count == 0:
    return 0.0
return (score_sub + score_sym) / count

def update_weights(self):
    """
    Met à jour la matrice des pondérations  $\omega$  selon la règle :
     $\omega(t+1) = \omega(t) + \eta [S(i,j) - \tau \omega(t)]$ 
    Pour chaque paire  $(i,j)$  avec  $i < j$ , la mise à jour est effectuée et
    la symétrie est rétablie.
    """
    new_omega = self.omega.copy()
    for i in range(self.N):
        for j in range(i+1, self.N):
            S_ij = self.compute_synergy(i, j)
            # Calcul de la mise à jour pour la paire (i,j)
            delta = self.eta * (S_ij - self.tau * self.omega[i, j])
            new_val = self.omega[i, j] + delta
            new_omega[i, j] = new_val
            new_omega[j, i] = new_val # Symétrie
    self.omega = new_omega

def run_iterations(self, iterations: int = 10):
    """
    Exécute un nombre d'itérations de mise à jour de la matrice  $\omega$  et affiche les valeurs moyennes.
    """
    for t in range(iterations):
        self.update_weights()
        mean_val = np.mean(self.omega)
        print(f"Itération {t+1}: moyenne des  $\omega$  = {mean_val:.4f}")

# Exemple d'utilisation du schéma modulaire dans un SCN multimodal
def main():
    # Création d'une liste d'entités avec différentes modalités
    entities = [
        # Entités sub-symboliques (images ou audio) : on simule avec des vecteurs numpy
        Entity("E1", "sub", [0.2, 0.4, 0.6]),
        Entity("E2", "sub", [0.1, 0.3, 0.5]),
        Entity("E3", "sub", [0.25, 0.35, 0.45]),
        # Entités symboliques (textuelles ou conceptuelles)
        Entity("E4", "sym", "chat"),
        Entity("E5", "sym", "chat"),
        Entity("E6", "sym", "chien"),
        # Entités mixtes : ici nous les traiterons par fusion (le calcul renverra la moyenne des scores disponibles)
        Entity("E7", "sub", [0.5, 0.2, 0.1]),
        Entity("E8", "sym", "oiseau")
    ]
    # Initialisation du cœur du SCN avec les entités
    scn_core = SCNCores(entities, eta=0.1, tau=0.2)

    # Affichage initial de la matrice des pondérations
    print("Matrice  $\omega$  initiale:")
    print(np.round(scn_core.omega, 4))

```

```

# Exécuter plusieurs itérations de mise à jour
scn_core.run_iterations(iterations=10)

# Affichage final de la matrice des pondérations
print("Matrice ω finale:")
print(np.round(scn_core.omega, 4))

if __name__ == "__main__":
    main()

```

Explications Complémentaires

1. ModuleSynergySub

Ce module est dédié au calcul de la synergie entre des entités dont les représentations sont sub-symboliques, c'est-à-dire des vecteurs continus (par exemple, les sorties d'un CNN pour des images ou des descripteurs audio). Dans l'implémentation proposée, nous utilisons la **similarité cosinus** :

$$\text{similarity}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

ce qui permet d'obtenir un score de similarité compris entre 0 et 1. Ce module lève une exception si des entités d'un type incompatible lui sont passées.

2. ModuleSynergySym

Ce module gère les entités de nature symbolique. Pour simplifier, nous utilisons ici un test d'égalité entre les données associées aux entités. Dans un contexte réel, on pourrait envisager des mesures plus fines (par exemple, des distances sémantiques basées sur des graphes ontologiques ou des embeddings pré-entraînés).

3. SCNCORE

Le cœur du SCN, *SCNCore*, orchestre la dynamique globale en maintenant la matrice des pondérations ω et en appliquant la règle de mise à jour. Pour chaque paire d'entités, il interroge le module de synergie approprié (en fonction de la modalité) pour obtenir $S(i, j)$ et met à jour $\omega_{i,j}$ selon le schéma DSL. La symétrie de la matrice est assurée lors de la mise à jour.

4. Cohabitation et Fusion

Dans le cas où deux entités présentent des modalités différentes (par exemple, une entité sub-symbolique et une entité symbolique), nous avons implémenté une stratégie simple qui tente de combiner les scores obtenus par chacun des modules. Si l'un des modules ne peut pas traiter la paire, le score de l'autre est utilisé seul. Ce mécanisme de fusion simple peut être étendu ou modifié selon les besoins spécifiques d'intégration multimodale.

Conclusion

La conception modulaire du SCN, telle que présentée dans la Section 5.9.1.1, repose sur la séparation des tâches de calcul de la synergie selon la nature des entités. En divisant le problème en sous-modules spécialisés (*ModuleSynergySub* pour les données sub-symboliques et *ModuleSynergySym* pour les données symboliques) et en centralisant la dynamique de mise à jour dans le *SCNCore*, nous obtenons une architecture claire, extensible et facilement maintenable. L'implémentation en Python ci-dessus traduit cette approche de manière concrète, offrant un cadre pédagogique complet pour la compréhension et l'expérimentation de SCN multimodaux dans un contexte de Deep Synergy Learning.

5.9.1.2. Expliquer comment se fait la boucle de mise à jour concrètement

Dans un SCN multimodal, la boucle de mise à jour est structurée de manière à intégrer deux aspects complémentaires :

- D'une part, le calcul du score de **synergie** $S(i,j)$ entre chaque paire d'entités, qui dépend de la nature (modalité) des données de chaque entité ;
- D'autre part, la mise à jour de la matrice des pondérations ω selon une règle d'auto-organisation (par exemple, une règle additive) qui se base sur le score $S(i,j)$.

Nous allons décrire concrètement les étapes d'une itération complète de mise à jour du SCN.

A. Les Étapes de la Boucle de Mise à Jour

Pour chaque paire d'entités (i,j) , le SCN doit déterminer la valeur de synergie qui mesure la « compatibilité » ou l'**affinité** entre les deux entités. Dans un SCN multimodal, cette opération se fait de la manière suivante :

- **Identification de la modalité** : Chaque entité \mathcal{E}_i porte une étiquette (par exemple, « sub » pour sub-symbolique, « sym » pour symbolique). Le **SCNCore** détermine ainsi si la paire (i,j) doit être traitée par le module *ModuleSynergySub* (données continues, vecteurs issus d'images ou d'audio) ou par *ModuleSynergySym* (données symboliques ou conceptuelles).
- **Délégation du calcul** : Le SCNCore appelle le module de synergie approprié pour obtenir un score $S(i,j)$. Ce score est généralement normalisé (par exemple, dans l'intervalle $[0,1]$).

Une fois le score $S(i,j)$ connu, la mise à jour des pondérations se fait via une règle d'auto-organisation. Dans la forme additive classique, la mise à jour s'exprime par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

où :

- η est le **taux d'apprentissage**,
- τ est le **coefficent de décroissance** qui assure une régulation en empêchant $\omega_{i,j}$ de croître indéfiniment.

Ce calcul est effectué pour chaque paire (i,j) concernée. Dans une implémentation pratique, on ne met à jour que les liaisons actives (par exemple, celles qui sont au-dessus d'un certain seuil ou dans un voisinage défini) afin de limiter le coût computationnel lorsque le nombre d'entités est très grand.

Après la mise à jour des valeurs :

- **Logs et Traçabilité** : Chaque modification de $\omega_{i,j}$ est enregistrée dans un journal (log) pour permettre le suivi et la détection d'anomalies ultérieures.
- **Watchers** : Des modules de surveillance vérifient que l'incrément $\Delta\omega_{i,j}$ n'est pas trop important (par exemple, qu'il ne dépasse pas un seuil δ).
- **Synchronisation (pour les architectures distribuées)** : Dans un SCN réparti, la mise à jour locale est synchronisée avec d'autres blocs via des messages inter-blocs afin de garantir la cohérence globale du réseau.

Une fois que toutes les liaisons ont été mises à jour et que les contrôles ont été effectués, la nouvelle matrice $\omega(t+1)$ sert d'état de départ pour la prochaine itération. La boucle se répète ainsi, permettant au SCN de converger vers une organisation stable et de former des clusters représentatifs des synergies entre entités.

B. Implémentation en Python

Nous présentons ici un exemple d'implémentation qui intègre les modules suivants :

- **ModuleSynergySub** : pour le calcul de la synergie entre entités sub-symboliques (par exemple, en utilisant la similarité cosinus).
- **ModuleSynergySym** : pour le calcul de la synergie entre entités symboliques (ici, une simple comparaison d'égalité).
- **SCNCore** : le cœur du SCN qui maintient la matrice ω et orchestre la boucle de mise à jour.

Vous trouverez ci-dessous le code complet, avec des commentaires détaillés pour expliquer chaque étape.

```
import numpy as np
from abc import ABC, abstractmethod
from typing import Any, List

# Définition d'une classe Entity pour représenter une entité multimodale
class Entity:
    """
    Représente une entité dans le SCN.

    Attributs:
        id (str): Identifiant unique de l'entité.
        modality (str): 'sub' pour sub-symbolique, 'sym' pour symbolique.
        data (Any): Données associées (par exemple, un vecteur pour sub ou une chaîne pour sym).
    """

    def __init__(self, id: str, modality: str, data: Any):
        self.id = id
        self.modality = modality
        self.data = data

# Classe de base abstraite pour le calcul de synergie
class ModuleSynergy(ABC):
    @abstractmethod
    def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
        """
        Calcule et renvoie un score de similarité entre entity1 et entity2.
        """
        pass

# Module pour les données sub-symboliques
class ModuleSynergySub(ModuleSynergy):
    def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
        # Vérification des modalités
        if entity1.modality != 'sub' or entity2.modality != 'sub':
            raise ValueError("ModuleSynergySub ne peut traiter que des entités sub-symboliques.")
        # On suppose que les données sont des vecteurs numpy
        vec1 = np.array(entity1.data)
        vec2 = np.array(entity2.data)
        norm1 = np.linalg.norm(vec1)
        norm2 = np.linalg.norm(vec2)
        if norm1 == 0 or norm2 == 0:
            return 0.0
        similarity = np.dot(vec1, vec2) / (norm1 * norm2)
        # On normalise pour obtenir un score dans [0, 1]
```

```

return max(0.0, similarity)

# Module pour les données symboliques
class ModuleSynergySym(ModuleSynergy):
    def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
        if entity1.modality != 'sym' or entity2.modality != 'sym':
            raise ValueError("ModuleSynergySym ne peut traiter que des entités symboliques.")
        # Méthode simple : renvoie 1 si les données sont identiques, 0 sinon
        return 1.0 if entity1.data == entity2.data else 0.0

# SCNCore : le cœur du SCN
class SCNCores:
    def __init__(self, entities: List[Entity], eta: float = 0.1, tau: float = 0.2):
        """
        Initialise le SCN avec une liste d'entités, et définit les paramètres  $\eta$  et  $\tau$ .
        La matrice  $\omega$  est initialisée avec de petites valeurs aléatoires.
        """
        self.entities = entities
        self.N = len(entities)
        self.eta = eta
        self.tau = tau
        # Initialisation de la matrice  $\omega$  (symétrique) de dimension  $N \times N$ 
        self.omega = np.random.uniform(0.01, 0.05, size=(self.N, self.N))
        self.omega = (self.omega + self.omega.T) / 2.0

    # Instanciation des modules de synergie
    self.module_synergy_sub = ModuleSynergySub()
    self.module_synergy_sym = ModuleSynergySym()

    def compute_synergy(self, i: int, j: int) -> float:
        """
        Calcule le score de synergie  $S(i,j)$  en fonction des modalités des entités.
        """
        e1 = self.entities[i]
        e2 = self.entities[j]
        if e1.modality == 'sub' and e2.modality == 'sub':
            return self.module_synergy_sub.compute_similarity(e1, e2)
        elif e1.modality == 'sym' and e2.modality == 'sym':
            return self.module_synergy_sym.compute_similarity(e1, e2)
        else:
            # Pour les cas mixtes, on combine les scores (ici par moyenne simple)
            scores = []
            try:
                scores.append(self.module_synergy_sub.compute_similarity(e1, e2))
            except ValueError:
                pass
            try:
                scores.append(self.module_synergy_sym.compute_similarity(e1, e2))
            except ValueError:
                pass
            return np.mean(scores) if scores else 0.0

    def update_weights(self):
        """
        Met à jour la matrice  $\omega$  pour chaque paire  $(i, j)$  en appliquant la règle :
         $\omega(t+1) = \omega(t) + \eta [S(i,j) - \tau \omega(t)]$ 
        """


```

```

"""
new_omega = self.omega.copy()
for i in range(self.N):
    for j in range(i + 1, self.N):
        S_ij = self.compute_synergy(i, j)
        delta = self.eta * (S_ij - self.tau * self.omega[i, j])
        new_val = self.omega[i, j] + delta
        # Application éventuelle d'un mécanisme de saturation (ici simple clamping)
        new_val = min(new_val, 1.0) # par exemple, on impose ω_max = 1.0
        new_omega[i, j] = new_val
        new_omega[j, i] = new_val # symétrie
self.omega = new_omega

def run_update_cycle(self, iterations: int = 10):
"""
Exécute la boucle de mise à jour pour un nombre donné d'itérations.
À chaque itération, on met à jour ω et on peut enregistrer l'état ou effectuer des contrôles.
"""

for t in range(iterations):
    self.update_weights()
    mean_omega = np.mean(self.omega)
    print(f"Itération {t+1} : moyenne de ω = {mean_omega:.4f}")
    # Ici, on pourrait ajouter des appels à des modules de logs ou watchers

# Exemple d'utilisation
def main():
    # Création d'un ensemble d'entités multimodales
    entities = [
        # Entités sub-symboliques : représentées par des vecteurs (ex. d'images ou d'audio)
        Entity("E1", "sub", [0.2, 0.4, 0.6]),
        Entity("E2", "sub", [0.1, 0.3, 0.5]),
        Entity("E3", "sub", [0.25, 0.35, 0.45]),
        # Entités symboliques : représentées par des chaînes de caractères
        Entity("E4", "sym", "chat"),
        Entity("E5", "sym", "chat"),
        Entity("E6", "sym", "chien"),
        # Entités mixtes, pour démonstration, nous utilisons un calcul de fusion simple
        Entity("E7", "sub", [0.5, 0.2, 0.1]),
        Entity("E8", "sym", "oiseau")
    ]

    # Initialisation du SCNCore avec les entités
    scn_core = SCNCore(entities, eta=0.1, tau=0.2)
    print("Matrice ω initiale :")
    print(np.round(scn_core.omega, 4))

    # Exécution de la boucle de mise à jour sur 10 itérations
    scn_core.run_update_cycle(iterations=10)

    print("Matrice ω finale :")
    print(np.round(scn_core.omega, 4))

if __name__ == "__main__":
    main()

```

A. Cycle d’Itération

Chaque itération dans le SCN se décompose en plusieurs étapes :

286. Calcul de $S(i, j)$

Pour chaque paire (i, j) (typiquement, on parcourt uniquement $i < j$ pour maintenir la symétrie), le **SCNCore** interroge le module de synergie adéquat (sub-symbolique ou symbolique) via la méthode *compute_synergy*. Cette fonction retourne un score qui reflète l’affinité entre les entités i et j .

287. Application de la règle de mise à jour

À partir de la valeur $S(i, j)$ obtenue, la mise à jour se fait en ajoutant un terme $\eta [S(i, j) - \tau \omega_{i,j}]$ à la valeur actuelle de $\omega_{i,j}$. Dans notre exemple, nous appliquons aussi une saturation (clamping à 1.0) afin d’éviter des valeurs trop élevées.

288. Contrôles et Enregistrement

Après chaque mise à jour, des opérations de contrôle peuvent être exécutées (par exemple, via des watchers ou des logs) pour assurer la stabilité et la traçabilité de la dynamique. Dans notre implémentation, nous affichons la moyenne de la matrice ω pour suivre l’évolution globale.

289. Itération Suivante

La nouvelle matrice ω devient l’état de départ pour l’itération suivante. Cette boucle permet de simuler l’auto-organisation du réseau, qui converge progressivement vers une structure stable.

B. Gestion de la Multimodalité

Le **dispatching** de la fonction de calcul de synergie se fait dans la méthode *compute_synergy* de la classe **SCNCore**. Selon que les entités soient de type « sub » ou « sym », la méthode délègue le calcul au module approprié. Pour les cas mixtes, une stratégie simple de fusion (ici, la moyenne) est appliquée, mais cette logique peut être enrichie selon les exigences de votre système.

C. Extension et Intégration

Dans un système réel, d’autres modules pourraient être intégrés à ce pipeline :

- Des modules d’**inhibition** ou de **saturation** plus élaborés, qui ajusteraient les mises à jour en fonction de la dynamique globale.
- Des mécanismes de **logging** et de **surveillance** (voir les sections 5.8) qui enregistreraient et contrôleraient en temps réel la validité des mises à jour.
- Des protocoles de **synchronisation** pour des SCN distribués, permettant de garantir que la mise à jour se fasse de manière cohérente entre plusieurs nœuds.

Conclusion

La boucle de mise à jour dans un SCN multimodal se réalise en plusieurs étapes : d’abord, le calcul de la synergie $S(i, j)$ via des modules spécialisés qui traitent les modalités propres à chaque type d’entité, puis l’application de la règle de mise à jour sur la matrice des pondérations ω dans le SCNCore, suivi de contrôles complémentaires et enfin le passage à l’itération suivante. L’implémentation Python proposée illustre concrètement ce processus en séparant la logique de calcul de synergie (via *ModuleSynergySub* et *ModuleSynergySym*) de la gestion globale de la dynamique

d'auto-organisation (dans *SCNCore*). Ce schéma modulaire permet ainsi une architecture claire, extensible et apte à gérer efficacement la multimodalité, tout en assurant la robustesse et la cohérence du système.

Ci-dessous, vous trouverez un code complet qui reprend la structure modulaire déjà développée (avec les classes *Entity*, *ModuleSynergySub*, *ModuleSynergySym* et *SCNCore*), auquel nous ajoutons une classe **SynergyDispatcher** et une fonction **sen_distributed_iteration**. Ces éléments illustrent la manière dont le calcul de la synergie peut être dispatché selon le type des entités et comment, dans un scénario distribué, la mise à jour des pondérations se fait en tenant compte de l'appartenance locale ou non d'une entité.

```
import numpy as np
from abc import ABC, abstractmethod
from typing import Any, Dict, List, Tuple

#####
# 1. Classes de Base pour les Entités et Modules de Synergie
#####

class Entity:
    """
    Représente une entité dans le SCN.

    Attributs:
        id (str): Identifiant unique de l'entité.
        modality (str): Type de modalité, par exemple 'subsymbolic' ou 'symbolic'.
        data (Any): Données associées à l'entité (vecteur pour subsymbolic, chaîne pour symbolic).
    """

    def __init__(self, id: str, modality: str, data: Any):
        self.id = id
        self.modality = modality # Exemple : 'subsymbolic' ou 'symbolic'
        self.data = data

    # Classe abstraite pour le calcul de synergie
    class ModuleSynergy(ABC):
        @abstractmethod
        def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
            """
            Calcule et renvoie un score de similarité entre entity1 et entity2.
            """
            pass

        # Module pour les données subsymboliques (ex. images, audio)
        class ModuleSynergySub(ModuleSynergy):
            def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
                if entity1.modality != 'subsymbolic' or entity2.modality != 'subsymbolic':
                    raise ValueError("ModuleSynergySub ne peut traiter que des entités subsymboliques.")
                # Exemple : similarité cosinus sur des vecteurs
                vec1 = np.array(entity1.data)
                vec2 = np.array(entity2.data)
                norm1 = np.linalg.norm(vec1)
                norm2 = np.linalg.norm(vec2)
                if norm1 == 0 or norm2 == 0:
                    return 0.0
                similarity = np.dot(vec1, vec2) / (norm1 * norm2)
```

```

# On renvoie une valeur entre 0 et 1
return max(0.0, similarity)

# Module pour les données symboliques (ex. mots, concepts)
class ModuleSynergySym(ModuleSynergy):
    def compute_similarity(self, entity1: Entity, entity2: Entity) -> float:
        if entity1.modality != 'symbolic' or entity2.modality != 'symbolic':
            raise ValueError("ModuleSynergySym ne peut traiter que des entités symboliques.")
        # Exemple simple : 1.0 si les données sont identiques, 0 sinon.
        return 1.0 if entity1.data == entity2.data else 0.0

#####
# 2. SynergyDispatcher
#####

class SynergyDispatcher:
    """
    Classe qui répartit le calcul de la synergie selon le type des entités.
    Elle fait appel aux modules spécialisés pour les entités subsymboliques et symboliques.
    """
    def __init__(self, synergy_sub: ModuleSynergy, synergy_sym: ModuleSynergy):
        self.sub = synergy_sub
        self.sym = synergy_sym

    def get_synergy(self, entity1: Entity, entity2: Entity) -> float:
        # Si les deux entités sont subsymboliques
        if entity1.modality == "subsymbolic" and entity2.modality == "subsymbolic":
            return self.sub.compute_similarity(entity1, entity2)
        # Si les deux entités sont symboliques
        elif entity1.modality == "symbolic" and entity2.modality == "symbolic":
            return self.sym.compute_similarity(entity1, entity2)
        else:
            # Pour un cas cross-modal, on peut par exemple renvoyer la moyenne des scores obtenus sur les deux module
            s
            scores = []
            try:
                scores.append(self.sub.compute_similarity(entity1, entity2))
            except Exception:
                pass
            try:
                scores.append(self.sym.compute_similarity(entity1, entity2))
            except Exception:
                pass
            return np.mean(scores) if scores else 0.0

#####
# 3. SCNCORE (Version Simplifiée)
#####

class SCNCORE:
    """
    Le cœur du SCN qui gère la matrice des pondérations  $\omega$  et orchestre la mise à jour.
    """
    def __init__(self, entities: List[Entity], eta: float = 0.1, tau: float = 0.2):
        self.entities = entities
        self.N = len(entities)

```

```

self.eta = eta
self.tau = tau
# Initialisation de la matrice ω avec de petites valeurs aléatoires (symétrique)
self.omega = np.random.uniform(0.01, 0.05, size=(self.N, self.N))
self.omega = (self.omega + self.omega.T) / 2.0
# Instanciation du dispatcher pour calculer la synergie
self.dispatcher = SynergyDispatcher(ModuleSynergySub(), ModuleSynergySym())

def compute_synergy(self, i: int, j: int) -> float:
    """
    Calcule S(i,j) en déléguant au dispatcher.
    """
    return self.dispatcher.get_synergy(self.entities[i], self.entities[j])

def update_weights(self):
    """
    Met à jour la matrice ω selon la règle additive :
    ω(t+1) = ω(t) + η [ S(i,j) - τ ω(t) ]
    """
    new_omega = self.omega.copy()
    for i in range(self.N):
        for j in range(i + 1, self.N):
            S_ij = self.compute_synergy(i, j)
            delta = self.eta * (S_ij - self.tau * self.omega[i, j])
            new_val = self.omega[i, j] + delta
            # Optionnel : clamp pour éviter un dépassement excessif
            new_val = min(new_val, 1.0)
            new_omega[i, j] = new_val
            new_omega[j, i] = new_val # symétrie
    self.omega = new_omega

def run_iterations(self, iterations: int = 10):
    for t in range(iterations):
        self.update_weights()
        print(f"Itération {t+1}: moyenne de ω = {np.mean(self.omega):.4f}")

#####
# 4. Fonction pour la Mise à Jour dans un SCN Distribué
#####

def is_local(entity: Entity, local_block_ids: List[str]) -> bool:
    """
    Simule une fonction qui détermine si une entité appartient au bloc local.
    Ici, on suppose que l'identifiant de la modalité peut servir à cet effet.
    """
    return entity.id in local_block_ids

def apply_inhibition_saturation(w_value: float, i: int, j: int) -> float:
    """
    Applique éventuellement un mécanisme d'inhibition ou de saturation.
    Pour cet exemple, on effectue simplement un clamping à 1.0.
    """
    return min(w_value, 1.0)

# Exemple d'interface réseau simplifiée pour la communication inter-blocs.
class NetInterface:

```

```

def request_synergy_other_block(self, entity1: Entity, entity2: Entity) -> float:
    """
    Simule une requête pour obtenir  $S(i, j)$  depuis un autre bloc.
    Dans une implémentation réelle, cette méthode enverrait une requête réseau.
    """
    # Pour simplifier, on renvoie une valeur fixe (par exemple, 0.5)
    return 0.5

def send_local_updates(self, block_id: str, w_local: Dict[Tuple[int, int], float]):
    """
    Simule l'envoi des mises à jour locales vers un orchestrateur central.
    """
    print(f"Bloc {block_id}: envoi des mises à jour, nombre de liens mis à jour = {len(w_local)}")

```

```

def scn_distributed_iteration(block_id: str,
                               w_local: Dict[Tuple[int, int], float],
                               local_links: List[Tuple[int, int]],
                               synergy_module: SynergyDispatcher,
                               update_rule,
                               entities: List[Entity],
                               net_interface: NetInterface):
    """
    Fonction simulant une itération de mise à jour dans un SCN distribué.
    """

```

Paramètres:

block_id : identifiant du bloc.
w_local : dictionnaire local des pondérations ω pour les liaisons gérées par ce bloc.
local_links : liste des paires (i, j) appartenant à ce bloc.
synergy_module : instance de *SynergyDispatcher* pour calculer $S(i, j)$ localement.
update_rule : fonction qui applique la règle DSL, par exemple:

$$\text{update_rule}(w_{\text{old}}, s_{\text{val}}) = w_{\text{old}} + \eta (s_{\text{val}} - \tau * w_{\text{old}})$$

entities : liste globale des entités.
net_interface : interface pour échanger les mises à jour avec d'autres blocs.

```

for (i, j) in local_links:
    # Détermine si l'entité j est locale au bloc courant.
    # Pour cet exemple, nous supposons que la liste local_links ne contient que des paires locales.
    s_val = synergy_module.get_synergy(entities[i], entities[j])
    w_old = w_local.get((i, j), 0.0)
    w_new = update_rule(w_old, s_val)
    w_new = apply_inhibition_saturation(w_new, i, j)
    w_local[(i, j)] = w_new

    # Envoi des mises à jour locales via l'interface réseau
    net_interface.send_local_updates(block_id, w_local)

```

```

#####
# 5. Fonction d'Update Rule DSL (Exemple Additif)
#####

```

```

def update_rule_additive(w_old: float, s_val: float, eta: float = 0.1, tau: float = 0.2) -> float:
    """
    Applique la règle additive :  $w_{\text{new}} = w_{\text{old}} + \eta (s_{\text{val}} - \tau * w_{\text{old}})$ 
    """
    return w_old + eta * (s_val - tau * w_old)

```

```

#####
# 6. Exemple d'Utilisation Globale
#####

def main():
    # Création d'une liste d'entités (certaines subsymboliques et certaines symboliques)
    entities = [
        Entity("E1", "subsymbolic", [0.2, 0.4, 0.6]),
        Entity("E2", "subsymbolic", [0.1, 0.3, 0.5]),
        Entity("E3", "subsymbolic", [0.25, 0.35, 0.45]),
        Entity("E4", "symbolic", "chat"),
        Entity("E5", "symbolic", "chat"),
        Entity("E6", "symbolic", "chien")
    ]

    # Initialisation du SCNCORE pour une utilisation non distribuée
    scn_core = SCNCORE(entities, eta=0.1, tau=0.2)
    print("Matrice ω initiale:")
    print(np.round(scn_core.omega, 4))

    scn_core.run_iterations(iterations=5)
    print("Matrice ω finale:")
    print(np.round(scn_core.omega, 4))

    # Exemple de mise à jour distribuée
    # Supposons que le bloc local gère les entités E1, E2, E3 (indices 0,1,2)
    local_block_ids = ["E1", "E2", "E3"]
    local_links = [(0, 1), (0, 2), (1, 2)]
    # Initialisation d'un store local des pondérations pour ce bloc
    w_local = {(i, j): 0.05 for (i, j) in local_links}
    # Utilisation d'une instance de SynergyDispatcher (dépendant de la modalité)
    dispatcher = SynergyDispatcher(ModuleSynergySub(), ModuleSynergySym())
    # Création d'une interface réseau simulée
    net_interface = NetInterface()
    print("\nMise à jour distribuée pour le bloc 'Bloc_A' sur les entités locales E1, E2, E3:")
    scn_distributed_iteration("Bloc_A", w_local, local_links, dispatcher, update_rule_additive, entities, net_interface)
    print("Store local w après update:")
    print({k: round(v, 4) for k, v in w_local.items()})

if __name__ == "__main__":
    main()

```

Expliations Détailées

1. La Boucle de Mise à Jour dans le SCNCORE

- **Calcul de la Synergie $S(i,j)$:**

La méthode `compute_synergy` du **SCNCORE** fait appel au **SynergyDispatcher**, lequel, en fonction des modalités des entités i et j , appelle le module approprié (`ModuleSynergySub` ou `ModuleSynergySym`). Ainsi, la logique de calcul de la synergie est encapsulée et séparée du reste du système.

- **Application de la Règle DSL :**

La méthode `update_weights` parcourt toutes les paires (i, j) avec $i < j$, calcule $S(i, j)$, et applique la règle additive

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$$

Le résultat est ensuite éventuellement passé par une fonction de saturation (ici, on impose un maximum de 1.0).

- **Cycle d'Iteration :**

La méthode `run_iterations` effectue plusieurs itérations de mise à jour et affiche la moyenne de la matrice ω afin de suivre la progression globale.

2. La Fonction `scn_distributed_iteration`

Cette fonction simule le comportement d'un bloc dans un environnement distribué. Pour chaque liaison locale (définie dans `local_links`), la fonction :

- Vérifie si la paire est locale (pour simplifier, nous considérons ici que toutes les paires sont locales, mais la fonction pourrait être étendue pour interroger d'autres blocs via `net_interface`).
- Calcule $S(i, j)$ via le module de synergie (dispatcher).
- Applique la règle DSL (ici, `update_rule_additive`) pour obtenir la nouvelle valeur de $\omega_{i,j}$.
- Applique un mécanisme d'inhibition/saturation.
- Envoie, en fin de traitement, les mises à jour locales vers un orchestrateur via `net_interface.send_local_updates`.

3. La Classe `SynergyDispatcher`

Cette classe permet de déléguer le calcul de la synergie en fonction du type des entités. Elle teste la modalité de chaque entité et appelle le module correspondant. En cas de modalités mixtes, une stratégie de fusion (ici, une moyenne) est appliquée.

Conclusion

Ce développement illustre concrètement la boucle de mise à jour d'un SCN multimodal. La structure modulaire est assurée par la séparation entre le calcul de la synergie (géré par les modules `ModuleSynergySub`, `ModuleSynergySym` et leur dispatcher associé) et la gestion centrale de la dynamique des pondérations (dans `SCNCore`). De plus, dans un contexte distribué, la fonction `scn_distributed_iteration` démontre comment intégrer la mise à jour locale des pondérations avec la communication inter-blocs via une interface réseau. Cette architecture modulaire permet ainsi d'assurer une grande flexibilité et une évolutivité dans le traitement de données multimodales tout en maintenant la cohérence de la dynamique du SCN.

5.9.2.1. Entités Logiques, Vecteurs, Gestion via la Synergie Multiple

Dans un **Synergistic Connection Network** (SCN) hybride, l'objectif est de traiter simultanément des entités de nature très différente. D'une part, certaines entités sont purement sub-symboliques : il s'agit typiquement de vecteurs numériques issus de descripteurs (ex. : embeddings d'images, audio ou texte). D'autre part, d'autres entités sont purement symboliques et représentent des objets logiques, des formules ou des concepts issus d'ontologies. Enfin, il arrive fréquemment que certaines entités combinent ces deux aspects, comme par exemple un concept lexicalisé qui dispose à la fois d'un embedding vectoriel et d'une définition logique formalisée. Pour répondre à cette hétérogénéité, le SCN doit être capable de calculer plusieurs fonctions de similarité ou de compatibilité, puis de fusionner ces résultats

en un unique score $S(i, j)$ qui guidera l'auto-organisation du réseau. Ce mécanisme est désigné par le terme de **synergie multiple**.

A. Rappel : Sub-symbolique et Symbolique

Sur le plan mathématique, les **entités sub-symboliques** sont généralement représentées par des vecteurs x_i^{sub} appartenant à un espace vectoriel $\mathcal{X}_{\text{sub}} \subseteq \mathbb{R}^d$. Ces représentations numériques proviennent de techniques d'extraction de features, telles que celles issues des réseaux de neurones convolutionnels (CNN) pour les images, ou des méthodes de traitement du signal pour l'audio. La similarité entre deux entités sub-symboliques peut être évaluée par des mesures continues comme la similarité cosinus ou une fonction exponentielle de la distance euclidienne, par exemple :

$$S_{\text{sub}}(i, j) = \exp(-\alpha \|x_i^{\text{sub}} - x_j^{\text{sub}}\|^2) \quad \text{ou} \quad S_{\text{sub}}(i, j) = \frac{x_i^{\text{sub}} \cdot x_j^{\text{sub}}}{\|x_i^{\text{sub}}\| \|x_j^{\text{sub}}\|},$$

où $\alpha > 0$ est un paramètre de réglage.

En revanche, les **entités symboliques** sont représentées par des objets logiques ou des symboles, notés ξ_i^{sym} , qui vivent dans un espace non vectoriel, voire discret, \mathcal{X}_{sym} . Le calcul de leur synergie ne repose pas sur des distances métriques classiques, mais plutôt sur des critères de compatibilité sémantique ou logique. Par exemple, pour deux concepts issus d'une ontologie, on peut définir :

$$S_{\text{sym}}(i, j) = g(\xi_i^{\text{sym}}, \xi_j^{\text{sym}}),$$

où g est une fonction qui évalue le degré de similarité sémantique, pouvant être basée sur des mesures d'alignement ou de co-occurrence dans un graphe conceptuel.

B. Synergie Multiple : Principe et Fusion des Scores

Face à la nécessité de gérer des entités hétérogènes, le SCN hybride définit une **synergie globale** pour un couple d'entités (i, j) en combinant les contributions issues des aspects sub-symboliques et symboliques. Mathématiquement, cette combinaison se traduit par :

$$S_{\text{hyb}}(i, j) = \alpha S_{\text{sub}}(i, j) + \beta S_{\text{sym}}(i, j),$$

où α et β sont des poids non négatifs qui déterminent l'importance relative de chaque modalité. Le choix des coefficients α et β peut être fixe ou adaptatif en fonction des caractéristiques des entités :

- **Entités sub-symboliques pures** : on peut imposer $\beta = 0$ afin que seule la contribution vectorielle soit prise en compte.
- **Entités symboliques pures** : on peut fixer $\alpha = 0$.
- **Entités mixtes** : des valeurs telles que $\alpha = 0.7$ et $\beta = 0.3$ permettent de donner une importance plus forte à la représentation sub-symbolique tout en conservant une influence significative de la dimension symbolique.

D'autres stratégies de fusion sont envisageables. Par exemple, le score global pourrait être défini par le **produit** des sous-scores, ou par une fonction non linéaire telle qu'un maximum ou une combinaison pondérée par une fonction softmax. Ces stratégies ont pour objectif de garantir que le score $S(i, j)$ reflète à la fois la proximité dans l'espace vectoriel et la cohérence sémantique ou logique, renforçant ainsi la pertinence de la mise à jour de la matrice ω .

C. Stockage des Entités et Mises à Jour dans un Espace Hybride

Chaque entité \mathcal{E}_i dans un SCN hybride est représentée par une paire de composantes :

$$\mathcal{E}_i = (x_i^{\text{sub}}, \xi_i^{\text{sym}}),$$

où x_i^{sub} est un vecteur dans \mathbb{R}^d et ξ_i^{sym} est une représentation symbolique. La fonction de synergie globale $S_{\text{hyb}}(i, j)$ est alors utilisée dans la règle de mise à jour de la matrice des pondérations, qui demeure inchangée par rapport à la formulation classique du DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hyb}}(i, j) - \tau \omega_{i,j}(t)].$$

Ici, l'apport de la synergie multiple permet d'enrichir le processus d'auto-organisation en tenant compte des deux dimensions de l'information. Ce mécanisme assure que la connexion entre deux entités est renforcée si elles présentent à la fois une similarité vectorielle élevée et une forte compatibilité sémantique.

Sur le plan **ingénierie**, cette approche se traduit par l'implémentation d'un module de type *ModuleSynergyHyb* ou, de manière plus générale, par un **dispatcher** capable d'orienter le calcul vers les modules spécialisés appropriés, puis de fusionner les résultats selon une stratégie choisie (par exemple, la somme pondérée présentée ci-dessus).

Conclusion

La gestion des entités logiques et sub-symboliques dans un SCN hybride repose sur une approche de **synergie multiple** qui combine plusieurs fonctions de similarité. Chaque entité \mathcal{E}_i est dotée d'une représentation duale, x_i^{sub} pour les aspects sub-symboliques et ξ_i^{sym} pour les aspects symboliques, et la synergie globale entre deux entités est obtenue par une combinaison linéaire (ou autre) des sous-scores :

$$S_{\text{hyb}}(i, j) = \alpha f_{\text{sub}}(x_i^{\text{sub}}, x_j^{\text{sub}}) + \beta g_{\text{sym}}(\xi_i^{\text{sym}}, \xi_j^{\text{sym}}).$$

D'un point de vue mathématique, cette approche enrichit la définition habituelle de $S(i, j)$ en la rendant capable de capturer la dualité des informations disponibles, tandis que, d'un point de vue ingénierie, elle s'appuie sur des modules spécialisés qui se combinent dans un schéma modulaire et extensible. Ce modèle hybride permet ainsi au SCN de former des clusters qui sont cohérents à la fois sur le plan des représentations numériques et sur le plan des relations sémantiques ou logiques, garantissant une auto-organisation plus robuste et pertinente dans des environnements complexes et multimodaux.

5.9.2.2. Rapide cas d'usage, mini-dataset

Dans un SCN hybride, l'objectif est de combiner des entités provenant de deux mondes différents : les entités sub-symboliques qui sont représentées par des vecteurs numériques et vivent dans un espace continu $\mathcal{X}_{\text{sub}} \subseteq \mathbb{R}^d$, et les entités symboliques qui se présentent sous forme de règles logiques ou d'axiomes et qui n'ont pas de représentation vectorielle directe. Dans de nombreuses applications, il est intéressant de fusionner ces deux dimensions afin de mieux capturer la richesse des informations disponibles sur les entités.

A. Description du Mini-Dataset

Pour illustrer le concept, considérons un mini-dataset composé de 10 entités réparties comme suit :

290. Entités Sub-symboliques (les entités \mathcal{E}_1 à \mathcal{E}_5)

Chaque entité est représentée par un vecteur de dimension 4. Par exemple :

- $\mathcal{E}_1: (0.2, 0.8, 0.1, 0.1)$
- $\mathcal{E}_2: (0.1, 0.75, 0.05, 0.3)$

- $\mathcal{E}_3: (0.6, 0.2, 0.05, 0.15)$
- $\mathcal{E}_4: (0.3, 0.6, 0.2, 0.1)$
- $\mathcal{E}_5: (0.25, 0.65, 0.15, 0.2)$
- **Entités Symboliques** (les entités \mathcal{E}_6 à \mathcal{E}_{10})

Chaque entité est représentée par un ensemble simple de règles ou d'axiomes. Par exemple :

- $\mathcal{E}_6: \{A \rightarrow B, B \wedge C \rightarrow D\}$
- $\mathcal{E}_7: \{X \rightarrow Y, \neg(Z \wedge Y)\}$
- $\mathcal{E}_8: \{P \rightarrow Q\}$
- $\mathcal{E}_9: \{A \rightarrow B, B \rightarrow C\}$
- $\mathcal{E}_{10}: \{M \rightarrow N, N \wedge O \rightarrow P\}$

Ce mini-dataset représente ainsi un échantillon réduit d'entités où la première partie apporte une information continue (ex. issue d'un moteur NLP ou d'un CNN), et la seconde des informations purement logiques. L'idée est d'observer comment le SCN peut, à travers le calcul d'une **synergie multiple**, regrouper ces entités selon des critères qui tiennent compte simultanément de la proximité vectorielle et de la compatibilité sémantique.

B. Principe de la Synergie Multiple

Pour chaque paire d'entités (i, j) , on définit la synergie globale $S(i, j)$ comme une combinaison des deux sous-synergies :

$$S(i, j) = \alpha S_{\text{sub}}(i, j) + \beta S_{\text{sym}}(i, j),$$

où :

- $S_{\text{sub}}(i, j)$ est la similarité entre les représentations vectorielles (ex. similarité cosinus) pour les entités sub-symboliques ;
- $S_{\text{sym}}(i, j)$ est un score de compatibilité logique ou sémantique, qui peut être calculé par le recouvrement ou l'alignement des règles ;
- α et β sont des coefficients (non négatifs) qui pondèrent l'influence de chaque modalité. Par exemple, pour des entités mixtes, on pourra choisir $\alpha = 0.7$ et $\beta = 0.3$.

Si les entités sont de même nature (pures sub-symboliques ou purement symboliques), on peut adapter ces coefficients en conséquence (ex. $\beta = 0$ pour des paires sub-symboliques pures).

C. La Règle de Mise à Jour DSL dans le Contexte Hybride

La mise à jour de la matrice des pondérations ω suit la règle classique du DSL, c'est-à-dire :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

où le score $S(i, j)$ est ici le score hybride $S(i, j) = \alpha S_{\text{sub}}(i, j) + \beta S_{\text{sym}}(i, j)$. Ainsi, le renforcement de la liaison entre deux entités repose sur l'accord simultané de leurs représentations vectorielles et de leurs composantes logiques. L'objectif est que la dynamique renforce les liens entre entités « proches » sur les deux dimensions, formant ainsi des clusters cohérents tant sur le plan neuronal (sub-symbolique) que sur le plan sémantique (symbolique).

Implémentation en Python du Cas d'Usage avec Mini-Dataset

Nous proposons ci-dessous un code Python qui illustre la gestion d'un mini-dataset hybride par un SCN. Ce code définit :

- Une classe **Entity** qui intègre une représentation sub-symbolique et/ou symbolique.
- Deux modules de synergie, l'un pour les entités subsymboliques et l'autre pour les entités symboliques.
- Un module hybride qui fusionne ces deux scores à l'aide des coefficients α et β .
- Un **SCNCore** qui applique la règle de mise à jour DSL sur la matrice des pondérations ω à partir des scores hybrides.

```
import numpy as np
from abc import ABC, abstractmethod
from typing import Any, List
```

```
#####
# 1. Classe Entity pour le Mini-dataset Hybride
#####
```

```
class Entity:
```

```
    """
```

Représente une entité hybride dans le SCN.

Chaque entité combine une représentation sub-symbolique (vecteur) et une représentation symbolique (ensemble de règles ou axiomes).

Attributes:

id (str): Identifiant unique.

modality (str): 'hybrid' pour les entités combinées.

sub_data (List[float]): Vecteur numérique (peut être None pour entités purement symboliques).

sym_data (Any): Représentation symbolique (par exemple, une liste ou un set de règles).

```
"""
```

```
def __init__(self, id: str, sub_data: List[float] = None, sym_data: Any = None):
    self.id = id
    self.modality = 'hybrid'
    self.sub_data = sub_data
    self.sym_data = sym_data
```

```
#####
# 2. Modules de Synergie pour les Composantes
#####
```

```
class ModuleSynergySub(ABC):
```

```
    @abstractmethod
```

```
    def compute_similarity(self, vec1: List[float], vec2: List[float]) -> float:
        pass
```

```
class CosineSimilarity(ModuleSynergySub):
```

```
    def compute_similarity(self, vec1: List[float], vec2: List[float]) -> float:
        v1 = np.array(vec1)
        v2 = np.array(vec2)
        norm1 = np.linalg.norm(v1)
        norm2 = np.linalg.norm(v2)
        if norm1 == 0 or norm2 == 0:
```

```

        return 0.0
similarity = np.dot(v1, v2) / (norm1 * norm2)
return max(0.0, similarity)

class ModuleSynergySym(ABC):
    @abstractmethod
    def compute_similarity(self, rules1: Any, rules2: Any) -> float:
        pass

class SimpleRuleMatch(ModuleSynergySym):
    def compute_similarity(self, rules1: List[str], rules2: List[str]) -> float:
        """
        Calcule la similarité comme le rapport du nombre de règles communes sur le nombre total de règles.
        """
        if not rules1 or not rules2:
            return 0.0
        set1, set2 = set(rules1), set(rules2)
        common = set1.intersection(set2)
        total = set1.union(set2)
        return len(common) / len(total) if total else 0.0

#####
# 3. Module de Synergie Hybride (Dispatcher)
#####

class ModuleSynergyHyb:
    """
    Calcule le score hybride  $S(i, j)$  pour des entités hybrides.

    La synergie globale est définie par :
    
$$S_{hyb}(i,j) = \alpha * S_{sub}(i,j) + \beta * S_{sym}(i,j)$$

    """

    def __init__(self, alpha: float = 0.7, beta: float = 0.3):
        self.alpha = alpha
        self.beta = beta
        self.sub_module = CosineSimilarity()
        self.sym_module = SimpleRuleMatch()

    def compute_hybrid_similarity(self, entity1: Entity, entity2: Entity) -> float:
        # Calcul de la similarité sub-symbolique
        if entity1.sub_data is not None and entity2.sub_data is not None:
            s_sub = self.sub_module.compute_similarity(entity1.sub_data, entity2.sub_data)
        else:
            s_sub = 0.0
        # Calcul de la similarité symbolique
        if entity1.sym_data is not None and entity2.sym_data is not None:
            s_sym = self.sym_module.compute_similarity(entity1.sym_data, entity2.sym_data)
        else:
            s_sym = 0.0
        # Fusion linéaire des deux scores
        return self.alpha * s_sub + self.beta * s_sym

#####
# 4. SCNCORE pour le SCN Hybride
#####

```

```

class SCNCore:
    """
    SCNCore gère la matrice des pondérations  $\omega$  et la mise à jour selon la règle DSL hybride.
    """

    def __init__(self, entities: List[Entity], eta: float = 0.1, tau: float = 0.2):
        self.entities = entities
        self.N = len(entities)
        self.eta = eta
        self.tau = tau
        # Initialisation de  $\omega$  avec de petites valeurs aléatoires (symétrique)
        self.omega = np.random.uniform(0.01, 0.05, size=(self.N, self.N))
        self.omega = (self.omega + self.omega.T) / 2.0
        # Instanciation du module de synergie hybride
        self.hybrid_module = ModuleSynergyHyb(alpha=0.7, beta=0.3)

    def compute_synergy(self, i: int, j: int) -> float:
        return self.hybrid_module.compute_hybrid_similarity(self.entities[i], self.entities[j])

    def update_weights(self):
        """
        Met à jour  $\omega$  selon la règle DSL hybride.
        """

        new_omega = self.omega.copy()
        for i in range(self.N):
            for j in range(i + 1, self.N):
                S_ij = self.compute_synergy(i, j)
                delta = self.eta * (S_ij - self.tau * self.omega[i, j])
                new_val = self.omega[i, j] + delta
                # Clamping à 1.0
                new_val = min(new_val, 1.0)
                new_omega[i, j] = new_val
                new_omega[j, i] = new_val
        self.omega = new_omega

    def run_iterations(self, iterations: int = 10):
        for t in range(iterations):
            self.update_weights()
            print(f"Itération {t+1} : moyenne de  $\omega$  = {np.mean(self.omega):.4f}")

#####
# 5. Mini-dataset et Exemple d'Utilisation
#####

def main():
    # Création d'un mini-dataset de 10 entités hybrides
    # Entités 1 à 5 : sub-symboliques (vecteurs de dimension 4)
    entities = [
        Entity("E1", sub_data=[0.2, 0.8, 0.1, 0.1]),
        Entity("E2", sub_data=[0.1, 0.75, 0.05, 0.3]),
        Entity("E3", sub_data=[0.6, 0.2, 0.05, 0.15]),
        Entity("E4", sub_data=[0.3, 0.6, 0.2, 0.1]),
        Entity("E5", sub_data=[0.25, 0.65, 0.15, 0.2]),
        # Entités 6 à 10 : symboliques (ensembles de règles)
        Entity("E6", sym_data=["A->B", "B&C->D"]),
        Entity("E7", sym_data=["X->Y", "¬(Z&Y)"]),
        Entity("E8", sym_data=["P->Q"]),

```

```

Entity("E9", sym_data=["A->B", "B->C"]),
Entity("E10", sym_data=["M->N", "N&O->P"])
]

# Exemple : certaines entités mixtes pourraient avoir les deux types de données.
# Ici, nous ne créons que des entités purement sub-symboliques ou symboliques pour la simplicité.

# Initialisation du SCNCore pour le mini-dataset hybride
scn_core = SCNCore(entities, eta=0.1, tau=0.2)
print("Matrice ω initiale:")
print(np.round(scn_core.omega, 4))

# Exécute 10 itérations de mise à jour
scn_core.run_iterations(iterations=10)

print("Matrice ω finale:")
print(np.round(scn_core.omega, 4))

if __name__ == "__main__":
    main()

```

Expliations Complémentaires

A. Mini-Dataset Hybride

Le mini-dataset contient 10 entités réparties de la manière suivante :

- Les entités **sub-symboliques** (E1 à E5) sont définies par des vecteurs en \mathbb{R}^4 . Ces vecteurs pourraient provenir d'un modèle d'embedding ou d'un descripteur d'image/audio.
- Les entités **symboliques** (E6 à E10) sont définies par des listes de chaînes représentant des règles logiques ou des axiomes. Dans un contexte réel, ces données pourraient être extraites d'une ontologie ou d'un système de raisonnement logique.

B. Calcul de la Synergie Multiple

Le module **ModuleSynergyHyb** combine le score de similarité sub-symbolique, calculé par la similarité cosinus, et le score symbolique, évalué ici par un simple ratio de règles communes. Le score hybride est obtenu via une combinaison linéaire pondérée par α et β . Par exemple, avec $\alpha = 0.7$ et $\beta = 0.3$, si deux entités possèdent un score subsymbolique de 0.8 et un score symbolique de 1.0, le score hybride sera :

$$S_{\text{hyb}} = 0.7 \times 0.8 + 0.3 \times 1.0 = 0.56 + 0.3 = 0.86.$$

Ce score est ensuite utilisé dans la mise à jour des pondérations.

C. Règle de Mise à Jour DSL Hybride

La mise à jour de $\omega_{i,j}$ se fait toujours selon la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hyb}}(i,j) - \tau \omega_{i,j}(t)],$$

ce qui garantit que les liens se renforcent lorsque la synergie est élevée et se décroissent lorsque le score est faible, tout en maintenant une dynamique stable.

D. Observation des Résultats

En exécutant le script, vous pourrez observer la matrice ω évoluer au fil des itérations. On s'attend, par exemple, à voir que les entités sub-symboliques similaires (ex. E1 et E2) voient leur lien se renforcer, tandis que les entités symboliques identiques (E6 et E9 pourraient présenter une forte similarité si leurs règles se recoupent) voient également leur connexion s'accroître. Par ailleurs, des liens entre des entités de modalités différentes (par exemple, entre E2 et une entité symbolique) recevront un score fusionné qui, même s'il est modeste, pourra créer un pont hybride entre les deux mondes.

Conclusion

Le cas d'usage présenté dans cette section illustre de manière concrète la gestion des entités hybrides dans un SCN. À partir d'un mini-dataset composé d'entités sub-symboliques (vecteurs) et symboliques (règles logiques), le SCN combine plusieurs fonctions de synergie pour calculer un score global $S(i,j)$. Ce score hybride, intégré dans la règle de mise à jour DSL, permet au réseau de renforcer les liens entre entités similaires sur plusieurs dimensions simultanément. L'implémentation en Python montre comment structurer ce processus à l'aide de modules dédiés – le module de synergie hybride, le SCNCORE et la règle DSL – afin d'assurer une auto-organisation cohérente et robuste dans un contexte multimodal.

5.9.3.1. Rappeler la Distribution Possible, la Synchro Inter-Robots

Dans les systèmes robotiques ou multi-agents, un SCN (Synergistic Connection Network) permet de modéliser et de renforcer la coopération entre chaque robot ou agent à travers des liaisons pondérées $\omega_{i,j}$. Ces pondérations reflètent la capacité de deux entités à collaborer et s'auto-organiser en fonction de leurs états locaux, lesquels intègrent des informations telles que la position, les capteurs, et même des objectifs spécifiques. Dans ce contexte, la distribution des $\omega_{i,j}$ au sein du réseau et la synchronisation inter-robots sont des éléments essentiels pour assurer une cohérence globale, même dans des environnements dynamiques et distribués.

Nous détaillons ci-après les principaux aspects de cette problématique.

A. Pourquoi un SCN Distribué pour la Robotique ou le Multi-Agent ?

1. Multiplicité et Autonomie des Robots

Dans un essaim composé de dizaines voire de centaines de robots, chacun possède un état local caractérisé par sa position, son orientation, ses mesures capteurs et ses objectifs propres. Mathématiquement, on considère l'ensemble des robots comme un ensemble d'entités $\mathcal{E} = \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ et chaque lien $\omega_{i,j}$ représente la force de coopération entre l'agent i et l'agent j . La synergie $S(i,j)$ peut être définie, par exemple, en fonction de la proximité géographique ou de la complémentarité dans la tâche (ex. couverture de zone, échange d'informations), et la mise à jour des $\omega_{i,j}$ via une règle DSL permet de renforcer les connexions efficaces et de diminuer celles qui sont moins pertinentes.

2. Décentralisation et Scalabilité

Pour un essaim de grande taille, il est impraticable de disposer d'un serveur central collectant toutes les informations et calculant globalement l'ensemble des $\omega_{i,j}$. Ainsi, dans un SCN distribué, chaque robot gère localement une portion des liens, souvent ceux qui concernent ses voisins immédiats. La synchronisation inter-robots s'effectue alors par échanges périodiques de mises à jour ou de résumés d'état. Du point de vue mathématique, on peut modéliser le système comme un ensemble de sous-systèmes autonomes qui communiquent via des opérateurs de synchronisation, de manière à obtenir un équilibre global malgré des retards et une mise à jour asynchrone.

3. Adaptation à un Environnement Dynamique

Les robots évoluent dans des environnements changeants et leur état se modifie en continu. Un SCN distribué doit donc permettre une mise à jour dynamique et continue des pondérations ω . Le fait que chaque robot ne connaisse qu'une partie locale du réseau, avec des échanges inter-robots ponctuels, rend le système non-autonome. Cela signifie

que la fonction de mise à jour F appliquée aux ω dépend de variables externes et temporelles, rendant la convergence vers un état fixe moins réaliste qu'une adaptation continue aux changements. L'important est alors d'assurer que les communications et la synchronisation permettent de maintenir une cohérence globale acceptable, même si la matrice ω évolue en permanence.

B. Synchronisation Inter-Robots et Paramètres de Mise à Jour

1. Protocole de Communication et Synchronisation

Dans un environnement distribué, la synchronisation inter-robots repose sur un protocole de communication qui permet d'échanger les mises à jour des pondérations ou des informations résumées sur les états locaux. Typiquement, chaque robot diffuse périodiquement des messages contenant ses mises à jour $\Delta\omega_{i,j}$ aux agents voisins ou à un orchestrateur local. Ces messages peuvent être envoyés de manière asynchrone, ce qui induit des délais et des déphasages dans la synchronisation. Sur le plan mathématique, on peut considérer que la mise à jour locale d'un robot i est donnée par un opérateur F_i et que la synchronisation globale se fait par une agrégation de ces opérateurs, par exemple :

$$\omega_{i,j}(t+1) = F_i(\omega_{i,j}(t)) + \Delta_{comm}(t),$$

où $\Delta_{comm}(t)$ représente la correction apportée par les échanges inter-robots pour harmoniser les valeurs de $\omega_{i,j}$ dans l'ensemble du réseau. Des mécanismes de consensus (comme ceux inspirés des algorithmes de Paxos ou de Raft) peuvent être utilisés pour minimiser les incohérences dues aux retards.

2. Paramètres de Mise à Jour : η et τ

La règle de mise à jour classique est donnée par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Ici :

- η est le **taux d'apprentissage** qui détermine la réactivité du système aux changements de synergie. Un η élevé rend le système très sensible aux variations, mais peut également conduire à des oscillations, notamment en présence de retards de synchronisation.
- τ est le **coefficent de décroissance** qui sert de régulateur pour éviter une croissance excessive des pondérations. Dans un environnement robotique dynamique, un choix judicieux de τ permet de modérer les variations en fonction des perturbations externes (par exemple, des changements rapides de position).

Le réglage de ces paramètres doit prendre en compte le niveau de bruit dans les communications et les délais de synchronisation. Une réactivité trop forte (valeur élevée de η) peut compromettre la stabilité en raison des communications asynchrones entre robots.

3. Convergence et Dynamique Continue

Contrairement à des systèmes statiques qui cherchent une convergence vers un état fixe, les systèmes robotiques fonctionnent dans des environnements en perpétuel changement. Ainsi, la matrice ω est amenée à évoluer continuellement. Le protocole de synchronisation doit alors assurer que, malgré des retards et des mises à jour locales asynchrones, l'ensemble du SCN conserve une **cohérence globale** suffisante pour permettre la formation de clusters opérationnels. Par exemple, dans un essaim de drones surveillant une zone, même si les valeurs de $\omega_{i,j}$ changent en fonction des mouvements et des échanges, la synchronisation assure que les drones restent regroupés en sous-équipes effectuant des missions spécifiques.

C. Impact sur la Coordination et la Répartition des Tâches

L'architecture distribuée d'un SCN offre plusieurs avantages pour la coordination inter-robots :

- **Formation de Groupes** : Les robots dont les synergies sont élevées se regroupent, formant des clusters qui correspondent à des équipes collaborant sur des tâches communes (par exemple, surveillance d'une zone, cartographie, ou transport d'objets).
- **Adaptation Dynamique** : En fonction des changements d'état (position, orientation, capteurs) et des échanges inter-robots, le réseau s'adapte en temps réel. Un robot peut changer de cluster s'il se rapproche d'un autre groupe, ce qui permet une répartition flexible des missions.
- **Robustesse Distribuée** : Chaque robot ne gère qu'une partie locale des informations et les synchronise avec ses voisins. Cela rend le système moins vulnérable à une panne centralisée et facilite la redondance et la reprise après une défaillance.

Sur le plan **mathématique**, on modélise le système comme un ensemble d'équations différentielles discrètes couplées, où chaque robot applique localement une mise à jour selon la règle DSL, et où les corrections de synchronisation agissent comme des perturbations qui tendent à harmoniser l'ensemble du réseau. Cette approche permet de garantir, malgré l'asynchronie et les délais de communication, que le SCN atteint un état de coordination fonctionnelle.

Conclusion

L'application d'un SCN distribué dans le domaine de la robotique ou des systèmes multi-agents repose sur deux axes essentiels : d'une part, la capacité de chaque robot à gérer localement ses liaisons $\omega_{i,j}$ via un état propre et des échanges périodiques avec ses voisins ; d'autre part, l'existence d'un protocole de synchronisation inter-robots qui permet de consolider ces informations de manière asynchrone tout en maintenant une cohérence globale.

Du point de vue **mathématique**, la mise à jour de la matrice ω se fait via des opérateurs locaux F_i couplés par des termes de synchronisation correctifs, garantissant ainsi un équilibre dynamique dans un environnement non-stationnaire. Du point de vue **ingénierie**, cette architecture distribuée assure que chaque robot peut adapter sa participation à l'auto-organisation en temps réel, facilitant la formation spontanée de clusters et la répartition efficace des tâches dans un essaim.

En résumé, la distribution possible des pondérations et la synchronisation inter-robots constituent des mécanismes clés pour que l'auto-organisation d'un SCN soit à la fois robuste, flexible et adaptée aux environnements dynamiques, garantissant ainsi une coordination efficace et distribuée au sein d'un essaim robotique ou d'un système multi-agent.

5.9.3.2. Visualisation d'un SCN en Essaim Robotique

I. Analyse et Motivations de la Visualisation d'un SCN en Essaim Robotique

A. Objectifs et Bénéfices

Dans un environnement multi-agent, la matrice des pondérations ω issue du SCN encode l'intensité des interactions entre robots. Toutefois, lorsqu'on manipule un grand nombre de robots, il devient difficile d'extraire intuitivement la structure du réseau à partir de simples tableaux ou listes de valeurs numériques. La visualisation permet alors de :

291. Comprendre le comportement collectif

Une représentation graphique (par exemple, un graphe en 2D) permet de détecter visuellement la formation de clusters ou de groupes coopératifs. Chaque robot est représenté par un nœud, et les arêtes qui relient deux robots sont colorées ou dont l'épaisseur varie en fonction de la valeur de $\omega_{i,j}$. Ainsi, des arêtes épaisses et de

couleurs vives indiquent une forte coopération, tandis que des arêtes fines ou absentes traduisent une faible synergie.

292. Diagnostiquer et Ajuster les Paramètres

En visualisant l'évolution des liens, un opérateur peut détecter des anomalies (par exemple, des liens excessivement forts ou faibles) qui pourraient résulter d'un mauvais paramétrage ou d'une défaillance du système. Ceci aide à régler les paramètres de la dynamique d'auto-organisation, tels que le taux d'apprentissage η et le coefficient de décroissance τ .

293. Suivre l'Adaptation Dynamique

Dans un environnement robotique dynamique, la disposition des robots change en continu. La visualisation permet de suivre en temps réel comment les robots se regroupent, se séparent ou se réorganisent en fonction de leur position et des échanges de données (par exemple, les retours capteurs ou les informations de mission).

B. Approches de Visualisation

Plusieurs méthodes de représentation graphique sont envisageables :

294. Graphe en 2D ou 3D

On positionne chaque robot en fonction de ses coordonnées physiques (par exemple, sur un plan réel) et on trace des arêtes entre les robots. La couleur et l'épaisseur des arêtes sont déterminées par la valeur de $\omega_{i,j}$. Un *layout force-directed* peut être utilisé pour obtenir une représentation qui révèle la structure topologique des interactions.

295. Heatmap ou Matrice de Pondérations

Pour un essaim de taille modérée, une heatmap de la matrice ω fournit une vision globale des niveaux d'interaction. Les valeurs élevées apparaissent en couleurs vives, ce qui facilite l'identification des clusters internes.

296. Vue de Clusters

Un algorithme de détection de communautés (ex. Louvain) peut être appliqué pour extraire les groupes d'agents fortement connectés. La visualisation se fait alors par des couleurs ou des labels indiquant à quel cluster appartient chaque robot.

C. Exemple d'Application : Essaim d'une Douzaine de Robots

Considérons un essaim de 12 robots terrestres évoluant dans une zone. Chaque robot, assimilé à une entité \mathcal{E}_i , possède des liaisons $\omega_{i,j}$ avec les autres robots, qui évoluent selon la dynamique d'auto-organisation décrite par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Pour faciliter la compréhension du comportement collectif, on peut visualiser le réseau de robots à intervalles réguliers. Ainsi, sur une carte 2D, chaque robot est représenté par un nœud (placé selon sa position réelle ou simulée), et les arêtes entre les robots sont dessinées avec une épaisseur ou une couleur proportionnelle à la valeur de $\omega_{i,j}$. Cette représentation permet de voir par exemple :

- Des clusters où des robots se regroupent pour effectuer une mission commune (fortes interactions),
- Des robots isolés ou en dérive, dont les liens sont faibles,
- La répartition spatiale des sous-groupes et l'évolution de la coordination dans le temps.

II. Implémentation Python pour la Visualisation d'un Essaim de 12 Robots

Nous proposons ci-après un programme Python complet qui simule un essaim de 12 robots. Le programme crée des positions aléatoires pour chaque robot sur un plan, génère une matrice de pondérations ω simulant des interactions (par exemple, basée sur la distance entre robots), et affiche le graphe du réseau en utilisant la bibliothèque **NetworkX** et **Matplotlib**.

Le programme intègre les étapes suivantes :

- Génération des positions des robots,
- Calcul d'une matrice de pondérations simulée, où une valeur élevée de $\omega_{i,j}$ correspond à des robots proches ou en coopération,
- Visualisation du graphe avec des arêtes dont l'épaisseur et la couleur reflètent la force $\omega_{i,j}$.

```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
from typing import Any, List, Tuple # Importation de Tuple et d'autres types

# Pour assurer la reproductibilité
np.random.seed(42)

#####
# 1. Génération des Positions des Robots
#####

def generate_robot_positions(num_robots: int, area_size: Tuple[float, float] = (100, 100)) -> np.ndarray:
    """
    Génère des positions aléatoires pour num_robots dans une zone de dimensions area_size.
    Renvoie un tableau de forme (num_robots, 2).
    """
    x_positions = np.random.uniform(0, area_size[0], num_robots)
    y_positions = np.random.uniform(0, area_size[1], num_robots)
    return np.column_stack((x_positions, y_positions))

# 2. Calcul de la Matrice de Pondérations
#####

def compute_weight_matrix(positions: np.ndarray, sigma: float = 20.0) -> np.ndarray:
    """
    Calcule une matrice de pondérations  $\omega$  basée sur une fonction gaussienne de la distance.
    Plus deux robots sont proches, plus  $\omega$  est élevé.
    La fonction utilisée est :  $\omega(i,j) = \exp(-d(i,j)^2 / (2 * \sigma^2))$ .
    """
    num_robots = positions.shape[0]
    omega = np.zeros((num_robots, num_robots))
    for i in range(num_robots):
        for j in range(i + 1, num_robots):
            distance = np.linalg.norm(positions[i] - positions[j])
            weight = np.exp(-(distance ** 2) / (2 * sigma ** 2))
            omega[i, j] = weight
            omega[j, i] = weight # symétrie
```

```

return omega

#####
# 3. Visualisation du SCN
#####

def visualize_scn(positions: np.ndarray, omega: np.ndarray, threshold: float = 0.2):
    """
    Visualise le SCN sous forme de graphe où les nœuds sont les positions des robots.
    Seules les arêtes dont la pondération est supérieure au seuil sont affichées.
    L'épaisseur et la couleur des arêtes varient en fonction de la valeur de  $\omega$ .
    """

    num_robots = positions.shape[0]
    G = nx.Graph()
    # Ajout des nœuds avec leur position
    for i in range(num_robots):
        G.add_node(i, pos=(positions[i, 0], positions[i, 1]))

    # Ajout des arêtes avec les poids
    for i in range(num_robots):
        for j in range(i + 1, num_robots):
            weight = omega[i, j]
            if weight > threshold: # Seuil pour afficher une liaison
                G.add_edge(i, j, weight=weight)

    pos = nx.get_node_attributes(G, 'pos')
    weights = [G[u][v]['weight'] for u, v in G.edges()]

    # Normaliser les poids pour l'épaisseur des arêtes
    max_weight = max(weights) if weights else 1.0
    widths = [4 * (w / max_weight) for w in weights]

    # Choix d'une palette de couleurs (ici, on utilise la colormap viridis)
    edge_colors = [plt.cm.viridis(w) for w in weights]

    plt.figure(figsize=(10, 8))
    nx.draw_networkx_nodes(G, pos, node_size=300, node_color='lightblue')
    nx.draw_networkx_labels(G, pos)
    nx.draw_networkx_edges(G, pos, width=widths, edge_color=edge_colors)
    plt.title("Visualisation d'un SCN en Essaim Robotique")
    plt.axis('off')
    plt.show()

#####
# 4. Programme Principal
#####

def main():
    num_robots = 12
    positions = generate_robot_positions(num_robots, area_size=(100, 100))
    omega = compute_weight_matrix(positions, sigma=20.0)

    # Affichage des positions
    plt.figure(figsize=(8, 6))
    plt.scatter(positions[:, 0], positions[:, 1], c='blue', s=100)
    for i, (x, y) in enumerate(positions):

```

```

plt.text(x, y, f"i", fontsize=12, color='white', ha='center', va='center')
plt.title("Positions des robots dans l'essaim")
plt.xlabel("X")
plt.ylabel("Y")
plt.show()

# Visualisation du SCN sous forme de graphe
visualize_scn(positions, omega, threshold=0.2)

if __name__ == "__main__":
    main()

```

Explinations

297. Importation des Types

Nous avons ajouté *from typing import Tuple* (ainsi que d'autres types utiles) pour pouvoir utiliser le type *Tuple[float, float]* dans la fonction *generate_robot_positions*.

298. Génération des Positions

La fonction *generate_robot_positions* crée un tableau de positions aléatoires pour un nombre donné de robots dans une zone définie par *area_size*.

299. Calcul de la Matrice de Pondérations

La fonction *compute_weight_matrix* calcule une matrice ω en appliquant une fonction gaussienne sur la distance entre chaque paire de robots. Plus deux robots sont proches, plus la valeur retournée sera proche de 1.

300. Visualisation du SCN

La fonction *visualize_scn* construit un graphe avec NetworkX où chaque robot est un nœud, et les arêtes sont tracées uniquement pour des liaisons dont le poids dépasse un seuil donné. L'épaisseur et la couleur des arêtes sont proportionnelles à la valeur des pondérations, ce qui permet de visualiser la force des connexions.

301. Programme Principal

La fonction *main()* orchestre l'ensemble du processus :

- Génération des positions pour 12 robots,
- Calcul de la matrice ω ,
- Affichage des positions sur un scatter plot,
- Visualisation du graphe du SCN.

Conclusion

Ce cas d'usage illustre comment un SCN appliqué à un essaim robotique permet non seulement de modéliser les interactions et la coopération entre les robots, mais aussi de visualiser en temps réel ou par snapshots la structure émergente du réseau. La visualisation – réalisée ici via un graphe 2D où l'épaisseur et la couleur des arêtes reflètent la force des liaisons $\omega_{i,j}$ – fournit un outil puissant de diagnostic et de suivi de la dynamique d'auto-organisation, essentiel

pour le débogage, l'ajustement des paramètres, et la gestion des missions dans des environnements robotiques complexes.

Le programme Python proposé offre une simulation simple d'un essaim de 12 robots et permet d'observer comment les liaisons se répartissent et se renforcent en fonction des distances. Cette approche peut être étendue à des systèmes de plus grande envergure ou intégrée dans des outils de supervision pour des déploiements réels de systèmes multi-agents ou robotiques distribués.

5.10. Conclusion et Ouverture

5.10.1. Bilan du Chapitre

- Récapituler la mise en place d'une **architecture SCN** :
 1. Structures de données (matrices, adjacency),
 2. Modules (synergie, mise à jour, inhibition),
 3. Interfaces (ajout d'entités, extraction de clusters),
 4. Distribution, sécurité, etc.

5.10.2. Lien vers Chapitres 6, 7, 8

- Chap. 6 : Apprentissage Synergique Multi-Échelle (exploitation de cette architecture pour gérer différents niveaux).
- Chap. 7 : Algorithmes d'Optimisation et Méthodes d'Adaptation Dynamique (performances, recuit plus avancé).
- Chap. 8 : DSL Multimodal (comment la structure SCN se déploie à grande échelle avec divers flux).

5.10.3. Conclusion Générale

- Souligner que ce **Chapitre 5** fait le lien entre la **dynamique** (Ch. 4) et les **approches plus spécialisées** (Ch. 6, 7, 8) :
 - On dispose désormais de la \textbf{colonne vertébrale} pour implanter un SCN, en modulaire ou en distribué, assurant robustesse et évolutivité.

5.10. Conclusion et Ouverture

Après avoir exploré la **dynamique** d'un SCN (chapitre 4) et esquissé divers **exemples** de mise en œuvre dans des cas multimodaux, hybrides et distribués, ce chapitre 5 s'est consacré à la **structure générale** d'un SCN sur le plan « architecture et ingénierie ». Nous avons vu comment l'on passe du schéma mathématique $\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j), \dots)$ à une organisation modulaire — avec des **modules** dédiés (calcul de la synergie, mise à jour, inhibition, etc.) —, des interfaces temps réel (ajout/suppression d'entités, extraction de clusters), et une approche distribuée (sharding, synchronisation, robustesse) garantissant la **scalabilité** et la **fiabilité** du système.

5.10.1. Bilan du Chapitre

Le présent **chapitre 5** a exposé les fondements **techniques** et **architecturaux** nécessaires à l'**implémentation** d'un **Synergistic Connection Network (SCN)** à partir des principes **mathématiques** vus antérieurement. L'ensemble des **équations** formant la dynamique de mise à jour $\omega(t+1) = F(\omega(t))$ a ainsi trouvé sa transposition dans des structures de données appropriées, des “modules” logiques, et des mécanismes de distribution et de sécurité. Il convient de souligner plusieurs avancées majeures qui ont jalonné ce chapitre :

Dans un premier temps, la notion de **structure de données** a été clarifiée. Suivant la **taille** du SCN et la **parsimonie** attendue (c'est-à-dire la proportion de liens réellement actifs), on opte pour une **matrice dense** $\{\omega_{i,j}\}$ ou une **représentation sparsed** : listes d'adjacence, dictionnaires (hashmaps), etc. D'un point de vue **mathématique**, cela revient à discriminer quelles paires (i,j) ont un $\omega_{i,j}$ pertinent, et à gérer la “portion active” de ω . Dans un SCN de grande dimension, les mécanismes de “top-k” ou de filtrage ϵ -radius (cf. Section 5.4.3) encouragent la parcimonie, rendant l'implémentation scalable.

Ensuite, le concept de **module** s'est imposé comme un moyen de séparer la **logique** du calcul de la **synergie** $S(i,j)$ (Section 5.4) de la logique du **noyau** (ou *core*) qui pilote la **mise à jour** de ω . On a souligné que le **Module Synergie** pouvait décliner plusieurs formules : par exemple, une fonction S_{sub} pour des entités sub-symboliques (vecteurs), une fonction S_{sym} pour des entités symboliques, etc. Parallèlement, le **Module Mise à Jour** (Section 5.5) orchestre la **règle DSL** (additive, multiplicatif, etc.), ainsi que divers compléments (inhibition compétitive, saturation) en s'appuyant sur les équations décrites au chapitre 4. L'organisation en “modules” favorise la **clarté** et la **modularité** : on introduit ou modifie une règle d'inhibition sans toucher au code de similarité, et inversement.

La question des **interfaces** (Section 5.6) a ensuite été abordée. On s'est penché sur les moyens d'**ajouter** ou de **retirer** des entités E_i en temps réel, provoquant une reconfiguration de la matrice ω . Cette flexibilité s'avère cruciale dans des environnements où l'ensemble des entités n'est pas figé (flux de données, scénarios robotiques, bases évolutives). De plus, il a été question des méthodes pour **extraire** les clusters ou les liens forts, afin d'alimenter des modules externes de visualisation ou de classification. Ces interfaces s'intègrent parfaitement à la logique d'**auto-organisation** du SCN, tout en permettant un usage “ouvert” vers d'autres systèmes.

Enfin, le chapitre a insisté sur la **distribution** (Section 5.7) et la **sécurité** (Section 5.8). La distribution, qu'elle se concrétise par des **sous-SCN** interconnectés ou par un paradigme de **microservices**, vise à gérer des SCN de très grande échelle, tout en évitant un point central unique. Sur le plan **mathématique**, cela correspond à décomposer la matrice ω en shards, et à déployer un protocole de **synchronisation** (asynchrone ou synchrone) pour conserver la **cohérence** du réseau. D'un point de vue **ingénierie**, des solutions de **sharding**, d'équilibrage de charge, et de **méta-SCN** (Section 5.7.2) concrétisent ce modèle. Concernant la **sécurité**, on a introduit des méthodes de **logs**, de **watchers**, et de **checks d'anomalies** (cf. 5.8.1, 5.8.3), ainsi que des techniques de **basculement automatique** et de **redondance** (5.8.2) pour qu'un SCN distribué ne cède pas face à une attaque ou à une panne matérielle. Le chapitre a ainsi présenté la “colonne vertébrale” d'une architecture résiliente, apte à faire tourner un SCN en conditions réelles ou hostiles.

En synthèse, ce **chapitre 5** constitue un “**canevas**” technique pour **réaliser** un SCN à partir d'un socle **mathématique** (mise à jour $\omega(t+1) = F(\omega(t))$, synergie S). Son propos s'est montré large : il a exposé les structures de données, l'organisation modulaire (Module Synergie, Module Mise à Jour), l'ouverture d'interfaces (ajout/retrait d'entités, extraction de clusters), la gestion distribuée et sécurisée. Ce sont autant d'éléments indispensables pour que la **dynamique d'auto-organisation** décrite dans les chapitres précédents devienne un **système** informatique fiable et évolutif, qu'il s'agisse d'un SCN multimodal, robotique ou d'un usage encore différent.

5.10.2. Lien vers Chapitres 6, 7, 8

Au terme de ce **chapitre 5**, où l'on a soigneusement défini la **colonne vertébrale** d'un SCN (Synergistic Connection Network) du point de vue **architectural** et **implémentationnel**, il est naturel de se pencher sur la manière dont cette infrastructure va s'articuler avec les développements plus **avancés** des chapitres ultérieurs. Les **modules**, le **partage** ou la **distribution**, la **sécurité**, et l'ensemble des mécanismes présentés servent de **fondation** à des thématiques plus spécialisées, telles que la **multi-échelle** (Chapitre 6), les **optimisations** et **adaptations** (Chapitre 7), ou encore l'**expansion** du DSL à de vastes environnements **multimodaux** (Chapitre 8).

Chap. 6 : Apprentissage Synergique Multi-Échelle

Le **Chapitre 6** aborde la gestion d'un SCN fonctionnant à **plusieurs niveaux** ou **plusieurs échelles** d'entités et de synergies. L'idée consiste à permettre, par exemple, qu'un SCN se décompose simultanément en **micro-clusters** très fins et en **macro-structures** plus globales, évoquant différents degrés de granularité ou d'abstraction. D'un point de vue **mathématique**, on retrouve la notion de répéter la logique d'**auto-organisation** (règle DSL, inhibition, saturation) à plusieurs *couches*, ou de passer par un "méta-niveau" supervisant l'assemblage des clusters locaux. D'un point de vue **architecture** (développée dans ce chapitre 5), on mobilise la séparation en **modules** (synergie, mise à jour) et la **distribution** (Section 5.7), pour distinguer différentes **couches** ou **blocs**. Le **multi-échelle** enrichit la capacité d'**auto-organisation** : le SCN n'agit plus seulement à un plan unique, mais adopte un "zoom" local et global, ce qui implique une orchestration subtile. Le **Chapitre 6** prolonge donc l'**infrastructure** décrite ici, en l'appliquant à une situation où plusieurs **niveaux** de représentation s'interconnectent.

Chap. 7 : Algorithmes d'Optimisation et Méthodes d'Adaptation Dynamique

Le **Chapitre 7** explore des **mécanismes** plus avancés pour améliorer la convergence, la stabilité ou le temps de réponse d'un SCN. S'il est vrai que les **règles** DSL de base (additive, multiplicative, etc.) constituent un socle déjà efficace, on peut vouloir intégrer des techniques plus élaborées (recuit simulé raffiné, heuristiques inspirées de la physique statistique, descentes adaptatives, etc.) pour atteindre un **minimum** d'énergie plus global ou pour éviter certaines bifurcations indésirables. Or, l'**architecture** (Chap. 5) décrivant la séparation "**Module Mise à Jour**" (Section 5.5) se prête idéalement à ces substitutions ou compléments : on peut injecter un algorithme d'optimisation "avancé" dans la boucle de mise à jour, sans toucher à la structure sous-jacente (modules de synergie, distribution). Le **Chapitre 7** mettra donc l'accent sur la notion de **performance** et de **résilience** algorithmique d'un SCN, tirant profit de l'organisation modulaire et des protocoles distribués détaillés ici.

Chap. 8 : DSL Multimodal

Si l'on veut aller plus loin dans la **multimodalité** — c'est-à-dire manipuler, à *grande échelle*, plusieurs types de données (images, texte, audio, signaux divers) —, le **Chapitre 8** se penchera sur les méthodes requises pour gérer un SCN imposant, agrémentant des entités de natures variées. Les concepts d'**architecture** (Section 5.9) et de **distribution** (Section 5.7) prennent une importance cruciale : lorsque le nombre d'entités et de flux multimodaux grandit, il devient indispensable de recourir aux solutions de **sharding** ($O(n^2)$ liens), de **synchronisation** inter-blocs, et de **sécurisation** (Section 5.8) présentées en détail dans ce chapitre. Le **Chapitre 8** focalisera sur la **logique** DSL appliquée à des sources massives (vision, langage, son), mobilisant la modularité (ModuleSynergy pour chaque modalité) et l'éventuel recuit (5.5.4) pour une **auto-organisation** multimodale puissante.

Conclusion sur le Lien avec Chapitres 6, 7, 8

Le **Chapitre 5** a défini l'**ossature** : structures de données, organisation en "modules" (synergie, mise à jour), interfaces pour la distribution et la sécurité. Ce **canevas** permet de **concrétiser** la dynamique $\omega(t) \rightarrow \omega(t + 1)$ dans des cas d'usages complexes. Les **chapitres 6, 7, 8** profitent directement de cette infrastructure :

302. Le **Chap. 6** poussera l'idée de **multi-échelle**, montrant comment la modularité et les protocoles distribués soutiennent une auto-organisation hiérarchique.

303. Le **Chap. 7** traitera des **algorithmes d'optimisation** avancés, trouvant dans la structure modulaire (Section 5.5) un champ favorable pour injecter de nouvelles heuristiques.

304. Le **Chap. 8** explorera la **multimodalité** à vaste échelle, s'appuyant sur les outils de distribution (5.7) et de robustesse (5.8) pour orchestrer un SCN en présence de multiples flux (images, textes, signaux audio, etc.).

Au total, ce passage vers les **chapitres 6, 7, 8** montre comment l'**architecture** établie dans ce chapitre 5 sert de **fondation** pratique et conceptuelle, permettant d'**élargir** la portée du DSL vers des dispositifs multi-niveaux, des algorithmes plus fins, et des environnements massivement multimédias ou hétérogènes.

5.10.3. Conclusion Générale

Le **chapitre 5** établit la véritable **charnière** entre la formulation mathématique de la **dynamique** d'un SCN et les usages avancés (multi-échelle, optimisations, multimodalité) que l'on retrouvera dans les prochains chapitres. Jusque-là, le **chapitre 4** avait détaillé la mise à jour $\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j))$ et ses variations (additive, multiplicative, etc.). Toutefois, ces principes demeurent purement **théoriques** tant qu'ils ne sont pas insérés dans une **architecture** capable d'héberger la matrice ω , de gérer le calcul du score de synergie, et d'assurer la gestion distribuée ou la robustesse nécessaire à un usage concret.

Ce **chapitre 5** a précisément comblé cet écart, en introduisant un **canevas** technique, du niveau des **structures de données** jusqu'aux mécanismes de **sécurité**.

Dans un premier mouvement, on a rappelé que la **taille** du SCN (et la nature plus ou moins parcimonieuse de ses liaisons) détermine le choix des **structures** : matrice dense $\{\omega_{i,j}\}$ ou schémas plus **spars** (listes d'adjacence, hashmap, etc.). D'un point de vue **mathématique**, cela revient à définir plus précisément le "domaine" de ω : quelles paires (i,j) se retrouvent effectives dans la boucle, et comment on indexe ou on stocke la partie active.

On a ensuite consolidé l'idée de séparer un **Module de Synergie** (Section 5.4), qui encapsule le calcul $S(i,j)$ — éventuellement différent selon les modalités sub-symboliques ou symboliques —, et un **Module Mise à Jour** (Section 5.5), chargé d'appliquer la règle DSL, d'incorporer l'inhibition, la saturation ou le recuit. Cette **modularité** se révèle décisive pour l'extensibilité : on peut changer l'algorithme de synergie ou expérimenter une nouvelle règle d'évolution $\omega_{i,j}(t+1)$ sans déstabiliser tout le code.

Le chapitre a aussi présenté l'existence d'**interfaces** (5.6) permettant d'**ajouter** ou de **supprimer** des entités \mathcal{E}_i en temps réel, de **consulter** des clusters ou des liens dominants, etc. Sur le plan **ingénierie**, cela confère un **visage** plus "ouvert" au SCN : on peut l'intégrer à d'autres modules, l'utiliser pour la visualisation ou l'analyse d'un graphe, ou procéder à des modifications dynamiques (scénario en flux continu).

Puis on a traité la **distribution** (Section 5.7), moment-clé pour gérer des SCN de forte dimension ou distribués géographiquement. Les principes de **sharding** (couper la matrice ω en fragments), de **communication inter-blocs** (messages synchrones ou asynchrones), et de **synchronisation** (invariants, bornes) assurent une **scalabilité** horizontale et une résilience face au volume massif de liaisons. D'un point de vue **mathématique**, on retrouve un ensemble d'opérateurs locaux couplés, où la mise à jour $\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j))$ doit être cohérente même avec les retards et la répartition topologique.

Enfin, les **aspects de sécurité** (Section 5.8) et de **robustesse** donnent les briques manquantes pour qu'un SCN opère dans des contextes éventuellement hostiles ou soumis à défaillance : détection d'anomalies ($\omega_{i,j}$ qui croît anormalement), logs, watchers, réactivation en cas de panne, duplication (failover, redondance). Le SCN n'est plus simplement un algorithme isolé : il devient un **système** complet, capable de résister aux sabotages et aux coupures, tout en maintenant son auto-organisation.

Lien direct avec la Dynamique (Chap. 4).

L'essentiel des équations de mise à jour, comme $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$, prennent **forme concrète** lorsqu'on sait où stocker ω , comment appeler la fonction $S(i,j)$, et comment manipuler ω en flux continu (ajout/suppression d'entités). Les sections 5.4 (Module Synergie) et 5.5 (Module Mise à Jour) traduisent directement les formulations mathématiques (Chap. 4) en entités logicielles, paramétrables et agencées dans un pipeline.

Porte d'accès aux Chapitres 6, 7 et 8.

Les chapitres qui suivent s'adossent à l'infrastructure présentée ici :

- Le **Chap. 6** traite du “multi-échelle”, c'est-à-dire de SCN capables de se structurer selon plusieurs niveaux (micro-clusters, macro-structures). Il mobilise la répartition (Section 5.7) et la modularité (Sections 5.4, 5.5) pour illustrer l'auto-organisation dans des contextes hiérarchiques ou à échelles imbriquées.
- Le **Chap. 7** introduit des **algorithmes d'optimisation** et des heuristiques d'amélioration (recuit plus poussé, stratégies de minimisation d'énergie, etc.). On y voit comment la “colonne vertébrale” du SCN, déjà paramétrée (Section 5.5, 5.5.4), se prête à l'insertion de techniques plus avancées, tant dans un cadre local que distribué.
- Le **Chap. 8** approfondit la notion de **DSL Multimodal**, où l'on manipule potentiellement des flux massifs (vision, audio, langage). Les principes d'architecture (modules de synergie spécialisés, distribution) et de sécurité (Section 5.8) se révèlent alors indispensables pour supporter la “montée en charge” et la gestion de multiples types d'entités.

Robustesse et Évolutivité.

On retiendra que le **chapitre 5** propose une **colonne vertébrale** : choix de structures (dense, sparse, NoSQL), organisation modulaire (Synergie, Mise à Jour), distribution à grande échelle (sharding, synchronisation), mécanismes de logs, watchers, reprise sur panne. L'ensemble confère à la **dynamique DSL** (Chap. 4) la solidité nécessaire pour s'exécuter dans des conditions **réalistes** et **évolutives**. On peut donc envisager un SCN opéré en cluster, mis à jour en flux, protégé contre des anomalies ou des sabotages, et capable de réassigner ses entités au gré de l'évolution.

Conclusion générale du Chapitre 5 :

On dispose désormais des **bases** d'ingénierie et d'architecture pour qu'un **SCN** devienne un système complet, joignant l'**auto-organisation** interne (mise à jour ω) à une gestion modulaire (synergie, distribution, sécurité) et à la **flexibilité** (ajout/suppression d'entités, extraction de liens). Les **chapitres suivants** (6, 7, 8) puissent dans cette infrastructure pour développer :

- La **multi-échelle** (Chap. 6),
- Les **algorithmes d'optimisation** (Chap. 7),
- Les **extensions multimodales** plus complexes (Chap. 8).

Ce faisant, la **puissance** du Deep Synergy Learning se déployera pleinement, soutenue par un canevas technique et algorithmique solide.

Chapitre 6 : Apprentissage Synergique Multi-Échelle et Structures Fractales

Chapitre 6 : Apprentissage Synergique Multi-Échelle et Structures Fractales	827
6.1. Introduction	Erreur ! Signet non défini.
6.1. Introduction	829
6.1.1. Contexte et Motivation	829
6.1.2. Objectifs du Chapitre	832
6.1.3. Structure du Chapitre	837
6.2. Principes de l'Apprentissage Multi-Échelle	840
6.2. Principes de l'Apprentissage Multi-Échelle	841
6.2.1. Hiérarchies Synergiques et Niveaux d'Abstraction	841
6.2.2. Théorie des Systèmes Emboîtés	845
1. Processus d'Agrégation.....	849
2. Pseudo-code.....	849
6.2.3. Rétroactions et Anticipations Multi-Niveau.....	851
6.3. Fractalité et Auto-Similarité dans le DSL	858
6.3.1. Concept de Fractales en IA	858
6.3.2. Auto-Similarité dans les Réseaux Synergiques.....	863
6.3.3. Modélisation et Indicateurs	870
6.3.4. Avantages et Limitations	876
6.4. Interactions Multi-Niveau et Coordination.....	881
6.4.1. Bottom-Up vs. Top-Down	881
6.4.2. Communication Synergiques entre Niveaux	890
6.4.3. Synchronisation et Clustering	895
6.5. Dynamique et Algorithmes Multi-Échelle.....	902
6.5.1. Agrégation Progressive (Bottom-Up)	902
6.5.2. Division ou Zoom (Top-Down)	907
6.5.3. Hybridation (Approche Mixte).....	913
6.5.4. Algorithmes de Mise en Œuvre	918
6.6. Études de Cas et Illustrations	926
6.6.1. Systèmes de Vision Multi-Modal	926

6.6.2. Analyse Contextuelle de la Parole et du Langage	930
B. Intérêt Mathématique et Pratique	931
6.6.3. Robotique Synergique : Intégration Sensorimotrice.....	935
6.6.4. Agents Conversationnels Riches et Contextuels	940
6.6.5. Applications dans la Simulation et la Prédition d'Événements	945
6.7. Conclusion.....	951
6.7. Conclusion.....	952
6.7.1. Synthèse des Contributions du Chapitre	952
6.7.2. Rôle de la Fractalité	953
6.7.3. Limites et Perspectives	953
6.7.4. Liens avec les Chapitres Suivants	954
6.7.5. Vision Globale	955

Dans ce **chapitre 6**, nous entrons dans une nouvelle dimension du **DSL** (Deep Synergy Learning) : l'**apprentissage multi-échelle**, où les synergies ne se limitent pas à un unique niveau de granularité mais se déclinent à **plusieurs niveaux** (micro, méso, macro, voire intermédiaires). Cette notion introduit également la perspective de **structures fractales**, c'est-à-dire une forme d'**auto-similarité** potentielle à différentes échelles du réseau, qui peut se manifester lorsque les lois d'organisation se répliquent à chaque palier.

6.1. Introduction

Le **chapitre 6** prolonge les fondements et l'architecture posés dans les chapitres précédents (notamment le chapitre 5 sur la mise en place technique d'un SCN) pour montrer comment on peut **gérer et exploiter** la diversité d'échelles dans le DSL. À l'issue de ce chapitre, on comprendra en quoi la multi-échelle ouvre la voie à une **coordination** plus souple entre divers niveaux (local, global...) et comment la fractalité peut modéliser ou expliquer l'émergence de motifs répétitifs au sein du réseau de synergies.

6.1.1. Contexte et Motivation

Le **DSL** (Deep Synergy Learning) repose sur des entités \mathcal{E}_i et leurs liens $\omega_{i,j}$, guidés par la fonction de synergie $S(i,j)$. Jusqu'ici, nous avons souvent envisagé la formation de clusters ou la dynamique d'auto-organisation à une **même échelle** (entités ou sous-ensembles d'entités). Or, dans de nombreux systèmes naturels (cérébraux, écologiques, sociaux) ou artificiels (réseaux complexes, infrastructures distribuées), on observe des **structures à plusieurs niveaux** de résolution :

- Des **micro-clusters** (petits groupes fortement cohérents),
- Des **macro-ensembles** plus vastes,
- Des **niveaux intermédiaires** jouant un rôle de pivot.

De plus, il se peut qu'un **même motif** (par exemple, un cluster en “étoile” ou un schéma cyclique) se **reproduise** à diverses échelles, suggérant une **auto-similarité** ou un comportement dit *fractal*.

6.1.1.1. Rappel : Le DSL (Deep Synergy Learning) fonctionne sur la base de synergies entre entités, mais l'échelle à laquelle on étudie ces synergies peut varier (local, global, intermédiaire)

Le **Deep Synergy Learning (DSL)** s'appuie, dans sa formulation de base, sur une règle d'**auto-organisation** qui met à jour les liaisons $\omega_{i,j}(t)$ en fonction de la synergie $S(i,j)$ évaluée entre les entités \mathcal{E}_i et \mathcal{E}_j . La mise à jour peut prendre la forme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

ou toute autre **règle DSL** (additive, multiplicative, etc.) suivant les paradigmes du chapitre 4. Sur un plan **mathématique**, cette dynamique ne dépend pas a priori de la “taille” ou de la “portée” de l'ensemble d'entités étudiées. Il reste donc à comprendre comment, dans les **chapitres** ultérieurs, on fait évoluer le SCN sur des **niveaux** ou **paliers** différents (local, global, intermédiaire) afin d'observer des **patterns** qui s'articulent entre eux. C'est ce que vise à rappeler cette **Section 6.1.1.1.**

Même si l'équation $\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j))$ reste la même, on peut l'**appliquer** à divers **sous-ensembles** d'entités ou l'observer à plusieurs **niveaux**.

Au niveau **micro**, on s'intéresse par exemple à la formation de petits *clusters* locaux autour de quelques entités fortement connectées. Ces entités peuvent correspondre à un groupe de robots partageant une tâche concrète, ou à un sous-ensemble de concepts quasi synonymes dans un SCN linguistique.

Au niveau **macro**, on observe l'émergence de grands ensembles unifiant plusieurs sous-groupes, voire la structuration complète d'un **réseau** dans son ensemble, où la synergie prend une dimension plus globale (missions d'envergure, communautés thématiques, etc.).

D'un point de vue **mathématique**, la forme de la **mise à jour** $\Delta\omega_{i,j}$ n'évolue pas, seul change le **regard** que l'on porte, soit en "zoomant" sur quelques entités et leurs liaisons, soit en "dézoomant" pour englober la totalité. Cela ouvre la voie à l'étude des **interactions** entre ces focales multiples et la manière dont la dynamique DSL se manifeste à chaque échelle.

Lorsque la même **logique** d'auto-organisation (synergie, renforcement, inhibition) se répète à plusieurs **niveaux** d'observation, il n'est pas rare de constater une sorte de **fractalité** ou d'**auto-similarité**. Concrètement, le *pattern* qu'on voit apparaître à échelle locale, par exemple un "cœur dense" entouré de liaisons plus faibles, peut se reproduire (en plus grand) au niveau global. Sur un plan **mathématique**, cela signifie que la forme fonctionnelle de $F(\omega, S)$ ne dépend pas explicitement de la taille du système : si l'on isole un "cluster local" et qu'on y réapplique la même règle DSL, on peut observer un *motif* semblable à celui obtenu dans le réseau global. Cette invariance d'échelle s'apparente à la **fractalité** (ou "auto-similarité"), un concept clé qui peut se manifester dès lors que la "loi" de renforcement/inhibition demeure identique quel que soit le nombre d'entités ou la dimension du cluster. On pourrait ainsi repérer, dans un sous-SCN local, la même topologie (par exemple un noyau en cycle, ou des ramifications arborescentes) que l'on voit émerger sur tout le réseau.

Les **bénéfices** d'étudier et de prendre en compte cette **multi-échelle** sont nombreux. Sur un plan **cognitif**, il est précieux d'analyser un cluster local de haute cohérence, sans forcément perdre de vue l'organisation globale, ce qui donne un **système** plus modulaire et plus robuste. Un cluster local, par exemple, pourra persister, grandir ou changer de nature, même si le niveau macro subit d'autres transformations. Dans le **DSL**, la possibilité de gérer ces multiples échelles ouvre la porte à des **SCN** plus vastes, répartis en "groupements" hiérarchiques, que l'on peut manipuler ou faire coopérer entre eux. Cela rejoint aussi l'idée (explicitée au chapitre 6) qu'on peut disposer d'un "méta-niveau" de type **meta-SCN** (chap. 5.7.2) supervisant l'agrégation des sous-blocs locaux, réutilisant la même dynamique DSL.

Conclusion.

La **variation d'échelle** au sein d'un **SCN** n'est pas un simple artifice, mais bien une propriété fondamentale du **DSL** : la même **dynamique** de mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ s'applique tout autant au niveau local (quelques entités connectées) qu'au niveau global (tout le réseau), et c'est l'**observation** alternée de ces deux paliers qui révèle parfois un **caractère fractal**. Cette fractalité se manifeste lorsque la *même forme* d'auto-organisation ressurgit à plusieurs **niveaux** de granularité, s'exprimant dans la topologie des *clusters*. Ainsi, l'analyse **multi-échelle** s'impose comme une **extension naturelle** de la dynamique DSL, prolongeant la discussion des chapitres passés et introduisant le thème d'un **SCN** "à plusieurs échelons" qui sera développé dans l'ensemble du **Chapitre 6**.

6.1.1.2. Nécessité de s'intéresser aux différents niveaux (micro-clusters vs. macro-structures) et à la possible auto-similarité (concept fractal) qui peut émerger dans la répartition des entités

Le **Deep Synergy Learning (DSL)**, en tant que paradigme d'**auto-organisation** reposant sur le renforcement (ou l'inhibition) des liaisons $\omega_{i,j}$, ne se cantonne pas nécessairement à une seule **échelle** d'observation. Dans beaucoup de configurations pratiques, les entités $\{\mathcal{E}_i\}$ et leurs liaisons finissent par s'**ordonner** simultanément à plusieurs **niveaux** : on repère parfois un **micro-cluster** local de quelques entités fortement interconnectées, coexistant avec un **macro-cluster** plus vaste où la cohésion moyenne, tout en restant notable, est moins intense. Cette **pluralité d'échelles** soulève plusieurs questions d'intérêt méthodologique et mathématique, notamment la manière dont les processus de renforcement $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ se manifestent tant au niveau **local** (quelques noeuds) qu'au niveau **global** (regroupements massifs).

Micro-Clusters et Macro-Structures : une réalité fréquente

Les **micro-clusters** désignent de petits *noyaux* regroupant un **faible** nombre d'entités, mais reliées avec une grande **densité** ou de forts poids $\omega_{i,j}$. Par exemple, dans un **SCN** gérant un essaim de robots, ce peut être 2–3 robots partageant une **synergie** intense (même mission, même zone). Dans un **SCN** cognitif, il s'agirait de 4–5 concepts étroitement liés

par co-occurrence ou complémentarité sémantique. À l'inverse, un **macro-cluster** inclut un **grand** nombre d'entités, potentiellement un ensemble plus lâche mais occupant un rôle systémique (par ex. un sous-réseau d'agents répartis dans tout le système).

D'un point de vue **mathématique**, un micro-cluster apparaît comme un *sous-graphe* où les liaisons $\omega_{i,j}$ sont particulièrement élevées, tandis qu'un macro-cluster s'étend à un échelon supérieur, parfois *englobant* plusieurs micro-clusters internes. Cette cohabitation de niveaux multiples répond à une **variabilité** dans les rôles, les proximités ou les tâches assignées aux entités.

A. Mécanismes locaux vs. globaux et hiérarchie émergente

La même **règle** d'évolution $\omega_{i,j}(t+1) = F(\omega_{i,j}(t), S(i,j))$ s'applique a priori uniformément à l'ensemble du **SCN**. Cependant, le **renforcement** (ou l'inhibition) peut se manifester plus **rapidement** au niveau **local** : quelques entités proches en synergie renforcent vite leurs liaisons, formant un micro-cluster en quelques itérations. Au niveau **global**, la formation ou la stabilisation d'un grand cluster requiert plus de *temps* ou un certain enchaînement de fusions de micro-clusters. On observe alors une **hiérarchie** : le **DSL** ne converge pas uniquement vers "un cluster global" ou "un ensemble de micro-groupes", mais produit souvent des **strates** intermédiaires reliant micro et macro.

Cette **stratification** n'est pas juste un artefact : elle reflète la capacité d'un **SCN** à gérer différentes **échelles** de granularité, où les micro-groups jouissent d'une forte cohérence interne tandis que des macro-ensembles partagent un niveau d'affinité plus moyen mais suffisant pour la reconnaissance d'une "communauté" large. Sur le plan **ingénierie**, on exploite volontiers cette hiérarchie pour segmenter des **tâches** multiples (sous-groupes autonomes) tout en conservant une *vue* d'ensemble. Sur le plan **analyse**, cela souligne la **modularité** d'un SCN, précieuse dans de grands systèmes distribués (cf. Chap. 5.7).

B. Auto-similarité et phénomène fractal

Si les mêmes **mécanismes** de renforcement/inhibition demeurent constants à chaque **niveau**, on peut voir émerger des **motifs** qui se "reproduisent" à différentes échelles. C'est la notion d'**auto-similarité**, ou "**fractalité**". Sur un plan **mathématique**, on parle d'invariance d'échelle : la *topologie* ou la *distribution* des liens $\omega_{i,j}$ dans un petit cluster peut *ressembler* au motif global, simplement à une **taille** différente. Cette invariance est assez fréquente dans des systèmes biologiques ou cognitifs, où des *lois uniformes* (renforcement local, feedback global) peuvent induire des structures *self-similar*.

Pour illustrer, si un micro-cluster de 5 entités adopte une forme quasi "biclique" (deux sous-ensembles reliés par des poids forts), on peut découvrir que le grand cluster de 100 entités reproduit un motif analogue : un sous-ensemble principal et un second sous-ensemble, connectés par un pont central. L'origine de ce *pattern fractal* réside dans le fait que la **règle** DSL n'exclut pas, à petite ou grande échelle, des comportements similaires. Cette **fractalité** se révèle alors un indicateur d'une dynamique cohérente, où la forme d'organisation se répète localement et globalement.

C. Pourquoi étudier ces multiples échelles ?

D'un point de vue **fondamental**, cette pluralité d'échelles montre la **richesse** du DSL : la plasticité du SCN lui permet de façonner simultanément un **microniveau** (entités très liées) et un **macroniveau** (gros clusters englobant plusieurs micro-clusters). Cela introduit :

305. **Robustesse** : un micro-cluster n'est pas forcément dissous si la structure macro se reconfigure (et inversement).

306. **Vision holistique** : la même **énergie** ou la même **fonction** (Chap. 4) gère tout le réseau ; pourtant, l'**observation** différenciée (micro vs. macro) permet de saisir des phénomènes de *coopération locale* et *communalisation globale*.

307. **Clés d'analyse** : si la structure exhibe une **auto-similarité** notable, on peut quantifier son degré fractal par des indices topologiques ou géométriques, ce qui éclaire la dynamique d'auto-organisation.

Conclusion (6.1.1.2).

La mise en évidence de **multiples échelles** (micro-clusters et macro-structures) dans un **SCN** n'est pas un simple détail d'observation, mais relève d'une **nécessité théorique et pratique**. Les phénomènes de **renforcement** $\omega_{i,j}$ se déploient à divers niveaux, générant parfois un effet **fractal** : la constitution d'un *micro-cluster* ressemble, en miniature, à la consolidation d'un grand cluster. Comprendre ces différences et similitudes d'échelle, ainsi que les mécanismes d'**auto-similarité**, ouvre un nouveau champ de discussion. D'un côté, cela témoigne de la **versatilité** du DSL, capable de gérer des structures complexes et hiérarchiques. De l'autre, cela amorce la réflexion (développée dans ce **chapitre 6**) sur la conception de **SCN** multi-niveaux, articulés entre micro- et macro-organisation, suscitant des dynamiques d'**invariance d'échelle** susceptibles de produire des *patterns* auto-similaires (fractalité).

6.1.2. Objectifs du Chapitre

Après avoir mis en évidence (6.1.1) l'importance de considérer différentes **échelles** (micro, macro, intermédiaires) au sein d'un SCN (Synergistic Connection Network) et la possibilité qu'un **phénomène fractal** survienne, il est nécessaire de préciser **quels** sont les objectifs que ce chapitre se propose de remplir. En effet, la **multi-échelle** n'est pas simplement une curiosité : elle constitue un **cadre conceptuel** et **opérationnel** qui élargit la portée du DSL (Deep Synergy Learning), tout en soulevant des questions nouvelles en matière de **modélisation**, de **dynamique** et d'**implémentation**.

6.1.2.1. Expliquer comment la *multi-échelle* se traduit dans le DSL, et en quoi elle permet d'organiser et de coordonner différents niveaux d'abstraction

Le **Deep Synergy Learning (DSL)**, dans son expression la plus simple, décrit un réseau d'entités $\{\mathcal{E}_i\}$ dont les liaisons $\omega_{i,j}$ évoluent par le biais d'une règle d'**auto-organisation** (additive, multiplicative, etc.). Au cours des chapitres précédents, on a principalement envisagé cette dynamique dans le cadre d'un "**niveau unique**" : chaque entité interagit avec les autres via des synergies $S(i,j)$, et l'on observe un **SCN** global où l'on repère la formation de clusters (ou la répartition en sous-structures). Toutefois, dans des systèmes complexes, cette perspective **mononiveau** peut se révéler insuffisante, car l'**organisation** émerge souvent à **plusieurs échelles**. C'est précisément l'enjeu de cette section (6.1.2.1) : clarifier en quoi la *multi-échelle* s'intègre dans la logique du DSL et comment elle améliore la **coordination** entre divers niveaux d'abstraction.

Traduction de la multi-échelle dans le DSL

Sur le plan **mathématique**, la mise à jour des pondérations suit une équation générale du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j)\tau \omega_{i,j}(t)] \quad (\text{ou autre forme}).$$

Quand on introduit la *multi-échelle*, on considère que la **même** forme de règle DSL peut s'appliquer à plusieurs **niveaux** simultanément ou successivement. Autrement dit, chaque entité \mathcal{E}_i peut apparaître à un niveau "micro" (représentation plus fine) ou à un niveau "macro" (regroupement plus large de sous-entités), et la **dynamique** $\omega(t) \rightarrow \omega(t+1)$ peut être itérée aussi bien sur des micro-groupes que sur des macro-groupes.

Cette approche repose sur deux principes. D'abord, on **redéfinit** ce que signifie "entité" à chaque échelle : une **micro-entité** (ex. petite brique de connaissances, groupe restreint de neurones) ou une **macro-entité** (cluster plus étendu, aire plus large). Ensuite, on maintient la **règle d'auto-organisation** inchangée dans ses **fondements** : $\omega_{i,j}$ s'ajuste selon une synergie $S(i,j)$ et un terme de décroissance (ou d'inhibition). L'acte de "changer d'échelle" signifie, sur le plan analytique, "définir un second SCN" dont les nœuds sont désormais des **sous-ensembles** du premier SCN, et dont la **synergie** entre ces macro-nœuds s'évalue selon des agrégations de $\{\omega_{i,j}\}$ locales.

Organisation et coordination des différents niveaux

En introduisant la notion de **multi-échelle**, il ne s'agit pas seulement de contempler plusieurs niveaux de granularité ; on veut également que ces **niveaux** s'**influencent** mutuellement. Cela se traduit par :

- Des **flux ascendants** (bottom-up) : un **micro-cluster** émergeant (un petit ensemble fortement connecté à l'échelle locale) peut “remonter” vers une **macro-organisation**, se faisant reconnaître comme une entité cohérente dans le niveau supérieur.
- Des **flux descendants** (top-down) : le **macro-niveau**, doté d'une vue globale, peut imposer ou suggérer des contraintes aux micro-nœuds, par exemple en renforçant la liaison entre deux micro-entités si leur appartenance à un même macro-cluster s'avère cruciale pour la cohérence globale.

Ce processus de **coordination** multi-niveau confère au DSL une capacité à gérer des systèmes de **taille** et de **complexité** potentiellement beaucoup plus élevées. On peut, par exemple, répartir la **mise à jour** $\omega(t)$ selon des blocs locaux (micro) supervisés par un “meta-SCN” (macro), suivant les principes énoncés en Chapitre 5.7. Chaque **échelon** demeure fidèle à la règle DSL, mais opère sur des entités (ou clusters) d'échelle différente. On obtient alors une **architecture hiérarchique et modulaire** du SCN, évitant l'explosion combinatoire ou le chaos d'une organisation totalement “plate”.

Exemples concrets

Dans un **système cognitif** inspiré du cerveau, on peut assimiler le niveau “micro” à des micro-colonnes corticales (ou petits sous-réseaux de neurones) et le niveau “macro” à de grandes aires (voies sensorielles, cortex associatif). Les **mêmes** principes de synergie (co-activation neuronale) et d’update $\omega_{i,j}$ (synaptique) se répètent, de sorte que ce qui est acquis localement se reflète globalement. Dans un tel cadre, la *multi-échelle* illustre l'émergence et la stabilisation d'un schéma “macro” (ex. “aire visuelle connectée à aire frontale”) tout en conservant un ajustement local de micro-groupes.

En **robotique** multi-agent, le niveau “micro” peut consister en quelques robots géographiquement proches s'échangeant un flot de données et accroissant $\omega_{i,j}$ si leur collaboration s'avère fructueuse, tandis que le niveau “macro” traite la **mission** d'ensemble et “vois” un cluster englobant ces robots plus un autre groupe. Le SCN multi-échelle intègre alors la capacité de chaque micro-cluster à se spécialiser, tout en unifiant la planification globale.

Enjeux de la multi-échelle

Regrouper plusieurs **niveaux** dans un **SCN** soulève quelques questions spécifiques :

308. **Comment** modéliser la synergie entre macro-entités ? Faut-il calculer la **moyenne** ou le **maximum** des synergies entre leurs micro-composantes ?
309. **Comment** gérer la mise à jour ω si l'on bascule d'un niveau à l'autre ? On peut imaginer un *meta-SCN* (Chap. 5.7.2) où la synergie macro \hat{S} est engendrée par l'agrégation des synergies micro $\{S(i,j)\}$.
310. **Comment** éviter la désynchronisation entre micro- et macro-niveaux ? Cela rejoint les principes de coordination décrits plus haut : la dynamique est plus riche qu'un simple renforcement local, car un macro-cluster (englobant plusieurs micro-clusters) peut imposer ou suggérer une consolidation de certaines liaisons $\omega_{i,j}$.

Conclusion

La *multi-échelle* dans le **DSL** n'est pas seulement un ornement conceptuel : elle constitue un **prolongement** naturel de la logique d'auto-organisation à plusieurs **niveaux** d'abstraction. Du point de vue **mathématique**, cela signifie que la règle de mise à jour $\omega(t+1) = F(\omega(t), S)$ peut se répliquer (ou se composer) sur des échelons micro, macro et intermédiaires, conduisant à des phénomènes complexes d'**agrégation** et de **division** de clusters à différentes granularités. Sur le plan **ingénierie**, la multi-échelle offre un **levier** organisationnel : on peut circonscrire des **sous-groupes** très spécialisés (niveau micro) sans perdre la perspective d'ensemble (niveau macro). Enfin, cette organisation hiérarchique **facilite** la coordination dans les systèmes vastes, réduit la **complexité** en segmentant des **sous-problèmes**, et peut faire apparaître des phénomènes d'**auto-similarité** (cf. Section 6.1.1.2) rappelant la notion fractale. L'objectif du **Chapitre 6** sera d'approfondir ces idées et de décrire les mécanismes par lesquels le SCN gère la multi-échelle et orchestre la cohérence entre micro- et macro-structures.

6.1.2.2. Introduire la *fractalité* comme un cadre pour appréhender l'*auto-similarité* des patterns synergiques à diverses échelles

Lorsque l'on considère la **multi-échelle** dans un SCN (Synergistic Connection Network), il n'est pas rare de voir apparaître des **motifs** qui se **répètent** ou qui se **reproduisent** à plusieurs **niveaux** d'observation. Dans des systèmes complexes, cette forme de **répétition** ou d'**invariance d'échelle** est précisément décrite par la notion de **fractalité**. Par ailleurs, lorsque l'on observe qu'un micro-cluster local "ressemble", dans sa structure ou sa répartition de pondérations, à un macro-cluster plus vaste, on suspecte un **processus fractal** au sein du DSL. Le but de cette section (6.1.2.2) est donc de situer la *fractalité* dans la logique multi-échelle du **Deep Synergy Learning** et de montrer en quoi elle offre un *cadre théorique* pour analyser et comprendre l'**auto-similarité** qui peut émerger dans la répartition des entités.

A. Rappel du Concept de Fractalité

Une **fractalité** (ou structure **fractorisée**) se caractérise par l'**auto-similarité** à des échelles multiples. Géométriquement, dans un ensemble fractal au sens classique (Cantor, Von Koch, Sierpinski, etc.), la même "forme" se reproduit à plus petite échelle, parfois par homothétie ou transformations plus compliquées, d'où la notion d'**invariance d'échelle**. Sur le plan **mathématique**, une définition usuelle s'appuie sur :

$$\text{Auto-similarité} \Leftrightarrow \exists \text{ transformations } f_1, \dots, f_m: E = \bigcup_{k=1}^m f_k(E),$$

pour un ensemble E dit fractal. Les notions de **dimension fractale** ou **loi de puissance** (règle de type $\text{prob}(\text{taille} > s) \sim s^{-\alpha}$) apparaissent comme des marqueurs typiques.

B. Lien avec l'Auto-Similarité dans un Réseau

La **fractalité** est transposable à des **réseaux** (ou graphes) : on peut dire qu'un réseau exhibe une **structure fractale** si, en "zoomant" sur un sous-réseau (ou un cluster interne), on obtient une topologie "équivalente" (à un changement d'échelle près) au réseau global. D'un point de vue statistique, cela se manifeste souvent par des **lois de puissance** (distribution des degrés, distribution des tailles de clusters, etc.) et la conservation de certaines **mesures** topologiques quand on reconfigure le réseau à plus petit échelle (par ex. en regroupant des noeuds en "super-noeuds").

Dans un SCN, la **forme** du réseau émerge des pondérations $\{\omega_{i,j}\}$. Les *patterns synergiques* typiques (formation de sous-ensembles denses, arêtes intergroupes plus faibles) peuvent se **répéter** d'un micro-groupe à un macro-groupe. Il s'agit donc d'une **auto-similarité** dans la distribution de $\omega_{i,j}$, les *clusters* se formant à divers niveaux selon le même schéma (renforcement local, inhibition externe). Si l'on constate en pratique que cette logique donne, à plusieurs degrés de zoom, des **structures** analogues, on peut parler d'une **fractalité** ou d'un **mécanisme fractal** dans la répartition des entités $\{\mathcal{E}_i\}$.

C. Fractalité comme Cadre d'Analyse pour le DSL

1. Auto-similarité dans les Patterns Synergiques

Dans un DSL, un *pattern synergique* correspond à un certain agencement de liaisons $\{\omega_{i,j}\}$: par exemple, un *cycle dense*, un *noyau* fortement relié, ou un *clustering modulaire*. L'**auto-similarité** signifie que la **même forme** se retrouve à plusieurs niveaux : un micro-cluster pourrait présenter le même *type* de répartition de poids (forts au centre, décroissants en périphérie) qu'un macro-cluster englobant 50 entités, simplement "agrandi" ou "réplié" à une échelle supérieure.

Au niveau **mathématique**, on peut formaliser ceci en étudiant la "**dimension fractale**" (au sens box-counting ou similaire) du SCN : si cette dimension présente un comportement indiquant un *scaling law*, cela témoigne d'une invariance d'échelle dans l'organisation. De manière plus heuristique, on peut observer si la distribution des tailles de clusters $\{G_k\}$ suit une **loi de puissance** $P(\text{taille} = s) \sim s^{-\alpha}$. C'est un **signe** qu'il n'y a pas de *taille préférentielle*, ce qui est souvent gage d'une fractalité latente.

2. Notion de "Multi-niveau" fractal

Le **multi-niveau** (chap. 6.1.2.1) devient plus qu'une simple stratification : il se mue en **structure** réellement fractale si, en passant d'un niveau d'agencement micro à un niveau d'agencement macro, on reconstruit la "même dynamique" DSL, reproduisant la même topologie à une échelle démultipliée. Cela évoque des *systèmes biologiques* (une même logique neuronale répétée dans des modules de plus en plus grands, etc.) ou des *systèmes sociaux* (les mêmes mécanismes de collaboration localement et globalement, induisant des hiérarchies auto-similaires).

3. Bénéfices de l'Analyse Fractale

Robustesse et Invariance : La fractalité assure qu'une perturbation locale peut être absorbée par la répétition des structures ; si l'échelle micro subit une variation, l'échelle macro conserve son mode d'organisation, et réciproquement. **Mesures** : En quantifiant la dimension fractale (ou en repérant un exposant de loi de puissance), on caractérise la "façon" dont le réseau s'étale sur les diverses échelles.

Guidage : Si l'on souhaite que le SCN favorise une structure fractale (pour des raisons de modularité ou de robustesse), on peut paramétriser le DSL (les taux η , l'inhibition γ , etc.) de manière à encourager un schéma répétitif d'organisation.

D. Conclusion (6.1.2.2)

La **fractalité**, ou la *reconnaissance* d'un phénomène **auto-similaire** dans les *patterns* synergiques d'un **SCN**, ouvre un **cadre** analytique et conceptuel pour comprendre la "répétition" d'une logique d'auto-organisation à plusieurs **échelles**. Au-delà d'une simple observation, la fractalité procure :

- Une **explication** de la cohérence entre micro-clusters et macro-structures (la même forme se retrouve "répliquée" en plus grand),
- Un **outillage** de mesure (dimension fractale, lois de puissance) permettant de qualifier la "richesse" ou le "style" de la distribution $\{\omega_{i,j}\}$ et de l'auto-organisation.

Dans ce **chapitre 6**, se pencher sur la multi-échelle implique de solliciter le **concept fractal** pour appréhender la manière dont un **DSL** peut générer ou révéler ces *invariances d'échelle* dans la hiérarchie des clusters. Les sections suivantes détailleront les mécanismes et les implications techniques de la fractalité dans un SCN, montrant comment cette propriété de "répétition" à diverses granularités se retrouve dans la distribution des poids, l'organisation des clusters, et la dynamique de renforcement/inhibition elle-même.

6.1.2.3. Aborder les *modèles mathématiques*, les *dynamiques associées* et l'*implémentation* de l'apprentissage multi-échelle

Lorsque l'on traite la **multi-échelle** et la possible **fractalité** au sein d'un **SCN** (Synergistic Connection Network), il ne suffit pas de s'en tenir aux intuitions ou aux observations qualitatives. Il convient de proposer des **modèles mathématiques** explicites décrivant cette émergence d'échelles, d'**analyser** les dynamiques (lois de mise à jour, couplages inter-niveaux) et enfin de voir comment s'enchaîne l'**implémentation** concrète dans un système capable de manipuler de manière hiérarchique des entités et des clusters. Le **DSL** (Deep Synergy Learning) doit donc s'adapter pour gérer des **niveaux multiples**, tout en conservant la logique de renforcement/inhibition exposée jusqu'ici.

A. Modèles Mathématiques pour la Multi-Échelle

La première étape consiste à formuler un **SCN** à plusieurs **niveaux** : au niveau "de base" (micro), on a l'ensemble d'entités $\mathcal{E}_1, \dots, \mathcal{E}_n$ et leurs liaisons $\omega_{i,j}^{(0)}$. Quand un **cluster** local $\mathcal{C}_\alpha \subseteq \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ se dégage, on définit une *macro-entité* \mathcal{M}_α à l'échelle suivante (niveau 1). On construit alors une **nouvelle** matrice $\omega_{\alpha,\beta}^{(1)}$ décrivant les synergies **entre** ces macro-entités. Cette définition se poursuit pour des **niveaux** plus élevés ($k = 2, 3, \dots$), aboutissant à une hiérarchie. D'un point de vue purement **mathématique**, on peut :

311. Poser un *ensemble d'entités* $\mathcal{E}^{(0)}$ au niveau 0 (micro).

312. Définir par **agrégation** un ensemble $\mathcal{E}^{(1)}$ (macro-nœuds) où chaque $\mathcal{M}_\alpha^{(1)} \subseteq \mathcal{E}^{(0)}$ est un *cluster* repéré.

313. Répéter la logique, créant des ensembles $\mathcal{E}^{(k)}$ pour $k = 2, \dots, K$.

Ceci se rapproche des techniques de **coarse-graining** en physique statistique, où l'on regroupe des "sites" en "super-sites", et des approches de **quotient** de graphes en théorie des réseaux (chaque cluster devient un super-nœud). La mise à jour $\omega(t+1) = F(\omega(t))$ doit alors se décliner à chaque **palier**. Pour les liaisons entre macro-nœuds, on introduit une fonction d'**agrégation** Ψ des liaisons micro. Par exemple :

$$\omega_{\alpha,\beta}^{(k)} = \Psi(\{\omega_{i,j}^{(k-1)} \mid i \in \mathcal{M}_\alpha^{(k)}, j \in \mathcal{M}_\beta^{(k)}\}).$$

Une fois ces macro-pondérations définies, on peut **réappliquer** la règle DSL pour ajuster $\omega_{\alpha,\beta}^{(k)}$ en fonction d'une synergie $S^{(k)}(\alpha, \beta)$. Ainsi se construit un **modèle** multi-échelle strictement formalisé.

B. Dynamiques associées : couplage entre niveaux

En ayant défini des **niveaux** successifs ($k = 0, \dots, K$), il faut spécifier **comment** s'effectuent les **mises à jour** :

314. **Dynamique indépendante** : On peut imaginer que chaque niveau applique la **même** règle DSL (additive, multiplicative, etc.), comme si l'on disposait de $K + 1$ SCN indépendants. Le lien entre eux est simplement la **projection** (les macro-nœuds de k sont issus de sous-ensembles du niveau $k - 1$).

315. **Rétroaction inter-niveau** : on peut aller plus loin, en autorisant un **feedback** descendante (top-down) : si, au niveau macro, on constate un lien $\omega_{\alpha,\beta}^{(k)}$ extrêmement fort, on incite (au niveau $k - 1$) les liaisons $\{\omega_{i,j}^{(k-1)}\}$ reliant des entités micro internes à α, β à se consolider. Symétriquement, un cluster local au niveau $k - 1$ peut "remonter" un signal vers la macro-liaison $\omega_{\alpha,\beta}^{(k)}$.

316. **Schéma cyclique ou séquentiel** : Dans certains algorithmes, on effectue d'abord la **mise à jour** DSL sur le niveau micro (cette itération se stabilise), puis on agrège pour constituer des macro-nœuds, calcule $\omega^{(k+1)}$ et exécute quelques étapes de mise à jour macro, etc., dans un **cycle**. Cela aboutit à un SCN hiérarchique où chaque palier se stabilise partiellement avant de passer au suivant.

Ces dynamiques **couplées** (multi-niveau) peuvent favoriser une **auto-similarité** (Section 6.1.2.2) dans la répartition des clusters, puisque la *même logique* de renforcement/inhibition se répercute depuis le niveau local jusqu'au niveau global. Elles étendent la simple équation $\omega(t+1) = F(\omega(t))$ en un système **multi-échelle** :

$$\omega^{(k)}(t+1) = F_k(\omega^{(k)}(t), \omega^{(k-1)}(t), \omega^{(k+1)}(t)),$$

introduisant potentiellement un **couplage** $\omega^{(k)} \leftrightarrow \omega^{(k-1)} \leftrightarrow \omega^{(k+1)}$.

C. Implémentation de l'apprentissage multi-échelle : méthodes pratiques

Sur le plan **ingénierie**, on peut mettre en place une **architecture** logicielle modulaire. Chaque palier k dispose d'un *Module Mise à Jour* et d'un *Module Synergie*, gérant respectivement :

- La mise à jour $\omega^{(k)}$ via la règle DSL,
- Le calcul $S^{(k)}(\alpha, \beta)$ pour les macro-nœuds $\mathcal{M}_\alpha, \mathcal{M}_\beta$.

Une **passerelle** d'agrégation Ψ relie le niveau k au niveau $k - 1$, tandis qu'un **mécanisme** de feedback (optionnel) renvoie des consignes top-down au niveau inférieur. Selon la **complexité** du SCN, on applique un schéma :

317. **Séquentiel** : on met à jour d'abord $\omega^{(0)}$ (micro), on regroupe en macro-nœuds, on met à jour $\omega^{(1)}$ (macro), etc.

318. **Parallèle** : chaque niveau évolue en parallèle, on synchronise à intervalles réguliers.

319. **Distribué** : si ω est déjà réparti (Chap. 5.7), on peut répliquer la hiérarchie par bloc local, puis un *meta-niveau* global.

Des algorithmes concrets — par exemple, un “**hierarchical SCN**” — consistent à détecter automatiquement des clusters stables au niveau k , les étiqueter pour créer $\mathcal{E}^{(k+1)}$, initialiser $\omega^{(k+1)}$ par agrégation, et enfin lancer la règle DSL à ce niveau. Il s’agit donc d’une **fusion** entre le concept de “clustering itératif” (type agglomératif) et la dynamique DSL.

Conclusion

Le troisième volet de cette introduction à la **multi-échelle** est la **modélisation mathématique**, la **dynamique associée** et la **mise en œuvre** de l’**apprentissage** multi-niveau dans un SCN. Les points clés sont :

320. **Modèles** : définir la hiérarchie (niveau 0, 1, 2, …), les règles d’**agrégation** (comment on regroupe des entités en macro-nœuds) et la manière d’initialiser les pondérations $\omega^{(k)}$.

321. **Dynamiques** : préciser si chaque niveau applique la **même** règle DSL (additive, multiplicative, etc.), comment s’effectuent les **rétroactions** entre micro et macro, et de quelle façon la structure fractale (auto-similarité) se bâtit ou se maintient.

322. **Implémentation** : articuler un système modulaire (Module Synergie, Module Mise à Jour) capable de *monter* et *descendre* dans la hiérarchie, éventuellement de façon parallèle ou distribuée.

Grâce à ces approches, un **SCN** peut gérer de vastes ensembles d’entités dans un **cadre** unifiant la **multi-échelle** et l’**invariance d’échelle** (fractalité potentielle). On passe donc de la simple **observation** de multiples niveaux (Sections 6.1.1 et 6.1.2.1–2) à une **construction formelle** (règles, formules, algorithmes) permettant de réaliser effectivement un **apprentissage** hiérarchique, reflétant la richesse du **DSL** dans des contextes réels ou de recherche avancée.

6.1.3. Structure du Chapitre

Après avoir présenté (6.1.1 et 6.1.2) les **motivations** du DSL (Deep Synergy Learning) à aborder la **multi-échelle** et la **fractalité**, il est utile de donner un **aperçu** de l’organisation de ce chapitre 6, afin que le lecteur puisse se repérer dans les différentes sections et comprendre le fil conducteur.

6.1.3.1. Aperçu : (6.2) principes et hiérarchies, (6.3) fractalité et auto-similarité, (6.4) interactions multi-niveau, (6.5) dynamiques multi-échelles, (6.6) études de cas, (6.7) conclusion

La Section 6.1.3.1 s’attache à présenter la **trame du Chapitre 6**, qui conduit le lecteur depuis les **concepts** de multi-échelle dans un **SCN** (Synergistic Connection Network) et leur déclinaison sous la forme d’une **hiérarchie** (Section 6.2), jusqu’aux **détails** des dynamiques multi-niveau (Sections 6.4, 6.5) et les **études de cas** illustratives (Section 6.6). Chaque section vient ainsi approfondir l’un des grands **thèmes** introduits dans la première partie du chapitre (6.1), en offrant un fil directeur cohérent et progressif :

(6.2) Principes et hiérarchies

La Section 6.2 vise à poser les **règles générales** de la multi-échelle. Elle montre comment, dans un **SCN**, on peut distinguer plusieurs **niveaux** de granularité (micro, macro, voire intermédiaire), définissant une **hiérarchie**. D’un point de vue **mathématique**, cela exige de préciser :

323. Comment on **agrège** des entités de bas niveau (\mathcal{E}_i) en “super-nœuds” ou macro-entités au niveau supérieur.

324. Quelles sont les **lois** ou **règles** que l’on emploie pour moduler les liens ω entre macro-entités, en s’inspirant ou en prolongeant la dynamique DSL originelle.

Sur le plan conceptuel, cette partie s'intéresse à la **théorie des systèmes emboîtés** et introduit des arguments de **cohérence hiérarchique** (flux ascendants “bottom-up”, flux descendants “top-down”), illustrant l’idée que plusieurs paliers de représentation se coordonnent.

(6.3) Fractalité et auto-similarité

La **Section 6.3** développe la **notion de fractalité** (déjà esquissée en 6.1.2.2) comme un **cadre** pour analyser la possible **auto-similarité** dans les *patterns* synergiques d’un SCN multi-échelle. Concrètement, on décrit les **fondements** mathématiques de la fractalité (dimension fractale, lois de puissance, auto-similarité stricte ou statistique) et on explique en quoi un **SCN** peut, dans certains cas, présenter ces attributs. On approfondit aussi :

325. Les **indicateurs** à mesurer pour repérer la fractalité (distribution des tailles de clusters, coefficients d’échelle, etc.).

326. Les **limites** : tout réseau auto-organisé n’est pas fractal, et les conditions (paramétriques, algorithmiques) favorisant l’invariance d’échelle demeurent un sujet de recherche.

On met ainsi en avant la dimension **théorique** du **DSL** lorsqu’il rencontre des phénomènes d’**invariance d’échelle** : comment un micro-cluster “ressemble” à un macro-cluster, et comment un *même* schéma d’organisation se reproduit.

(6.4) Interactions multi-niveau

La **Section 6.4** se focalise sur les **mécanismes de coordination** entre les divers niveaux (ou paliers) d’un SCN multi-échelle. Plus précisément, elle décrit :

327. **Flux ascendants** (bottom-up) : si un cluster local se stabilise, comment cette stabilisation est “signalée” au niveau macro, conduisant à la consolidation d’un super-nœud ou à un réagencement global.

328. **Flux descendants** (top-down) : si le macro-niveau détecte un arrangement global stable ou un objectif supérieur, il peut “descendre” l’information vers les micro-nœuds pour orienter leurs liens $\omega_{i,j}$ ou imposer certaines synergies minimales.

Cette partie s’attache aussi à la notion de **synchronisation** inter-niveau : doit-on bloquer un niveau pendant qu’un autre se met à jour, ou peut-on évoluer en parallèle ? Les **conflits** (un micro-cluster local s’opposant aux objectifs macro) sont abordés, de même que la manière d’y remédier par un protocole multi-niveau.

(6.5) Dynamiques multi-échelles

La **Section 6.5** passe à la **dynamique** d’apprentissage elle-même : comment applique-t-on concrètement la **règle DSL** (additive, multiplicative...) à chaque palier, de façon séquentielle ou simultanée ? Plusieurs algorithmes sont proposés, illustrés par des **pseudo-codes**. Cette partie répondra, par exemple, aux interrogations :

329. **Ordonnancement** des mises à jour : met-on à jour toutes les liaisons $\omega^{(k)}$ au niveau k avant de passer au niveau $k + 1$, ou opère-t-on en *allers-retours* ?

330. **Aggrégation et Division** : comment un cluster macro peut se *subdiviser* si sa cohésion interne fléchit, comment un micro-cluster peut *fusionner* avec un autre, etc.

331. **Impact sur la complexité** : la structuration en **hiérarchie** rend-elle la gestion du $O(n^2)$ plus aisée en répartissant les calculs ?

Au final, la **Section 6.5** fournit un **cadre** algorithmique pour mettre en pratique la **multi-échelle** dans un SCN, reliant les définitions théoriques (Sections 6.2–6.3) et les **exemples** ou **études de cas** (Section 6.6).

(6.6) Études de cas

La **Section 6.6** montre **plusieurs** applications ou exemples concrets où le **DSL** multi-échelle apporte une **plus-value** :

332. **Vision multi-modale** : comment on segmente des “patchs” d’image en micro-clusters, puis on agrège des zones plus larges en macro-structures, etc.

333. **Analyse contextuelle du langage** : un texte divisé en segments (micro-clusters de mots), puis agrégé en chapitres ou thèmes (macro).

334. **Robotique sensorimotrice** : la gestion de micro-groupes de robots physiquement proches, un meta-niveau coordonnant plusieurs groupes.

335. **Agents conversationnels** : hiérarchie de modules cognitifs (lexicaux, sémantiques, pragmatiques) dans un SCN multi-niveau.

Chaque cas d’usage illustre la **capacité** du DSL à *emprunter* des chemins multi-niveau, parfois même fractals, pour organiser des ensembles hétérogènes de données ou d’entités.

(6.7) Conclusion

La **Section 6.7** clôt le chapitre en rappelant :

- L'**intérêt** des hiérarchies multi-échelle : robustesse, réduction de complexité, possibilité d’exploiter la fractalité, etc.
- Les **points clés** (lois d’agrégation, dynamiques multiples, feedback inter-niveau) et la **portée** de ces schémas.
- Les **perspectives** pour les chapitres suivants ou pour des recherches futures, mettant en avant la **souplesse** de la multi-échelle dans l'**auto-organisation** d’un SCN.

Conclusion

Avec ce **plan** (Sections 6.2–6.7), le **Chapitre 6** engagera le lecteur dans :

336. Une mise en place **conceptuelle** (6.2) de la multi-échelle et des hiérarchies,

337. Une plongée **théorique** (6.3) dans la fractalité et l’auto-similarité,

338. Une exploration **technique** (6.4, 6.5) des interactions et des dynamiques multi-niveau,

339. Une **application** concrète (6.6) via des études de cas,

340. Une **synthèse** (6.7) regroupant ces apports.

Cette progression s’inscrit dans la continuité de l’idée exposée en (6.1) : à mesure que l’on **monte** dans les **échelles**, le **DSL** peut révéler ou forger des clusters multi-niveaux, parfois fractals, offrant un cadre d’**auto-organisation** plus abouti et plus modulable qu’un réseau “à plat”.

6.2. Principes de l'Apprentissage Multi-Échelle

6.2.1. Hiérarchies Synergiques et Niveaux d'Abstraction

- 6.2.1.1. Notion de micro-cluster vs. macro-cluster.
- 6.2.1.2. Rôles et fonctions des différents niveaux (local, intermédiaire, global).
- 6.2.1.3. Exemples : micro-réseaux d'entités sensorielles vs. macro-ensembles pour une vue conceptuelle.

6.2.2. Théorie des Systèmes Emboîtés

- 6.2.2.1. Systèmes emboîtés : cellule → tissu → organe → organisme (analogie).
- 6.2.2.2. Transposition au DSL : agrégation progressive d'entités vers des super-nœuds (macro-nœuds).
- 6.2.2.3. Multi-niveau comme “pilier” pour la gestion de la complexité.

6.2.3. Rétroactions et Anticipations Multi-Niveau

- 6.2.3.1. Flux ascendants (bottom-up) : micro → macro.
- 6.2.3.2. Flux descendants (top-down) : macro → micro.
- 6.2.3.3. Communication entre niveaux synergiques et stabilisation.

6.2. Principes de l'Apprentissage Multi-Échelle

Lorsqu'on souhaite doter le **DSL** (Deep Synergy Learning) d'une capacité à se déployer sur **plusieurs échelles** (micro, intermédiaire, macro), il faut d'abord saisir l'idée de **hiérarchies synergiques** et des **niveaux d'abstraction** qu'elles génèrent. La section 6.2.1 introduit cette notion, puis 6.2.2 l'illustre via la "théorie des systèmes emboîtés" et 6.2.3 décrit comment ces niveaux interagissent par des flux ascendants (bottom-up) et descendants (top-down). L'objectif est de poser les **fondations mathématiques** permettant à un **SCN** (Synergistic Connection Network) de fonctionner harmonieusement à diverses granularités — du micro-cluster au macro-cluster — de façon coordonnée.

6.2.1. Hiérarchies Synergiques et Niveaux d'Abstraction

Dès qu'on dépasse un certain degré de complexité (grand nombre d'entités, de liens $\omega_{i,j}$), on constate souvent que les entités s'**auto-organisent** en **clusters** de tailles diverses, de petits regroupements locaux et, potentiellement, de grands ensembles plus étendus. Cette diversité de "tailles de clusters" correspond à la **pluralité** des échelles dans le réseau. On parle de **hiérarchies** quand ces clusters de niveau inférieur peuvent se regrouper (ou s'agréger) pour former des "super-clusters" plus vastes, suggérant différents **niveaux d'abstraction** — on peut alors manipuler des entités "brutes" (micro-niveau) ou des entités "agrégées" (macro-niveau).

6.2.1.1. Notion de micro-cluster vs. macro-cluster

Dans de nombreux contextes d'**auto-organisation** via un SCN, il se forme simultanément des **groupes** à différentes **tailles** et différents **degrés de spécialisation**. Il est alors opportun d'introduire la distinction entre **micro-clusters** et **macro-clusters**. Les premiers correspondent à des **noyaux** restreints d'entités intimement reliées, tandis que les seconds, plus vastes, englobent parfois plusieurs micro-clusters et présentent une cohésion au niveau global. Cette section (6.2.1.1) précise ces deux concepts, en montrant comment ils s'articulent dans le cadre d'un **SCN** et comment leur co-existence fonde l'idée même de **multi-échelle**.

Définition de micro-cluster et de macro-cluster

Un **micro-cluster** est un groupe de taille modeste, par exemple $\{\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_k}\}$, dans lequel les pondérations ω_{i_p, i_q} s'avèrent **fortement** élevées pour chaque couple (i_p, i_q) appartenant au groupe. On y retrouve généralement une **densité** ou une **somme** de liaisons particulièrement importante par rapport au nombre k d'entités. Ainsi, un micro-cluster a souvent vocation à remplir une **tâche locale**, quelques robots partagent une mission spécialisée, quelques concepts (en sémantique) forment un champ lexical soudé, ou quelques neurones (en neurosciences) constituent un module fortement interconnecté. La force d'un tel cluster se mesure par :

$$\Omega(\mathcal{C}) = \sum_{i, j \in \mathcal{C}} \omega_{i, j},$$

montrant que $\Omega(\mathcal{C})$ peut être élevé malgré un groupe \mathcal{C} de taille réduite.

Un **macro-cluster**, en revanche, désigne un sous-ensemble du SCN **plus large**, pouvant inclure plusieurs micro-clusters en son sein. Les entités y sont globalement **cohésives** (au sens d'une synergie moyenne non négligeable), même si toutes n'entretiennent pas nécessairement des liens très forts les unes avec les autres. Au niveau d'un macro-cluster, on observe plutôt une forme de **convergence** globale qui rassemble un ensemble conséquent d'entités, par exemple en vue d'un objectif commun, d'une thématique plus générale, ou d'une mission d'envergure. La mesure $\Omega(\mathcal{C})$ pour un macro-cluster \mathcal{C} peut aussi être importante, mais cette fois-ci, la taille $|\mathcal{C}|$ est grande et la densité moyenne peut être plus modérée, tout en restant assez élevée pour être distinguée du "reste" du réseau.

Exemples et Co-existence

Dans un **SCN** à fort nombre n d'entités, la co-existence de **micro-clusters** et de **macro-clusters** est fréquente. On rencontre :

341. Plusieurs **micro-clusters** (groupes de quelques entités) — chacun très cohésif.

342. Un (ou plusieurs) **macro-cluster(s)** rassemblant, à plus large échelle, certaines entités ou micro-clusters proches, formant un **palier** d'organisation supérieur.

Cette dualité émerge par la **dynamique** DSL. On peut voir un renforcement rapide et local conduire à la formation de petits noyaux soudés (micro-clusters), tandis que l'évolution à plus long terme ou à plus vaste échelle permet l'agrégation de plusieurs micro-groupes en un macro-cluster. Cela reflète la **réalité** de maints systèmes biologiques, cognitifs ou sociotechniques, où des structures de différentes tailles coexistent et se superposent.

Vers une Structure Hiérarchique

La présence simultanée de micro- et macro-clusters évoque naturellement une **hiérarchie** d'échelles. On peut, par exemple, poser :

- Le **niveau 0** (ou échelle micro) : chaque entité \mathcal{E}_i en détail, permettant de repérer des groupes restreints,
- Le **niveau 1** (ou échelle méso) : on regroupe certains micro-clusters pour former des sous-réseaux plus grands,
- Le **niveau 2** (ou échelle macro) : on identifie un ou plusieurs macro-clusters dominants, englobant éventuellement plusieurs niveaux inférieurs.

Chacun de ces niveaux peut s'analyser, dans le **SCN**, par une matrice de pondérations $\omega^{(k)}$ spécifique (voir Section 6.2.2), reflétant des *super-nœuds* (macro-entités) si l'on agrège les micro. L'**apprentissage** multi-échelle (chap. 6.4–6.5) s'appuiera alors sur des **règles** d'agrégation, de rétroaction (top-down, bottom-up) et de synchronisation inter-niveau pour maintenir la cohérence globale.

Conclusion

La **notion** de *micro-cluster* et de *macro-cluster* en **DSL** ne se réduit pas à une dichotomie ad hoc : elle traduit la **réalité** d'un SCN où l'on voit émerger simultanément, à des **échelles** distinctes, des groupes de taille modeste (micro) et d'autres plus imposants (macro). Cette coexistence constitue la **base** de la structure hiérarchique d'un **SCN** multi-échelle. Les micro-clusters peuvent manifester des synergies locales fortes, tandis que les macro-clusters, plus amples et plus lâches, contribuent à une cohésion d'ensemble ou à une mission commune de grande portée. Cette articulation (micro→macro) fonde une **hiérarchie** (ou un emboîtement) cruciale pour la réduction de complexité, la robustesse et l'**adaptation** du réseau, sujets qui seront approfondis dans la suite du chapitre.

6.2.1.2. Rôles et Fonctions des Différents Niveaux (local, intermédiaire, global)

Dans l'**étude** des **SCN** appliqués au **DSL**, il apparaît essentiel de distinguer plusieurs échelles d'organisation qui permettent de moduler la complexité du système de manière hiérarchique. La présente section expose la finalité et le rôle de trois niveaux d'abstraction distincts : le **niveau local** (ou micro-niveau), le **niveau intermédiaire** (ou mésos-niveau) et le **niveau global** (ou macro-niveau). Chacun de ces niveaux possède une fonction propre dans la dynamique d'auto-organisation du réseau, comme nous le développerons ci-après.

A. Niveau Local (Micro-Niveau)

Le **niveau local** correspond à la granularité la plus fine du SCN, où les entités individuelles $\{\mathcal{E}_i\}$ sont traitées telles qu'elles apparaissent, ou bien regroupées en micro-clusters très restreints. À ce niveau, la **matrice locale** des pondérations, que l'on peut noter $\omega^{(\text{local})}$ ou encore $\omega^{(0)}$, décrit les interactions directes entre entités selon la règle de mise à jour fondamentale du DSL.

Ce niveau local joue un rôle de **détection** initiale et de **spécialisation** en permettant une réactivité élevée, souvent à l'aide de paramètres tels que η_{loc} relativement élevés, afin de saisir rapidement les variations ou les nouveaux stimuli. Les micro-clusters ainsi formés constituent la base sur laquelle se construiront ultérieurement les niveaux supérieurs.

En effet, ces structures locales, fortement interconnectées (ce qui peut se traduire par une valeur élevée de la **synergie locale** $\Omega(\mathcal{C})$ pour un cluster \mathcal{C}), pourront être agrégées pour former des super-nœuds au niveau intermédiaire.

B. Niveau Intermédiaire (Méso-Niveau)

Le **niveau intermédiaire** intervient en regroupant les micro-clusters issus du niveau local pour former des super-nœuds, notés généralement $\{\mathcal{M}_\alpha\}$. À ce palier, la dynamique d'**agrégation** vise à condenser les interactions locales afin de rendre plus lisibles et gérables les relations complexes qui se tissent à l'échelle du réseau. La nouvelle matrice des pondérations, que nous pouvons désigner par $\omega^{(\text{inter})}$, codifie ainsi la synergie entre ces super-nœuds, par exemple via une fonction

$$\omega_{\alpha,\beta}^{(\text{inter})} = \Psi\left(\{\omega_{i,j}^{(\text{local})}\}_{i \in \mathcal{M}_\alpha, j \in \mathcal{M}_\beta}\right),$$

où Ψ est une fonction d'agrégation adaptée aux données locales. À ce niveau, le rôle principal est à la fois d'**interface** et de **coordination**. Le niveau intermédiaire permet d'informer le niveau global de l'état des sous-ensembles tout en transmettant, via des rétroactions descendantes, des consignes susceptibles d'ajuster la dynamique locale. Par ailleurs, la mise à jour des pondérations intermédiaires suit également une loi du type

$$\omega_{\alpha,\beta}^{(\text{inter})}(t+1) = \omega_{\alpha,\beta}^{(\text{inter})}(t) + \eta_{\text{inter}} [S_{\text{inter}}(\alpha, \beta) - \tau_{\text{inter}} \omega_{\alpha,\beta}^{(\text{inter})}(t)],$$

ce qui confère une homogénéité dans l'approche d'auto-organisation sur plusieurs échelles.

C. Niveau Global (Macro-Niveau)

Au sommet de la hiérarchie se situe le **niveau global** qui intègre l'ensemble des super-nœuds pour former une vision unifiée du réseau. Le niveau global, représenté par des macro-clusters $\{\mathcal{M}_\alpha^{(\text{global})}\}$, sert à dégager la **structure ultime** du SCN et à fournir une représentation condensée de la configuration globale du système. La matrice des pondérations globales, notée $\omega^{(\text{global})}$, est mise à jour selon une dynamique analogue à celle des niveaux inférieurs, par exemple

$$\omega_{\alpha,\beta}^{(\text{global})}(t+1) = \omega_{\alpha,\beta}^{(\text{global})}(t) + \eta_{\text{glob}} [S_{\text{glob}}(\alpha, \beta) - \tau_{\text{glob}} \omega_{\alpha,\beta}^{(\text{global})}(t)],$$

où $S_{\text{glob}}(\alpha, \beta)$ représente une synergie évaluée entre de grands ensembles, et η_{glob} ainsi que τ_{glob} sont des paramètres adaptés à cette échelle. Ce niveau a pour rôle principal de fournir une **vision d'ensemble** qui stabilise l'organisation du réseau et, le cas échéant, de servir d'interface avec des systèmes externes (par exemple, des modules de visualisation ou de décision). En outre, le niveau global participe à la **stabilisation** du SCN en verrouillant la convergence d'une configuration globale quand la synergie globale atteint un certain seuil.

D. Complémentarité et Couplage des Niveaux

La force d'un **SCN multi-échelle** réside dans l'**interaction** étroite entre les différents niveaux. Les rétroactions bottom-up et top-down jouent un rôle crucial en permettant aux micro-clusters de s'agrégger en super-nœuds, et aux structures globales de diffuser des contraintes qui orientent la dynamique locale. Cette **couverture hiérarchique** réduit la complexité du problème en divisant le réseau en sous-ensembles gérables, accélère la convergence en isolant les interactions locales des influences globales excessives, et favorise l'émergence d'une **auto-similarité fractale** lorsque la même règle DSL se répète sur plusieurs niveaux d'abstraction.

D'un point de vue mathématique, cette approche hiérarchique permet de passer d'un problème de complexité $\mathcal{O}(n^2)$ à des sous-problèmes de moindre dimension, ce qui facilite l'analyse de la **stabilité** et la conception de **régulateurs** à chaque échelle. Ainsi, les interactions locales peuvent être traitées indépendamment avant d'être agrégées au niveau global via des fonctions d'agrégation spécifiques telles que Ψ mentionnée ci-dessus.

Conclusion (6.2.1.2)

La **structuration** d'un SCN en niveaux (local, intermédiaire, global) représente bien plus qu'un simple outil d'organisation ; elle définit une **architecture hiérarchique** dans laquelle chaque palier remplit un rôle fonctionnel et mathématique distinct. Le **niveau local** se charge de capturer les interactions fines entre entités, le **niveau intermédiaire** regroupe ces interactions pour constituer des super-nœuds cohérents, et le **niveau global** offre une vue d'ensemble stabilisatrice et structurante. Les rétroactions entre ces niveaux permettent d'obtenir une **cohérence** à la fois micro et macro, tout en réduisant la complexité du système. Les chapitres suivants, notamment 6.2.2 et 6.2.3, détailleront les mécanismes algorithmiques et mathématiques permettant de mettre en œuvre cette hiérarchie et d'exploiter les propriétés d'auto-similarité qui en découlent, conformément aux principes exposés dans ce chapitre.

6.2.1.3. Exemples : micro-réseaux d'entités sensorielles vs. macro-ensembles pour une vue conceptuelle

Les rôles des différents niveaux (micro, intermédiaire, macro) dans un SCN s'appréhendent de manière plus concrète lorsqu'on considère des cas exemplaires de la **coexistence** de micro-réseaux d'entités très locales (capteurs, données brutes, etc.) et de macro-ensembles plus englobants (concepts, missions, sous-systèmes de grande dimension). La présente section illustre comment un SCN peut à la fois gérer des **micro-clusters** — typiquement quelques entités fortement reliées — et des **macro-clusters** représentant un agrégat plus large et plus abstrait. Deux exemples sont développés : (A) la formation de micro-réseaux sensoriels, (B) la formation de macro-ensembles conceptuels.

A. Micro-réseaux d'entités sensorielles

Un scénario fréquemment rencontré est celui d'un grand nombre de **capteurs** ou **petites sources** d'information (capteurs IoT, neurones sensoriels, etc.). Chaque capteur \mathcal{E}_i constitue une entité locale avec laquelle on peut former un micro-réseau. Dans un SCN, ces capteurs établissent des liens $\omega_{i,j}$ qui se renforcent s'ils détectent une similarité ou corrélation forte. On parle alors de **micro-cluster** dès lors qu'un petit ensemble de capteurs $\{\mathcal{E}_{i_1}, \dots, \mathcal{E}_{i_k}\}$ se découvre une synergie ω_{i_p, i_q} élevée.

Soit $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S_{\text{corr}}(i,j) - \tau \omega_{i,j}(t)]$, où $S_{\text{corr}}(i,j)$ exprime le degré de **corrélation** (ou autre similarité) entre les signaux sensoriels de i et j . Les micro-clusters se repèrent lorsqu'un groupe \mathcal{C} présente une somme interne

$$\Omega(\mathcal{C}) = \sum_{i,j \in \mathcal{C}} \omega_{i,j}$$

assez forte pour dépasser un seuil θ . Ces clusters, de taille modeste, reflètent des **interactions fines** et très **spécifiques** (par ex. capteurs proches géographiquement ou capteurs partageant la même tendance de mesure).

Au niveau micro, les entités demeurent très **réactives** : un capteur peut rapidement renforcer ses liens avec un voisin détectant le même événement. La dynamique DSL à ce stade est **rapide** et **locale**. Une fois stabilisés, ces micro-clusters peuvent être "fournis" en **agrégation** (Section 6.2.2) pour constituer un super-nœud au niveau supérieur (méso). Cela soulage la complexité globale en traitant chaque petit groupe cohésif comme une **brique** de niveau plus élevé.

B. Macro-ensembles pour une vue conceptuelle

À l'autre extrême, on observe parfois dans un SCN la formation de **macro-ensembles** englobant un large ensemble d'entités pour représenter un **concept** global ou une **mission** d'envergure. Dans un système cognitif, cela pourrait correspondre à un *champ sémantique* ; dans un système multi-robot, à un *groupe de grande taille* coopérant à un objectif.

Sur le plan **mathématique**, un macro-ensemble \mathcal{G} peut rassembler plusieurs dizaines, voire centaines d'entités. La force globale $\Omega_{\mathcal{G}} = \sum_{i,j \in \mathcal{G}} \omega_{i,j}$ n'a pas besoin d'être extrêmement dense (comme dans un micro-cluster) mais reste **significative** pour qualifier un **macro-cluster**. Ce super-groupe agit souvent comme un **super-nœud** dans un niveau macro, en lien avec d'autres macro-ensembles, formalisé par $\omega_{\Gamma, A}^{(\text{global})}$.

Exemples

343. Dans une **architecture neuronale inspirée du cerveau**, un “macro-ensemble” peut symboliser une grande aire associant des fonctions cognitives plus abstraites.
344. Dans un **réseau sémantique**, un macro-cluster incarne un **concept** (ex. “géométrie”, “mécanique quantique”) regroupant des micro-clusters de définitions, de synonymes, etc.
345. Dans la **robotique** multi-agent, un macro-ensemble désigne une “équipe étendue” : plusieurs sous-équipages plus petits collaborent pour un objectif final commun.

C. Articulation : du micro au macro

Le **niveau micro** (petits clusters sensoriels) se trouve à la base. Le **niveau macro** (grands ensembles conceptuels) se trouve au sommet. Entre les deux, un **méso-niveau** (chap. 6.2.2) peut regrouper des sous-groupes intermédiaires. Chaque palier propose une **granularité** distincte mais reliée :

346. Le micro-niveau, réactif et spécialisé,
347. Le méso-niveau, coordinateur ou “articulateur”,
348. Le macro-niveau, plus abstrait, offrant une synthèse plus large ou une “vision d’ensemble”.

Les **micro-réseaux sensoriels** génèrent des signaux qui s’agrègent en macro-ensembles conceptuels : la consolidation se fait par **flux descendant** (bottom-up). Symétriquement, le niveau macro peut imposer des **contraintes** ou **orientations** descendantes (top-down) : par exemple, un macro-objectif “réduire la consommation d’énergie” se traduit, au niveau micro, par un ajustement des pondérations $\omega_{i,j}$ favorisant certains capteurs plus pertinents pour cet objectif.

Conclusion

Ces deux **exemples** — (A) micro-réseaux sensoriels, (B) macro-ensembles conceptuels — illustrent clairement la **cohabitation** de micro-clusters et de macro-clusters dans un **SCN** à multiples échelles. Les micro-réseaux portent sur des entités localement cohésives (capteurs, neurones, etc.) tandis que les macro-ensembles englobent un **grand** sous-ensemble plus abstrait (missions globales, concepts cognitifs, grandes équipes, etc.). La **hiérarchie** (micro → méso → macro) ainsi constituée fournit une ossature pour un **DSL** multi-niveau, permettant de résoudre la complexité en *segmentant* l’auto-organisation et en facilitant des **interactions** ascendantes et descendantes entre différents paliers. La suite du chapitre (Section 6.2.2, etc.) détaillera comment formaliser l'**agrégation** et la **coordination** de ces niveaux, et comment les *synergies* ω peuvent se propager ou se combiner à travers la hiérarchie pour former un **SCN** véritablement multi-échelle.

6.2.2. Théorie des Systèmes Emboîtés

Lorsque l’on manipule des **SCN** (Synergistic Connection Networks) appelés à se structurer en **niveaux** (micro, intermédiaire, macro), il est souvent utile d’invoquer la **théorie des systèmes emboîtés**. Cette théorie, inspirée de modèles biologiques ou complexes, décrit comment des **entités de base** (cellules, micro-clusters) se regroupent en « tissus » (super-nœuds de niveau intermédiaire), puis en « organes » (macro-nœuds plus vastes), etc. La notion d'**emboîtement** formalise la hiérarchie progressive aboutissant, in fine, à un **organisme** global (un niveau macro englobant la plupart ou la totalité des entités).

6.2.2.1. Systèmes emboîtés : cellule → tissu → organe → organisme (analogie)

L’idée de **multi-échelle** et d'**emboîtement** dans un **SCN** (Synergistic Connection Network) trouve un écho naturel dans la **biologie**, où l’on décrit une hiérarchie Cellule → Tissu → Organe → Organisme. Chaque palier agrège des entités de taille ou de complexité plus faible pour former un ensemble supérieur doté de nouvelles fonctions. Cette

analogie éclaire comment, dans un **SCN** multi-échelle, on peut progressivement construire, de bas en haut, des **niveaux** de plus en plus **macro** tout en maintenant un lien avec les **interactions** plus fines du niveau inférieur.

1. Analogie Biologique

Il est courant, en biologie, de formaliser l'échelle **cellule** comme l'unité de base, dont la coordination et l'agrégation forment un **tissu** spécialisé (musculaire, nerveux, épithelial, etc.). Plusieurs tissus interdépendants constituent alors un **organe**, et la conjonction d'organes distincts mais coopérants donne naissance à un **organisme** entier. Sur le plan **hiérarchique**, on peut poser :

$$\text{Cellule} \rightarrow \text{Tissu} \rightarrow \text{Organe} \rightarrow \text{Organisme}.$$

Chacune de ces étapes se caractérise par l'**agrégation** ou l'**encapsulation** des entités de l'étape précédente, ainsi que par de **nouvelles** propriétés émergentes (régulation hormonale, vascularisation, etc.). Cet enchaînement incarne un **système emboité** : un tissu contient des cellules, un organe contient des tissus, etc.

2. Formalisation en Systèmes Emboités

D'un point de vue **mathématique**, on peut représenter la hiérarchie cellulaire–tissulaire–organique par des **sous-ensembles** successifs $\{\mathcal{E}_i^{(k)}\}$ où k identifie le niveau. Par exemple :

- Niveau $k = 0$: $\mathcal{E}_i^{(0)}$ correspond aux **cellules** (entités de base).
- Niveau $k = 1$: on agrège ces cellules en **tissus** $\mathcal{T}_\alpha^{(1)}$.
- Niveau $k = 2$: on réunit plusieurs tissus en **organes** $\mathcal{O}_\beta^{(2)}$.
- Niveau $k = 3$: la **somme** des organes interdépendants constitue un **organisme** $\mathcal{O}^{(3)}$.

Chaque niveau s'**emboîte** littéralement dans le suivant : les éléments de niveau k se rassemblent pour former les entités du niveau $k + 1$. Sur le plan formel, on peut définir une **application** Φ_k :

$$\Phi_k: \{\mathcal{E}_i^{(k)}\} \rightarrow \{\mathcal{E}_\alpha^{(k+1)}\},$$

qui regroupe (ou *coarse-grain*) les entités $\mathcal{E}_i^{(k)}$ en super-nœuds $\mathcal{E}_\alpha^{(k+1)}$. Cela **reproduit** la logique de passage de la cellule au tissu, puis du tissu à l'organe, etc.

3. Maintien des Interactions

Dans la biologie, on sait que les **tissus** n'annulent pas les interactions **locales** entre cellules ; elles se reflètent sous la forme de **communications** (signaux chimiques, électriques, etc.) à un **niveau supérieur**. Pareillement, un organe dépend de la cohérence entre ses tissus, et l'organisme de la synergie entre ses organes.

Sur le plan **mathématique** et plus spécifiquement dans un **SCN** multi-niveau, cette interdépendance se traduit :

349.Par l'existence de **fonctions d'agrégation** (Section 6.2.2.2) qui synthétisent les liaisons ω d'un niveau vers le suivant.

350.Par des **rétroactions** top-down (Section 6.2.3.1) dans lesquelles l'état macro peut influencer les pondérations micro.

La structure hiérarchique ainsi formée n'annule pas la réalité des interactions locales, mais les reflète dans des *super-pondérations* $\omega^{(k)}$ entre macro-nœuds, tout en laissant la possibilité que le niveau inférieur conserve sa propre dynamique DSL.

Exemple : cellule → tissu → organe → organisme

Prenons un organisme simple, par exemple un invertébré. Les **cellules** (niveau 0) se spécialisent et se coordonnent pour bâtir un **tissu** musculaire (niveau 1). À ce stade, Φ_0 associe un ensemble de cellules “musculaires” à un super-nœud “tissu musculaire”. Parallèlement, d’autres cellules deviennent un “tissu nerveux”. Au niveau 2, on réunit ces tissus en un organe (muscle + nerf + vaisseaux) via l’application Φ_1 . Et au sommet, l’**organisme** complet (niveau 3) résulte de la **cohésion** entre organes.

Formellement, si on note Γ_k la fonction de regroupement de niveau k à $k+1$, on a :

$$\Gamma_0(\{\text{cellules}\}) = \{\text{tissus}\}, \quad \Gamma_1(\{\text{tissus}\}) = \{\text{organes}\}, \quad \Gamma_2(\{\text{organes}\}) = \{\text{organisme}\}.$$

C’est exactement cette **progression hiérarchique** que l’on veut transposer dans le **DSL** multi-échelle : passer d’entités fines (micro-clusters) à des super-nœuds (mésos) puis à des macro-entités globales (macro-niveau), tout en conservant la mémoire des interactions locales ou intermédiaires.

Conclusion

La **théorie des systèmes emboîtés**, illustrée par l’analogie de la **biologie** (cellule → tissu → organe → organisme), propose un **cadre** pour comprendre comment des entités de niveau inférieur se regroupent ou se coordonnent afin de produire, à l’échelle supérieure, des entités plus massives et plus fonctionnellement riches. D’un point de vue **multi-échelle** dans un **SCN**, cette perspective confirme la pertinence de **former** et **assembler** des micro-clusters en super-nœuds, puis de répliquer ce processus jusqu’à atteindre un macro-niveau. Les interactions ω ne disparaissent pas, elles se reconstituent à un **nouveau palier** via des mécanismes d’**agrégation** ou de **coarse-graining** (Section 6.2.2.2) et peuvent être **rétroactionnelles** (top-down vs. bottom-up) (Section 6.2.3.1). Le tout vise à une **auto-organisation** hiérarchisée, plus “naturelle” et plus robuste, dans la lignée des principes biologiques qui inspirent souvent le DSL.

6.2.2.2. Transposition au DSL : agrégation progressive d’entités vers des super-nœuds (macro-nœuds)

Le passage de **niveaux** inférieurs (micro) à des **super-nœuds** plus abstraits (macro-nœuds) est un élément clef lorsqu’on applique l’idée de **systèmes emboîtés** (6.2.2.1) dans un **SCN** (Synergistic Connection Network) reposant sur le **DSL** (Deep Synergy Learning). Il s’agit de formaliser, dans un cadre mathématique et algorithmique, la manière dont un *ensemble* d’entités localement cohésives (k -ième niveau) se **fusionnent** pour former des **super-nœuds** au niveau $k+1$. Cette section décrit les **fondements** de ce mécanisme d’**agrégation progressive**, en expliquant comment on repère les clusters, comment on “condense” leurs pondérations, et de quelle manière la **dynamique** DSL peut se poursuivre à un niveau supérieur.

A. Notation Générale : Agrégation de Clusters

1. Ensemble initial (niveau 0)

On part d’un **niveau 0** composé d’entités “microscopiques” $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$, qui interagissent via une matrice $\omega^{(0)}$. La mise à jour se fait suivant la **règle DSL** :

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_0 [S_0(i,j) - \tau_0 \omega_{i,j}^{(0)}(t)].$$

Chaque $\omega_{i,j}^{(0)}$ peut s’interpréter comme la synergie (ou le degré de corrélation, ou de compatibilité) entre deux entités **basiques** du réseau.

2. Détection ou formation de clusters

Lorsque la **dynamique** DSL a suffisamment évolué (ou de manière continue), on identifie un **ensemble** de *micro-clusters* $\{\mathcal{C}_1^{(0)}, \dots, \mathcal{C}_r^{(0)}\} \subseteq \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. En pratique, on repère ces clusters à l’aide :

- D’un **seuil** sur $\omega_{i,j}^{(0)}$: si $\omega_{i,j} \geq \theta$, on les considère comme reliés.

- D'un **algorithme** de détection de communautés (modulaire, composantes fortement connectées).
- D'une **analyse** topologique (loi des distances, etc.) ou d'une **fonction** $\Omega(\mathcal{C}) = \sum_{i,j \in \mathcal{C}} \omega_{i,j}$.

Dès lors, chaque micro-cluster $\mathcal{C}_\alpha^{(0)}$ regroupe un certain nombre d'entités très cohésives. Ce sont ces regroupements que l'on va "remplacer" par des super-nœuds pour construire le niveau 1.

3. Agrégation en super-nœuds

Concrètement, on "condense" chaque micro-cluster $\mathcal{C}_\alpha^{(0)}$ en un **super-nœud** $\mathcal{N}_\alpha^{(1)}$. Ainsi, le *niveau 1* du SCN se compose :

$$\{\mathcal{N}_1^{(1)}, \dots, \mathcal{N}_r^{(1)}\}.$$

On parle alors d'une application de **coarse-graining** ou de "macro-agrégation" Γ_0 , telle que :

$$\Gamma_0: \{\mathcal{C}_\alpha^{(0)}\} \rightarrow \{\mathcal{N}_\alpha^{(1)}\}.$$

Les *indices* α, β marquent que chaque super-nœud du niveau 1 correspond à un *ensemble* de micro-entités du niveau 0. La question cruciale devient alors : **comment** définir les pondérations $\omega_{\alpha,\beta}^{(1)}$ entre super-nœuds $\mathcal{N}_\alpha^{(1)}$ et $\mathcal{N}_\beta^{(1)}$?

B. Définition des Pondérations entre Super-nœuds

1. Fonction d'agrégation Ψ

Pour **transposer** les pondérations $\omega_{i,j}^{(0)}$ au niveau 1, on se sert d'une **fonction** Ψ . Par exemple, une **moyenne** :

$$\omega_{\alpha,\beta}^{(1)} = \frac{1}{|\mathcal{C}_\alpha^{(0)}| \cdot |\mathcal{C}_\beta^{(0)}|} \sum_{i \in \mathcal{C}_\alpha^{(0)}, j \in \mathcal{C}_\beta^{(0)}} \omega_{i,j}^{(0)}.$$

Ou une **somme** ou un **maximum**. L'idée est de condenser les multiples liaisons $\omega_{i,j}^{(0)}$ internes à α (ou entre α et β) en une unique $\omega_{\alpha,\beta}^{(1)}$. Ce nouveau "réseau" (niveau 1) présente alors beaucoup moins de nœuds que le réseau de base.

2. Règle DSL au niveau 1

Une fois $\omega_{\alpha,\beta}^{(1)}$ initialisées par Ψ , on peut **appliquer** à ce niveau la même **dynamique** DSL (ou une version paramétrée différemment) :

$$\omega_{\alpha,\beta}^{(1)}(t+1) = \omega_{\alpha,\beta}^{(1)}(t) + \eta_1 [S_1(\alpha, \beta) - \tau_1 \omega_{\alpha,\beta}^{(1)}(t)],$$

afin que les super-nœuds α, β interagissent *entre eux* selon une logique similaire. De cette manière, un "**cluster de super-nœuds**" peut encore se former au niveau 1, ce qui **prépare** l'étape suivante (passage à un niveau 2).

C. Répétition jusqu'au Macro-Niveau

1. Itération du Processus

Si le **niveau 1** engendre lui aussi des clusters $\mathcal{C}_\gamma^{(1)} \subseteq \{\mathcal{N}_\alpha^{(1)}\}$, on **répète** : on agrège ces sous-groupes en **macro-nœuds** $\mathcal{N}_\gamma^{(2)}$. On définit $\omega_{\gamma,\delta}^{(2)}$ en "relisant" $\omega_{\alpha,\beta}^{(1)}$ via une fonction Ψ . On obtient ainsi :

$$\Gamma_1: \{\mathcal{C}_\gamma^{(1)}\} \rightarrow \{\mathcal{N}_\gamma^{(2)}\}.$$

Ce mécanisme se **poursuit** jusqu'à un certain **niveau K**, où l'on considère avoir atteint un **macro-nœud** quasi final (ou un petit nombre de macro-nœuds dominants). Chaque itération réduit la granularité et crée une **hiérarchie**.

2. Auto-similarité potentielle

Si, à chaque niveau, on utilise la *même forme* de la **règle DSL** et la *même fonction* Ψ , on peut constater une **auto-similarité** (cf. 6.3) : la manière dont les clusters se forment localement se **réplique** à l'échelle suivante. Cela ouvre la voie à une **fractalité** (lois de puissance, motifs récurrents) d'où l'intérêt d'étudier ce **multiniveau** dans la théorie du DSL.

3. Avantages

- 351. **Réduction de complexité** : on remplace un ensemble énorme de micro-pondérations $\omega_{i,j}^{(0)}$ par un plus petit ensemble $\omega_{\alpha,\beta}^{(1)}$.
- 352. **Modularité** : chaque niveau peut “gérer” ses pondérations, sans interférer constamment avec le niveau inférieur.
- 353. **Faculté de scale** : on peut potentiellement pousser la hiérarchie sur 2, 3... niveaux si le nombre n de micro-entités est très grand.

D. Mise en Œuvre Algorithmique

1. Processus d'Agrégation

Typiquement, un **algorithme** multi-échelle fonctionnant par itérations “aggrégation–mise à jour” :

- 354. **Clusterisation** (niveau k) : On repère les sous-ensembles $\mathcal{C}_\alpha^{(k)}$.
- 355. **Agrégation** : On crée des super-nœuds $\mathcal{N}_\alpha^{(k+1)}$ et initialise $\omega_{\alpha,\beta}^{(k+1)}$ via Ψ .
- 356. **Mise à jour** : On applique la règle DSL au niveau $k + 1$, construisant la structure $\omega^{(k+1)}$.
- 357. **Répétition** : Tant que $\omega^{(k+1)}$ suscite de nouvelles agrégations, on recommence (sinon on s'arrête).

2. Pseudo-code

```
function buildHierarchy(level=0, listOfEntities E^(0)):  
    # 1) Repérer clusters micro  
    clusters = detectClusters(E^(0), w^(0))  
    # 2) Construire super-nœuds  
    superNodes = []  
    for each cluster C in clusters:  
        superNode M = new SuperNode(C)  
        superNodes.add(M)  
  
    # 3) Agréger pondérations  
    w^(level+1) = computeWeights(superNodes, w^(level)) # via Psi  
  
    # 4) Mise à jour DSL  
    applyDSL(w^(level+1))  
  
    # 5) S'il reste de la structure interne  
    if new merges can appear:  
        buildHierarchy(level+1, superNodes)  
    else:  
        return superNodes
```

Cet algorithme illustre un schéma **ascendant** (bottom-up). Des variantes peuvent incorporer des **feedback top-down** (section 6.2.3).

Conclusion

Le passage d'un **niveau** micro (ou intermédiaire) à un **niveau** supérieur (super-nœuds, macro-nœuds) dans un **SCN** repose sur une **agrégation progressive** de clusters identifiés, de manière à construire peu à peu une **hiérarchie** d'entités emboîtées. **Mathématiquement**, on définit des **fonctions** Ψ pour "transposer" les pondérations d'un niveau au suivant, ainsi qu'une *règle DSL* adaptant le renforcement/inhibition à ces super-nœuds. **Algorithmiquement**, on réalise un processus *ascendant* : on détecte des clusters cohésifs, on les condense en un super-nœud, on recalcule la synergie au niveau agrégé, et on réitère jusqu'à aboutir à un macro-niveau plus stable.

Ce mécanisme formalise la notion d'un **SCN** multi-échelle de type "cellule → tissu → organe → organisme" (ou "micro-cluster → super-nœud → macro-nœud"). Il **prépare** à la discussion (6.2.2.3) sur la **coordination** multi-niveau, où l'on couplera l'agrégation bottom-up à des *rétroactions* top-down, assurant une auto-organisation fluide à tous les paliers de la hiérarchie.

6.2.2.3. Multi-niveau comme "pilier" pour la gestion de la complexité

Lorsque la taille n d'un **SCN** (Synergistic Connection Network) devient très grande (dizaines, centaines de milliers ou plus d'entités), on se retrouve face à une croissance potentielle de la complexité en $O(n^2)$ si l'on s'en tient à un **niveau unique**. Une telle situation rend difficile la mise à jour de la matrice ω et la détection de clusters significatifs. Le **multi-niveau** (ou multi-échelle), décrit dans les sections précédentes, se révèle alors un **pilier** incontournable pour contenir cette complexité et l'organiser de manière **hiérarchique**.

A. Réduction de la complexité via l'agrégation

Dans un **SCN** de niveau "zéro" (micro) avec n entités, la matrice $\omega^{(0)}$ peut compter jusqu'à $O(n^2)$ pondérations (chaque couple (i, j) peut a priori disposer d'un lien $\omega_{i,j}$). Chaque itération de la règle DSL pourrait, dans le pire cas, nécessiter $O(n^2)$ mises à jour si l'on ne pratique ni **parsimonie** (seuil, kNN, etc.) ni **multi-niveau**.

Lorsque l'on introduit un **niveau supérieur** (voir section 6.2.2.2) qui regroupe les entités en **super-nœuds**, la taille de la matrice $\omega^{(k+1)}$ peut chuter de $O(n^2)$ à $O(m^2)$ pour $m \ll n$. Cela signifie que la charge de gestion des pondérations au niveau $k + 1$ est désormais bien plus réduite. Dans un système hiérarchique allant jusqu'à un macro-niveau, on obtient plusieurs matrices $\omega^{(k)}$ de tailles décroissantes n, n_1, n_2, \dots

Si un système compte un **niveau 0** à n entités, on détecte des **clusters** et on regroupe ces entités en n_1 super-nœuds pour le **niveau 1**. Puis on applique le même principe pour passer de n_1 à n_2 au **niveau 2**, etc., produisant une suite $n_0 = n > n_1 > n_2 > \dots > n_K$. Souvent, $n_K \approx 1$ ou un petit nombre, correspondant à un nombre réduit de macro-nœuds.

Cette **hiérarchie** rend possible un **traitement** plus local au niveau inférieur, alors que le niveau supérieur opère sur moins de super-nœuds, entraînant un coût algorithmique plus faible ($O(n_k^2)$ au niveau k).

Supposons $n_0 = 10^5$ au niveau micro. On pourrait aisément se retrouver avec $O(10^{10})$ liens théoriques. Si on agrège en $n_1 = 10^3$ super-nœuds (niveau 1), on ne gère plus que $O((10^3)^2) = 10^6$ pondérations à ce niveau. En passant à un niveau 2 avec $n_2 \approx 10^2$, la complexité tombe encore à $O((10^2)^2) = 10^4$. Chaque **niveau** manipule donc une structure plus "légère", et l'ensemble forme un **organigramme** multi-niveau de l'auto-organisation.

B. Organisation hiérarchique et contrôle local vs. global

Le **niveau micro** (entités de base, éventuellement groupées en petits micro-clusters) s'occupe des **réactions** les plus fines ou rapides, avec un taux d'apprentissage η_{loc} potentiellement plus élevé. Cela s'explique par la nature **proche** et **réactive** des interactions locales : deux entités très similaires voient leurs liens se renforcer plus vite, sans impliquer l'ensemble du réseau.

L'avantage est que cette adaptation **ne surcarburera** pas la totalité du SCN : seules les entités localement concernées subissent des modifications en temps réel, tandis que les autres restent peu affectées. Cela correspond à un *contrôle local*, bien plus facile à **paralléliser** ou à distribuer.

Au niveau **macro**, ou global, la **structure** du SCN se décrit en termes de super-nœuds plus vastes : on “pilote” alors de *grandes portions* de l’ensemble, éventuellement en $O(m^2)$ liens si l’on ne compte que m macro-nœuds. Cela fournit un **contrôle plus large** mais moins détaillé (il ne faut pas intervenir sur chaque entité de base).

Ce macro-niveau peut fixer des **contraintes** ou envoyer des **feedback** descendant (top-down) pour orienter la dynamique micro, par exemple imposer des seuils minimaux de synergie pour que des entités participent à un macro-objectif.

Cette structuration (micro, méso, macro) évite qu’un **seul** algorithme doive gérer simultanément des centaines de milliers (ou millions) de liens. On crée plutôt **plusieurs** paliers, chacun manipulant un **réseau** de taille moindre, tout en assurant la **cohérence** grâce à la **rétroaction** (section 6.2.3). Sur le plan mathématique, on se dote d’une suite de matrices $\{\omega^{(0)}, \omega^{(1)}, \dots\}$ reliant des entités de *plus en plus agrégées*, et on applique la mise à jour DSL (additive, multiplicative, etc.) à chacun de ces niveaux.

C. Maintien de la cohérence multi-échelle

Le **couplage** multi-échelle se concrétise par des **fonctions d’agrégation** (pour passer du niveau k au niveau $k+1$) et par des **rétroactions** top-down (le niveau macro imposant ou suggérant des ajustements au niveau micro). Chaque matrice $\omega^{(k)}$ conserve alors une **trace** de la structure d’interaction héritée du niveau inférieur, sous forme “coarse-grained”.

On aboutit à un **système** de ponds $\omega^{(k)}(t)$ sur $O(n_k^2)$ liens, avec n_k (le nombre de super-nœuds au niveau k). Un opérateur Ψ_k définit $\omega^{(k+1)}$ à partir de $\omega^{(k)}$. Chaque $\omega^{(k)}$ suit sa propre dynamique DSL :

$$\omega_{\alpha,\beta}^{(k)}(t+1) = \omega_{\alpha,\beta}^{(k)}(t) + \eta_k [S_k(\alpha, \beta) - \tau_k \omega_{\alpha,\beta}^{(k)}(t)].$$

Le multi-niveau permet de restreindre η_k ou τ_k à des valeurs plus appropriées à la taille du niveau. On gère ainsi un **réseau** à $O(n_k^2)$ plutôt qu’à $O(n^2)$, tout en conservant une **vision** hiérarchique plus flexible.

En outre, un **événement** local (un micro-cluster s’effondre, un sabotage local, etc.) n’a pas à se propager instantanément au niveau macro, tant que l’agrégation au niveau supérieur n’y voit pas d’impact majeur. Il y a donc une **protection** top-down, ou un **isolement** partiel, renforçant la robustesse : le macro-niveau n’est pas submergé de fluctuations mineures.

De plus, la **modularité** facilite la maintenance et l’évolution d’un SCN : on peut substituer un bloc micro par un autre sans chambouler tout le macro-niveau.

Conclusion

Le **multi-niveau** ou **multi-échelle** constitue un **pilier** pour **gérer la complexité** dans un SCN volumineux. Sur le plan **mathématique**, on construit une suite hiérarchique de matrices $\omega^{(k)}$ où chaque palier présente moins de nœuds (super-nœuds) et moins de liaisons $O(n_k^2)$. Sur le plan **dynamique**, on répartit l’**auto-organisation** : le niveau local manipule liaisons fines et réactives, le macro-niveau gère de larges ensembles, et un niveau intermédiaire fait la jonction. Ce **scénario** de réduction progressive, combiné à des *opérateurs d’agrégation* et éventuellement à des **feedback** top-down, non seulement abaisse le coût algorithmique, mais renforce la **robustesse** et la **modularité** du SCN.

Ainsi, le **multi-niveau** s’avère un **levier** fondamental pour l’**auto-organisation** à grande échelle, tout en ouvrant la porte à des phénomènes d’**auto-similarité** (voir 6.3) si les mêmes règles DSL et fonctions d’agrégation se répliquent à tous les paliers.

6.2.3. Rétroactions et Anticipations Multi-Niveau

Dans une **organisation** hiérarchique (micro, intermédiaire, macro) — telle qu’exposée en (6.2.1) et (6.2.2) —, il est essentiel de comprendre **comment** ces niveaux interagissent. Les **flux ascendants** (bottom-up) font “remonter” l’information du micro-cluster vers des super-nœuds plus larges (macro-ensembles) ; les **flux descendants** (top-down)

permettent au niveau supérieur d'exercer un **contrôle** (ou une **influence**) sur la dynamique locale. Cette section (6.2.3) en décrit les mécanismes, à commencer par le flux ascendant (6.2.3.1).

6.2.3.1. Flux ascendants (bottom-up) : micro → macro

Dans un **SCN** (Synergistic Connection Network) multi-niveau, le **flux ascendant** (ou bottom-up) représente la façon dont la **structure** et la **dynamique** établies à un **niveau** local (micro) se répercutent vers les **niveaux supérieurs** (macro). Concrètement, c'est l'opération qui permet à des *micro-clusters* ou *super-nœuds* d'être "remontés" et agrégés pour former des macro-nœuds plus vastes. Cette section (6.2.3.1) détaille le **principe**, les **motifs** et les **bénéfices** de ce flux ascendant, en l'illustrant avec des exemples de robotique et de systèmes cognitifs.

A. Principe du flux ascendant

Le flux bottom-up suppose que des **groupes** d'entités (au niveau k) — par exemple un ensemble de micro-entités —, une fois **fortement cohésifs**, vont être **promus** ou **agrégés** en un **super-nœud** au niveau $k + 1$. Cela incarne mathématiquement la **transition** :

$$\mathcal{C}_\alpha^{(k)} \xrightarrow{\text{agrég.}} \mathcal{N}_\alpha^{(k+1)},$$

où $\mathcal{C}_\alpha^{(k)}$ est un cluster de niveau k et $\mathcal{N}_\alpha^{(k+1)}$ le super-nœud correspondant au niveau $k + 1$.

Dans la dynamique DSL, si $\mathcal{C}_\alpha^{(k)} \subset \{\mathcal{E}_i^{(k)}\}$ regroupe un ensemble d'entités ou de nœuds de niveau k , on définit les pondérations $\omega_{\alpha,\beta}^{(k+1)}$ par agrégation des pondérations $\omega_{i,j}^{(k)}$. Généralement, on emploie une fonction Ψ , telle que

$$\omega_{\alpha,\beta}^{(k+1)} = \Psi(\omega_{i,j}^{(k)} \mid i \in \mathcal{C}_\alpha^{(k)}, j \in \mathcal{C}_\beta^{(k)}).$$

L'essence du **flux ascendant** tient à ce que des **clusters** "stables" du niveau inférieur *remontent* et se placent comme **unités** (super-nœuds) au niveau supérieur.

Condition	de	cohésion
Pour qu'un cluster $\mathcal{C}_\alpha^{(k)}$ soit jugé "stable", on impose souvent une condition de type :		

$$\Omega(\mathcal{C}_\alpha^{(k)}) = \sum_{i,j \in \mathcal{C}_\alpha^{(k)}} \omega_{i,j}^{(k)}(t) > \theta_{\text{coh}},$$

où θ_{coh} est un seuil de cohésion. Une fois ce seuil dépassé, on *verrouille* $\mathcal{C}_\alpha^{(k)}$ et on crée un super-nœud $\mathcal{N}_\alpha^{(k+1)}$. C'est ce **verrouillage** qui incarne la décision de "promouvoir" un cluster micro en macro.

B. Motivation et Bénéfices

La **réduction de la complexité** constitue le premier atout majeur : le flux ascendant "filtre" l'information la plus dense ou la plus stable au niveau micro et ne "remonte" que des agrégats cohésifs. Cela dispense le niveau macro de manipuler directement toutes les entités une à une, puisqu'une **compression** ou *coarse-graining* est opérée sur les pondérations ω dans la transition micro → macro. Dans cet esprit, les blocs formés à l'échelle inférieure servent de briques de construction pour l'étape suivante, allégeant la matrice $\omega^{(k+1)}$.

La **spécialisation locale** se révèle tout aussi fondamentale : au niveau micro, chaque entité ou petit cluster agit avec une **réactivité** importante (par exemple un taux d'apprentissage plus grand η). Cette configuration permet de repérer rapidement une synergie forte. Ce n'est qu'une fois la cohésion atteinte que le cluster ainsi stabilisé est promu au niveau supérieur. En d'autres termes, le "détail" micro demeure géré en local, puis seul le résultat final de l'auto-organisation (un groupe cohésif) se transmet à l'échelle macro, favorisant l'efficacité et la robustesse de la construction hiérarchique.

Un **exemple formel** se produit lorsqu'un cluster $\mathcal{C} \subset \{\mathcal{E}_i^{(k)}\}$ a été identifié. Après détection, on construit un **super-nœud** $\mathcal{N}_{\alpha}^{(k+1)}$. Les pondérations $\omega_{\alpha,\beta}^{(k+1)}$ de ce nœud vis-à-vis d'un autre super-nœud $\mathcal{N}_{\beta}^{(k+1)}$ découlent alors d'une agrégation, typiquement :

$$\omega_{\alpha,\beta}^{(k+1)} = \Psi(\omega_{i,j}^{(k)} \mid i \in \mathcal{C}_{\alpha}^{(k)}, j \in \mathcal{C}_{\beta}^{(k)}),$$

où Ψ combine les poids $\omega_{i,j}^{(k)}$ (moyenne, maximum, minimum, etc.) selon les besoins. De cette manière, le **niveau $k + 1$** ne conserve qu'un nombre restreint de super-nœuds, ce qui allège considérablement la matrice $\omega^{(k+1)}$ et participe à la flexibilité globale du SCN.

C. Exemples Illustratifs

Robotique : micro-clusters

Dans un essaim de robots, le niveau local (micro) peut repérer un **mini-groupe** très coopérant (quelques robots). Ce *micro-cluster* ascendant devient alors un *super-robot* au niveau supérieur, consolidant ses liaisons envers d'autres mini-groupes. On obtient un “**groupe de groupes**” au niveau macro, construit *bottom-up*.

Système cognitif : patterns neuronaux

Dans une architecture cognitive, le micro-niveau (p. ex. des *feature maps* en vision) détecte des **clusters** de neurones (ou filtres) s'activant fortement. Une fois stabilisé, ce cluster “remonte” en tant que *macro-patch* pour le niveau associatif, permettant une **synthèse** plus globale. De cette manière, l'information *micro* (détails sensoriels) s'organise en *proto-représentations* qui deviennent des super-nœuds conceptuels ou symboliques.

Conclusion

Le **flux ascendant (bottom-up)** dans un SCN multi-niveau décrit la **migration** de l'information **micro** vers la **structure macro** : lorsqu'un cluster local se consolide (pondérations élevées), il se voit “promu” en super-nœud, réduisant la taille des matrices à manipuler au niveau supérieur et maintenant la **cohérence** de l'**auto-organisation**. Sur le plan **mathématique**, cela se formalise via une **fonction** Ψ d'agrégation, et un **critère** de cohésion $\Omega(\mathcal{C})$. Sur le plan **opérationnel**, ce mécanisme gère la **réactivité** fine en bas (micro) et la **vision** plus large en haut (macro), tout en **diminuant** la complexité par un principe de *filtrage* et de *coarse-graining*. La **section 6.2.3.2** abordera la **complémentarité** descendante (top-down), où le macro-niveau peut à son tour influencer ou “redescendre” sur le micro-niveau pour ajuster les pondérations.

6.2.3.2. Flux descendants (top-down) : macro → micro

Après avoir vu en section 6.2.3.1 comment les *micro-clusters* (ou ensembles locaux) “remontent” vers les niveaux supérieurs via le **flux ascendant**, il est tout aussi fondamental de comprendre le **flux descendant** (top-down). Celui-ci décrit la manière dont le **niveau macro** (ou intermédiaire, s'il est plus élevé) peut rétroagir et exercer une influence directe sur les **pondérations** ou la **dynamique** au niveau micro. Ce mécanisme top-down assure une **cohérence** globale du SCN (Synergistic Connection Network) en permettant aux intentions, visions ou constats du niveau supérieur de se **répercuter** localement dans les liens $\omega_{i,j}$.

A. Rôle du niveau macro vis-à-vis du niveau micro

Au **niveau macro**, on manipule un plus petit nombre de *super-nœuds* $\{\mathcal{M}_{\alpha}^{(k)}\}$, chacun agrégé depuis un niveau inférieur. Ce palier peut détecter une structure globale, par exemple l'émergence de deux **macro-clusters** se rapprochant ou la nécessité de les fusionner. Pour que cette **fusion** s'opère effectivement en bas (niveau micro), il peut être **nécessaire** que le macro-nœud envoie un signal top-down demandant d'**augmenter** certaines liaisons $\omega_{i,j}^{(\text{micro})}$ ou d'en **diminuer** d'autres, en fonction d'un objectif général.

D'un point de vue **mathématique**, on modélise la mise à jour au **niveau micro** par une équation augmentée :

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_0 [S_0(i,j) - \tau_0 \omega_{i,j}^{(0)}(t)] + \gamma_{\text{top-down}} h_{i,j}^{\text{(macro)}}(t),$$

où le terme $h_{i,j}^{\text{(macro)}}(t)$ encode l'**influence** descendante du niveau macro (par exemple, un **boost** si on souhaite rapprocher deux entités i,j appartenant à des macro-nœuds censés fusionner).

Imaginons un macro-niveau $\omega^{(k)}$ détectant que deux grands ensembles $\mathcal{N}_\alpha^{(k)}$ et $\mathcal{N}_\beta^{(k)}$ s'avèrent synergiques, et qu'il est donc logique de les **fusionner**. Cependant, la **réalité** micro peut comporter des liens $\omega_{i,j}^{(0)}$ (pour $i \in \mathcal{N}_\alpha^{(k)}, j \in \mathcal{N}_\beta^{(k)}$) trop faibles. Le macro-niveau envoie alors un message top-down pour **accroître** ces $\omega_{i,j}^{(0)}$, guidant la dynamique micro vers un regroupement effectif.

Inversement, si un conflit est détecté, un message top-down peut **abaisser** certaines synergies locales, évitant des ambiguïtés ou chevauchements contradictoires.

Sans ce flux descendant, la **logique** de chaque niveau local risquerait d'être **myope**, focalisée sur de petites agrégations sans percevoir l'intérêt d'une fusion plus large. Le **macro-niveau** peut imposer ou suggérer des orientations qui, sinon, ne surgiraient pas localement. Cela donne une **harmonisation** entre la réactivité fine en bas et la vision globale en haut.

B. Forme générale du flux descendant

On peut définir, pour chaque niveau k , une fonction ou un opérateur Λ_k qui transforme l'**état** $\omega^{(k)}(t)$ du réseau macro (ses liens, ses clusters) en un **terme** de correction $\Delta^{(k-1)}$ pour le niveau $k-1$. Concrètement :

$$\Lambda_k: \omega^{(k)}(t) \mapsto \Delta_{i,j}^{(k-1)}(t),$$

et on rajoute $\Delta_{i,j}^{(k-1)}(t)$ dans l'équation de mise à jour $\omega_{i,j}^{(k-1)}$. Cela rend possible des **ajustements** fins : augmenter/diminuer $\omega_{i,j}$ localement, changer la saturation, etc.

Dans la pratique, cette **rétroaction** top-down se concrétise par :

358. Un **boost** sur les liens $\omega_{i,j}^{(0)}$ considérés "importants" pour un but global (ex. fusion).

359. Un **découragement** (pénalité) sur d'autres liens si le macro-niveau détecte un conflit ou une "incompatibilité".

On peut programmer un **module** top-down qui, chaque X itérations, regarde $\omega^{(k)}$ et en déduit des $\Delta_{i,j}^{(k-1)}$ à appliquer. Ainsi, on obtient :

$$\omega_{i,j}^{(k-1)}(t+1) = \omega_{i,j}^{(k-1)}(t) + \Delta_{i,j}^{(k-1)}(t),$$

où $\Delta_{i,j}$ est *calculé* en fonction du macro-niveau.

Exemple

Soit un robot multi-niveau :

- Niveau **micro** : quelques robots $\{r_1, r_2, \dots\}$.
- Niveau **macro** : un "essaim" global défini par $\omega_{\alpha,\beta}^{(k)}$.

Si le macro-niveau souhaite que deux sous-essaims se coordonnent, il affecte une valeur positive Δ_{r_i,r_j} pour les robots r_i, r_j situés dans deux ensembles censés fusionner. Au cycle suivant, $\omega_{i,j}^{(0)}$ au niveau micro s'en voit **augmentée**, favorisant la coopération.

C. Finalité : Couplage Macro → Micro pour une cohérence globale

Le macro-niveau peut “voir” des **corrélations** ou **objectifs** plus vastes. S’il veut encourager une convergence entre macro-nœuds, mais remarque des liens micro trop faibles, il agit en top-down pour relever ces liens. Ce faisant, il “induit” de nouvelles synergies locales.

Le flux ascendant (micro → macro) consolide ce qui s’est stabilisé au niveau local. Le flux descendant (macro → micro) renforce ou corrige ce qui serait souhaitable pour la vision d’ensemble. Ensemble, ils forment un **cercle de rétroaction** : l’information monte, le macro la synthétise, puis la redescend sous forme de consignes. Cela rend la **dynamique** multi-niveau potentiellement plus stable ou plus rapide (moins d’essais-erreurs purement locaux).

En dernier lieu, ce mécanisme empêche l’**émiettement** excessif d’un SCN en multiples micro-groupes indépendants. Dès que le macro-niveau perçoit un intérêt commun, il enjoint aux entités de se rapprocher. Sur le plan **mathématique**, cela se formalise par des *termes d’influence* $\Delta_{i,j}$ insérés dans l’équation micro. Sur le plan **pratique**, c’est une manière de *diriger* ou *guider* localement l’auto-organisation vers une **harmonie** plus globale.

Conclusion

Le **flux descendant** (top-down) est la **contrepartie** naturelle du flux ascendant (bottom-up). Tandis que le **bottom-up** agrège les micro-clusters en super-nœuds, le **top-down** réinjecte dans la matrice micro les *connaissances* ou *objectifs* du macro-niveau. **Mathématiquement**, cela se matérialise par un **terme** additionnel dans la mise à jour $\omega_{i,j}^{(0)}$, déterminé par l’état $\omega^{(k)}$ ou $\omega^{(\text{macro})}$. Ainsi, le **SCN** hiérarchique profite d’une boucle multi-niveau complète :

- 360.Les micro-entités forment localement des clusters,
- 361.Le macro-niveau agrège ces clusters et y détecte un but ou une structure,
- 362.Il redescend un signal ajustant des liaisons micro pour mieux refléter l’objectif global.

Ce mécanisme **renforce la cohérence** et la **capacité adaptative** d’un SCN complexe, permettant une **auto-organisation** à la fois fine et globale.

6.2.3.3. Communication entre niveaux synergiques et stabilisation

L’instauration d’un **SCN** (Synergistic Connection Network) à multiples paliers — micro, intermédiaire, macro — repose sur l’existence d’une communication **verticale** entre ces niveaux, permettant la **rétroaction** complète : le **flux ascendant** (bottom-up) et le **flux descendant** (top-down). Cette section illustre la **coexistence** des deux sens de circulation et décrit de quelle manière ils aboutissent à une **stabilisation** globale, où chaque niveau coopère avec les autres pour garantir la cohérence d’ensemble du réseau.

A. Schéma Général de Communication

On suppose un **système hiérarchique** à K paliers, indexés par $k = 0, 1, \dots, K$.

- Le niveau k dispose d’un ensemble de nœuds (entités ou super-nœuds) et d’une matrice $\omega^{(k)}$.
- Le passage $\omega^{(k)} \rightarrow \omega^{(k+1)}$ constitue un **flux ascendant** (bottom-up), noté Γ_k ou Ψ (6.2.3.1).
- Le passage $\omega^{(k+1)} \rightarrow \Delta^{(k)}$ (une correction appliquée à $\omega^{(k)}$) s’interprète comme un **flux descendant** (top-down), noté Λ_{k+1} .

Cette structure forme un **couplage** entre niveaux :

$$\begin{cases} \omega^{(k+1)}(t) = \Gamma_k(\omega^{(k)}(t)), \\ \omega^{(k)}(t+1) = \omega^{(k)}(t) + \Delta^{(k)}(\omega^{(k+1)}(t)). \end{cases}$$

Dans les faits, la mise à jour $\omega^{(k+1)}(t+1)$ peut aussi dépend du nouvel état $\omega^{(k)}(t+1)$, ou être réalisée en mode **asynchrone** ; l'important est qu'il y ait un aller-retour permanent garantissant la cohésion.

Selon l'**implémentation**, on adopte un protocole :

- **Synchro**ne : après chaque itération au niveau k , on agrège (Γ_k), met à jour $\omega^{(k+1)}$, puis calcule la correction $\Delta^{(k)}$ à renvoyer.
- **Asynchrone** : chaque niveau avance à son rythme, et on synchronise moins fréquemment, créant un *retard* possible entre les informations montantes et descendantes.

Quoi qu'il en soit, le but est de conserver un **échange** soutenu : quand des micro-clusters se forment, le niveau macro en est informé ; quand le niveau macro discerne une organisation plus vaste, il renvoie un signal pour influer sur certaines liaisons microscopiques.

B. Stabilisation Multi-Niveau

Un SCN multi-niveau est dit **stabilisé** s'il existe un *point fixe* (ou un cycle) $\{\omega^{(k)*}\}_{k=0}^K$ tel que :

$$\Gamma_k(\omega^{(k)*}) = \omega^{(k+1)*}, \quad \Delta^{(k)}(\omega^{(k+1)*}) = 0, \quad \forall k.$$

Autrement dit, les agrégations bottom-up Γ_k reproduisent la configuration $\omega^{(k+1)*}$ déjà existante, tandis que les corrections top-down $\Delta^{(k)}$ sont nulles (pas de besoin d'ajustement). Cela signifie que la hiérarchie ne modifie plus ni les clusters ni les liens microscopiques.

La **convergence** vers un tel point fixe peut ne pas être triviale à démontrer. Des conditions de **stabilité** (taux d'apprentissage, seuils de cohésion, etc.) doivent être satisfaites. En pratique, on observe souvent que le SCN multi-niveau se stabilise ou aboutit à un *cycle* (un mouvement périodique restreint), plutôt que de rester en fluctuations incessantes. Dans certains cas, de légères oscillations peuvent persister, reflétant un **équilibre dynamique**.

Il peut se produire qu'un niveau **macro** "veuille" un certain regroupement alors que le niveau **micro** persiste à séparer les entités. Ceci peut générer des **conflits** ou des oscillations : le niveau macro *force* l'augmentation de certains liens, puis le niveau micro *décide* de les réduire, etc. Pour éviter cela, on peut recourir à des *mécanismes de temporisation* ou un *paramétrage* plus modéré ($\gamma_{\text{top-down}}$ pas trop élevé, etc.) assurant la douceur de la rétroaction.

C. Communication et Rétroaction Continue

Il est possible que *chaque* niveau applique la **même** règle DSL (additive, multiplicative, etc.), assurant une sorte d'**auto-similarité** du mécanisme. Mais on peut aussi imaginer des paramètres η_k, τ_k différents pour chaque palier : un niveau micro plus "rapide", un niveau macro plus "lent".

Cette modularité engendre un **système** multi-niveau dans lequel la **fréquence** et la **vitesse** de mise à jour diffèrent, encourageant une stabilisation plus locale en bas, puis une adaptation plus globale au sommet.

En pratique, l'information **monte** par agrégation (6.2.3.1) : on "condense" $\omega^{(k)}$ en $\omega^{(k+1)}$. Puis elle **redescend** par corrections (6.2.3.2), ce qui se matérialise par des **messages** top-down affectant $\omega^{(k)}$. Un système **synchrone** peut gérer ça en "phases" successives, un système **asynchrone** le fait plus irrégulièrement. D'un point de vue **mathématique**, on peut introduire des *délais* δ_k simulant une communication "à la volée".

Le résultat est une **pyramide** ou un "arbre" hiérarchique :

- le niveau 0 (micro) manipule $\omega^{(0)}$ et détecte des clusters,
- le niveau 1 construit $\omega^{(1)}$ en agrégeant,
- etc.

Chaque palier envoie **rétroactions** (top-down) et "résultats" (bottom-up) dans un **cycle** continu. Quand ce cycle se **calme** (ou s'équilibre), on a une **organisation** stable du SCN multi-niveau.

Conclusion

La **communication** entre niveaux synergiques — flux **ascendant** (bottom-up) et flux **descendant** (top-down) — aboutit, dans un **SCN** multi-échelle, à un **état de stabilisation** où chaque palier local ou macro fonctionne de façon cohérente. Cette rétroaction complète (ascendant + descendant) se formalise via :

363. Des **opérateurs** d'agrégation Γ_k (passage $\omega^{(k)} \rightarrow \omega^{(k+1)}$),

364. Des **fonctions** de correction $\Delta^{(k)}$ (descendant $\omega^{(k+1)} \rightarrow \omega^{(k)}$),

permettant un **cercle** de feed-back dans lequel les micro-clusters stabilisés s'insèrent dans des macro-nœuds, et les macro-nœuds influencent en retour les pondérations microscopiques. En combinant ces mécanismes, un **SCN** multi-niveau peut se montrer plus **robuste**, plus **rapide** à converger et plus **capable** d'organiser un très grand nombre d'entités. À ce stade, cette architecture hiérarchique ouvre la voie aux phénomènes d'**auto-similarité** ou de **fractalité** (6.3), dans lesquels le même schéma d'auto-organisation se répète à chaque palier.

6.3. Fractalité et Auto-Similarité dans le DSL

Après avoir exploré la hiérarchie multi-niveau (6.2) et les mécanismes de communication entre paliers (bottom-up, top-down), nous abordons maintenant la **possibilité** qu'un SCN (Synergistic Connection Network) puisse présenter une **fractalité** — c'est-à-dire une forme d'**auto-similarité** à travers différentes échelles. Dans l'histoire des systèmes complexes, la notion de fractales, venue de la géométrie et de la modélisation de phénomènes naturels, s'avère un **paradigme** puissant pour décrire les invariances d'échelle. La section 6.3.1 introduit le concept de fractales en IA, puis 6.3.2 et 6.3.3 détaillent comment cette auto-similarité peut se manifester et se mesurer dans le cadre du DSL, et enfin 6.3.4 discute les avantages et les limites de la fractalité appliquée aux réseaux synergiques.

6.3.1. Concept de Fractales en IA

Les **fractal(e)s** renvoient à des objets ou des systèmes dans lesquels on retrouve, à plusieurs échelles d'observation, une **structure** (ou un "motif") essentiellement identique, éventuellement modulo un facteur d'échelle. C'est cette idée d'**invariance d'échelle** ou d'**auto-similarité** qui caractérise la fractalité, qu'il s'agisse d'ensembles géométriques (courbe de Koch, ensemble de Cantor, etc.) ou de phénomènes physiques et biologiques (réseaux vasculaires, littoraux, ramifications dans un arbre, etc.).

6.3.1.1. Historique : Fractales dans la Modélisation de Phénomènes Naturels, Auto-similarité à Plusieurs Échelles

La notion de fractale, introduite et popularisée dans les années 1970–1980 notamment par Benoît Mandelbrot, a profondément transformé notre compréhension des phénomènes naturels complexes. En effet, nombre d'objets ou de processus, traditionnellement jugés irréguliers ou chaotiques, révèlent une structure sous-jacente caractérisée par l'**auto-similarité**, c'est-à-dire la répétition de motifs à différentes échelles. Cette propriété permet de modéliser, d'une manière relativement concise, des objets dont la complexité apparente dépasse les approches classiques de la géométrie euclidienne.

A. Émergence du Concept de Fractale

1. Les Travaux Pionniers de Mandelbrot

Au cœur des travaux de Mandelbrot se trouve l'observation que de nombreux phénomènes naturels — comme les côtes marines, les nuages, les réseaux d'agrégation ou même certains aspects des systèmes biologiques — présentent une irrégularité apparente qui, lorsqu'on les observe à des échelles différentes, révèle une structure récurrente. Dans son ouvrage fondateur, *The Fractal Geometry of Nature*, Mandelbrot montre que la mesure de la « longueur » d'une côte, par exemple, dépend de la résolution de l'observation, plus on affine la mesure, plus on découvre de détails, et par conséquent, la longueur calculée augmente de manière non linéaire. Cette propriété a conduit à la conceptualisation d'objets dont la dimension, dans le sens de Hausdorff ou de Minkowski, est non entière. Pour le flocon de Koch, l'un des exemples classiques de fractale, la dimension fractale est donnée par :

$$\dim_H(\text{Flocon de Koch}) = \frac{\ln 4}{\ln 3} \approx 1.2619,$$

ce qui indique que cet objet, bien que contenu dans le plan, possède une complexité qui dépasse la simple dimension linéaire.

2. Auto-similarité et Invariance d'Échelle

La propriété d'**auto-similarité** est l'un des piliers de la géométrie fractale. Un objet est dit **auto-similaire** s'il se recoupe avec lui-même à différentes échelles de grandeur, c'est-à-dire que si l'on effectue un zoom sur une portion de l'objet, cette portion présente la même structure (à un facteur d'échelle près) que l'ensemble. On peut exprimer cette propriété à l'aide de similitudes. Soit $F \subset \mathbb{R}^d$ un ensemble fractal, alors il existe un nombre fini de fonctions contractantes $\{f_1, f_2, \dots, f_n\}$ telles que :

$$F = \bigcup_{i=1}^n f_i(F).$$

Ce formalisme, qui fait partie de la théorie des systèmes itératifs par fonction (IFS), traduit l'invariance d'échelle propre aux fractales. Par ailleurs, de nombreux phénomènes naturels se caractérisent par des lois de puissance, telles que :

$$P(X > x) \sim x^{-\alpha},$$

ce qui illustre que la distribution de certaines quantités (taille des amas, fréquence des phénomènes sismiques, etc.) reste identique lorsqu'on change d'échelle d'observation.

B. Exemples de Phénomènes Naturels

1. Côtes Maritimes et Littoraux

L'un des exemples les plus emblématiques est celui de la mesure des côtes. Lorsque l'on mesure la longueur d'un littoral, la valeur obtenue dépend de la précision de la règle utilisée. Si l'on se limite à une règle de 100 mètres, on obtient une certaine valeur, mais en utilisant une règle de 1 mètre, la longueur mesurée augmente de manière significative. Ce phénomène, illustré par Mandelbrot, montre que les côtes possèdent une dimension fractale comprise entre 1 et 2, reflétant ainsi leur complexité géométrique.

2. Réseaux Vasculaires et Bronchiques

Les systèmes biologiques, tels que les réseaux vasculaires ou bronchiques, exhibent également des propriétés fractales. Dans ces systèmes, les structures se ramifient de manière récurrente : chaque branche se divise en sous-branches selon des ratios constants, ce qui permet d'optimiser la distribution des fluides ou de l'air. La structure auto-similarité assure que l'ensemble du réseau, du plus petit capillaire jusqu'à l'organe complet, obéit à des règles géométriques similaires.

3. Lois de Puissance dans les Agrégations

En physique statistique et en géophysique, on observe fréquemment que la distribution des tailles d'agrégats ou la fréquence des phénomènes (comme les séismes) suit une loi de puissance. Ces lois de puissance traduisent une invariance d'échelle dans l'agrégation des éléments, suggérant que la même dynamique de formation se répète quelle que soit l'échelle d'observation.

C. Fractales et Auto-similarité dans les SCN et l'IA

1. Réseaux Complexes et Dimension Fractale

Les réseaux complexes, qu'ils soient sociaux, biologiques ou informatiques, présentent souvent des distributions de degrés et des structures hiérarchiques qui rappellent les propriétés fractales. Par exemple, la répartition des degrés dans un réseau dit « scale-free » suit typiquement une loi de puissance, indiquant que la structure du réseau est auto-similaire. Cette propriété peut être mise en parallèle avec la dimension fractale des objets géométriques, suggérant que le même type de processus d'agrégation opère à différentes échelles.

2. Auto-organisation dans les SCN

Dans un SCN, la règle d'auto-organisation (par exemple, la mise à jour des pondérations $\omega_{i,j}$ selon la règle DSL) peut se répéter à plusieurs niveaux hiérarchiques. Si la même dynamique s'applique localement dans des clusters et se retrouve également au niveau global, on peut alors qualifier le système de fractal dans le sens où il présente une **auto-similarité** d'une échelle à l'autre. Cette observation permet de modéliser la formation de clusters et d'agrégats dans un SCN par des lois de puissance et d'invariance d'échelle, donnant ainsi une dimension supplémentaire à l'analyse des réseaux auto-organisés dans le domaine de l'IA.

3. Impact et Applications

L'étude des fractales et de l'auto-similarité a des implications profondes pour la modélisation de phénomènes naturels et pour la conception de systèmes d'intelligence artificielle. Dans les SCN, elle permet de mieux comprendre comment des dynamiques locales peuvent se répéter pour générer une organisation globale cohérente. De plus, la notion de

dimension fractale offre un outil quantitatif pour mesurer la complexité d'un réseau et pour évaluer la robustesse de son auto-organisation. Ces concepts trouvent des applications dans des domaines variés, allant de la modélisation des réseaux neuronaux à la prédition des comportements dans les systèmes complexes.

Conclusion

L'historique des fractales, tel qu'introduit dans les années 1970–1980 par Benoît Mandelbrot, a permis de révolutionner la modélisation des phénomènes naturels en introduisant la notion d'**auto-similarité** à plusieurs échelles. Des exemples emblématiques — tels que la mesure des côtes, les réseaux vasculaires ou les lois de puissance dans l'agrégation des phénomènes — illustrent que des objets apparemment irréguliers obéissent en réalité à des règles invariantes par zoom, caractérisées par une dimension fractale non entière. Dans le champ des SCN et de l'intelligence artificielle, ces concepts offrent un cadre théorique pour comprendre comment une dynamique locale de mise à jour (comme celle des pondérations $\omega_{i,j}$) peut se répéter à différentes échelles, aboutissant à des structures auto-organisées et hiérarchisées. Ainsi, la notion de fractalité et d'auto-similarité permet d'appréhender la complexité des réseaux complexes en mettant en évidence que, quelle que soit l'échelle d'observation, les mêmes principes d'agrégation et de répartition s'appliquent. Cette approche ouvre la voie à des applications avancées dans la modélisation de systèmes naturels et artificiels, où la cohérence d'ensemble émerge de l'itération répétée des mêmes règles de formation de clusters.

6.3.1.2. Intérêt pour le DSL : la Réurrence de Patterns Synergiques à Différentes Granularités

L'idée d'auto-similarité, centrale dans l'étude des fractales, se retrouve également dans le cadre du Deep Synergy Learning (DSL) lorsqu'on observe que les mêmes motifs d'organisation apparaissent à plusieurs niveaux de granularité, du micro au macro. Autrement dit, la dynamique qui régit l'évolution locale des pondérations $\omega_{i,j}$ se réplique à différentes échelles, conduisant à une structuration hiérarchique qui se caractérise par l'auto-similarité. Nous développons ci-après, en enrichissant l'argumentation par des éléments mathématiques, comment cette récurrence de patterns synergiques s'exprime et quels en sont les avantages pour un SCN.

A. Notion de Patterns Synergiques et Auto-similarité

1. Définition Mathématique d'un Motif

Considérons un petit graphe ou sous-réseau, noté \mathcal{P} , qui représente un motif synergique typique dans un SCN. Ce motif peut être défini par un ensemble de noeuds $\{1,2,\dots,r\}$ et une matrice de pondérations associée $\omega^{(\text{patt})} = (\omega_{a,b}^{(\text{patt})})_{1 \leq a,b \leq r}$. L'idée fondamentale est que, si l'on peut trouver, dans un grand réseau G (représenté par la matrice ω d'un SCN), un sous-ensemble $\mathcal{C} \subset \{1, \dots, n\}$ qui possède une topologie similaire à celle de \mathcal{P} , alors il existe une application bijective (ou une homothétie approchée)

$$\phi: \mathcal{C} \rightarrow \mathcal{P},$$

telle que pour tout $i,j \in \mathcal{C}$, la relation suivante est approximativement vérifiée :

$$\omega_{i,j} \approx \lambda \omega_{\phi(i),\phi(j)}^{(\text{patt})},$$

où λ est un facteur d'échelle positif. Cette relation exprime formellement l'**auto-similarité**, la structure du motif \mathcal{P} est retrouvée dans le sous-groupe \mathcal{C} à une échelle différente. Ainsi, le même arrangement de liens se répète, ce qui est caractéristique d'un comportement fractal dans un réseau.

2. Dimension Fractale et Couverture en Boîtes

Un autre outil mathématique issu de l'analyse des fractales est la **dimension de Hausdorff** (ou dimension fractale) qui, dans le contexte d'un SCN, peut être interprétée en examinant comment le nombre de sous-groupes distincts évolue en fonction de l'échelle de résolution. Plus précisément, en couvrant le réseau par des "boîtes" de taille ϵ (où ϵ

représente ici une tolérance dans la similarité des valeurs de ω ou dans la distance physique entre les agents), on observe que le nombre de boîtes nécessaires $N(\epsilon)$ satisfait typiquement une relation de type puissance :

$$N(\epsilon) \propto \epsilon^{-\dim_f},$$

où \dim_f est la dimension fractale du réseau. L'invariance d'échelle implique que, quelle que soit la granularité choisie (micro, méso ou macro), la structure du réseau conserve une distribution caractéristique des liaisons, reflétant l'uniformité de la dynamique d'auto-organisation.

B. Récurrence de Motifs via la Dynamique DSL

1. La Règle de Mise à Jour Uniforme

La dynamique du DSL s'exprime typiquement par une règle de mise à jour telle que :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ est la synergie mesurée entre les entités i et j , et η et τ sont des paramètres constants (taux d'apprentissage et coefficient de décroissance, respectivement). Le fait que cette même règle soit appliquée à chaque niveau du réseau signifie que le même opérateur F agit sur les pondérations, que l'on observe au niveau d'un micro-cluster ou d'un macro-cluster.

Autrement dit, l'auto-organisation locale (à l'échelle d'un petit groupe de robots par exemple) se fait selon une mécanique identique à celle du système global. Cela se traduit par la **répétition** de la même forme de dynamique :

$$\omega^{\text{micro}}(t+1) = F(\omega^{\text{micro}}(t), S^{\text{micro}}),$$

$$\omega^{\text{macro}}(t+1) = F(\omega^{\text{macro}}(t), S^{\text{macro}}),$$

avec S^{micro} et S^{macro} étant les scores de synergie évalués à des échelles différentes. Si ces scores sont obtenus par des mécanismes analogues – par exemple via des fonctions de similarité normalisées – alors la **forme** du système (et, en particulier, la distribution des ω) se révèle invariante d'échelle, c'est-à-dire fractale.

2. Lois de Puissance et Distribution de la Taille des Clusters

Une conséquence de l'auto-similarité dans un SCN est que la distribution des tailles de clusters peut suivre une **loi de puissance**. Plus formellement, si l'on note s la taille d'un cluster (le nombre d'entités qui le composent), on peut observer que :

$$P(s) \sim s^{-\alpha},$$

où α est un exposant caractéristique. Ce comportement scale-free indique que la même règle d'agrégation, appliquée à différentes échelles, donne lieu à la formation d'un petit nombre de grands clusters et à une majorité de clusters de taille modeste. Cette propriété fractale assure une **robustesse** du système puisque le mécanisme de formation des clusters n'est pas sensible à l'échelle, mais reste invariant, favorisant ainsi une hiérarchie cohérente d'organisations locales et globales.

3. Fusion et Invariance d'Échelle

Dans le cadre du DSL, le score de synergie $S(i,j)$ est souvent le résultat d'un processus de fusion de divers sous-scores (par exemple, issus de différentes modalités ou d'interactions à différents niveaux). Si l'on définit un score hybride comme :

$$S(i,j) = \alpha S_{\text{sub}}(i,j) + \beta S_{\text{sym}}(i,j),$$

et si ce même mécanisme de fusion est appliqué dans chaque sous-système du SCN, alors la même structure de calcul apparaît quel que soit le niveau d'agrégation. En particulier, si l'on peut identifier un motif \mathcal{P} à une échelle micro tel que

$$\omega_{i,j}^{\text{micro}} \approx \lambda \omega_{\phi(i),\phi(j)}^{(\mathcal{P})},$$

alors, en agrégant ces micro-clusters, le macro-cluster résultant exhibera une structure similaire (avec éventuellement un facteur d'échelle différent). Cela traduit l'invariance d'échelle et renforce l'idée que l'auto-organisation s'effectue de manière fractale.

C. Avantages pour l'Apprentissage Multi-échelle dans le DSL

L'intérêt de la récurrence des patterns synergiques dans un SCN se manifeste sur plusieurs plans :

365. Homogénéité des Mécanismes d'Auto-organisation

Lorsque la même règle d'agrégation-inhibition s'applique à toutes les échelles, le système gagne en simplicité conceptuelle. Un unique mécanisme, qu'on peut exprimer par $F(\omega(t))$, suffit pour régir la dynamique du réseau, de la plus petite unité à l'agrégation globale. Cette homogénéité simplifie l'analyse mathématique et la mise en œuvre algorithmique.

366. Robustesse et Résilience

La présence d'un comportement fractal – où la structure d'un micro-cluster se répète à l'échelle macro – signifie que le système est intrinsèquement résilient. En effet, si une partie du réseau est perturbée ou isolée, la dynamique locale qui reste intacte permet de reconstruire la structure globale par agrégation, sans nécessiter de réinitialisation complète.

367. Lois de Puissance et Détection d'Émergences

L'apparition de lois de puissance dans la distribution des tailles de clusters permet non seulement de quantifier la structure du réseau mais aussi d'identifier des seuils critiques dans l'évolution de l'auto-organisation. Par exemple, si la distribution des tailles de clusters se stabilise selon $P(s) \sim s^{-\alpha}$, alors la valeur de l'exposant α peut servir d'indicateur pour adapter dynamiquement les paramètres du DSL (comme η ou τ) afin de préserver la stabilité du système.

368. Simplicité du Modèle

Du fait que la même dynamique se répète à chaque niveau, le modèle global se réduit à une itération de la même règle. Cela permet de construire des simulations et des théories analytiques qui, en se concentrant sur une échelle donnée, s'appliquent ensuite par récurrence aux autres échelles, facilitant ainsi la compréhension de l'ensemble du processus d'auto-organisation.

Conclusion

La récurrence de patterns synergiques à différentes granularités dans un DSL traduit la présence d'une **auto-similarité** fractale dans la dynamique du réseau. En appliquant la même règle de mise à jour – par exemple, la mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

– à toutes les échelles, on observe que la même structure d'agrégation, d'inhibition et de renforcement des liens se répète, conduisant à la formation de clusters qui sont, à l'échelle micro, similaires à ceux qui se manifestent au niveau macro. Mathématiquement, cela se traduit par l'existence d'une fonction d'homothétie ϕ et d'un facteur d'échelle λ tels que :

$$\omega_{i,j} \approx \lambda \omega_{\phi(i),\phi(j)}^{(\text{patt})}.$$

Cette invariance d'échelle se manifeste également dans la distribution des tailles de clusters, souvent caractérisée par une loi de puissance de la forme $P(s) \sim s^{-\alpha}$. Sur le plan opérationnel, la répétition des mêmes mécanismes d'auto-organisation à différentes échelles confère au DSL une robustesse et une simplicité qui facilitent l'apprentissage multi-échelle. En effet, un même ensemble de paramètres et de règles peut être appliqué pour stabiliser la structure du réseau à la fois localement et globalement, sans avoir à recourir à des modèles ou des algorithmes distincts pour chaque niveau de granularité.

En définitive, la capacité du DSL à reproduire, à travers des itérations successives de sa règle de mise à jour, une structure fractale et auto-similaire, constitue l'un de ses atouts majeurs pour la modélisation de phénomènes complexes, qu'ils soient naturels ou artificiels. Ce comportement fractal, caractérisé par une invariance d'échelle, permet de comprendre comment des dynamiques locales simples peuvent aboutir à une organisation globale riche et hiérarchisée, rendant ainsi le système à la fois robuste, adaptatif et facile à analyser.

6.3.2. Auto-Similarité dans les Réseaux Synergiques

La fractalité, telle que discutée en (6.3.1), trouve un **terrain d'expression** privilégié dans un **SCN** (Synergistic Connection Network) lorsque l'on constate que les *patterns* d'auto-organisation se répètent, à la fois **localement** (dans de petits clusters) et **globalement** (à l'échelle d'un cluster plus vaste), sans nécessiter de règles qualitativement différentes. C'est cette **auto-similarité** à travers les échelles, ou *invariance d'échelle*, qui caractérise un **réseau fractal**.

6.3.2.1. Exemple : Un Cluster Local Reproduit la Même Structure que le Cluster Global (Formes d'Invariance d'Échelle)

Dans le contexte d'un SCN (Synergistic Connection Network), l'auto-organisation repose sur l'évolution locale des pondérations $\omega_{i,j}$ via des règles d'actualisation telles que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ est le score de synergie entre les entités i et j . Une des propriétés remarquables de cette dynamique est la possibilité d'observer des structures similaires à des échelles différentes – un phénomène que l'on qualifie d'**invariance d'échelle** ou d'**auto-similarité**. Autrement dit, un petit cluster (micro-cluster) de robots ou d'agents, qui se forme par des interactions locales fortes, peut présenter une configuration similaire à celle d'un cluster global (macro-cluster) englobant un nombre beaucoup plus important d'entités. Nous détaillons ci-après cette idée en intégrant des arguments mathématiques et en exposant les implications pour le DSL (Deep Synergy Learning).

A. Structure d'un Cluster Local et d'un Cluster Global

Supposons qu'on ait identifié, à l'échelle locale, un cluster \mathcal{C}_{loc} composé de n_{loc} entités qui sont fortement interconnectées. La topologie de ce cluster se caractérise par une distribution interne des pondérations

$$\{\omega_{i,j}^{(\text{loc})} \mid i, j \in \mathcal{C}_{\text{loc}}\},$$

qui présente par exemple une densité élevée, une répartition particulière (cycle, étoile, réseau complet partiel) ou encore une symétrie dans l'organisation des liens.

Au niveau global, on considère un cluster $\mathcal{C}_{\text{glob}}$ de taille beaucoup plus importante, $n_{\text{glob}} \gg n_{\text{loc}}$. L'hypothèse d'auto-similarité consiste à constater que la structure (ou le "motif") de \mathcal{C}_{loc} se retrouve, à un facteur d'échelle près, dans $\mathcal{C}_{\text{glob}}$. En d'autres termes, il existe une application ou un isomorphisme (éventuellement approché)

$$\phi: \mathcal{C}_{\text{glob}} \rightarrow \mathcal{P},$$

où \mathcal{P} représente un motif de référence qui est isomorphe (à un facteur d'échelle près) au motif local. Plus formellement, pour toute paire d'entités $i, j \in \mathcal{C}_{\text{glob}}$, la relation suivante s'exprime :

$$\omega_{\phi(i), \phi(j)}^{(\text{glob})} \approx \lambda \omega_{i,j}^{(\text{loc})},$$

où $\lambda > 0$ est un facteur d'échelle qui permet de “rescaller” la distribution des pondérations du cluster local afin de la faire correspondre à celle du cluster global. Ce résultat traduit l'invariance d'échelle : la topologie (la structure des connexions) se reproduit de manière similaire, malgré la différence d'ordre de grandeur du nombre d'entités.

B. Invariance d'Échelle et Modélisation Mathématique

1. Homothétie et Isomorphisme

Pour exprimer mathématiquement l'auto-similarité, on peut considérer que le motif local \mathcal{P} est obtenu par une transformation homothétique d'un sous-ensemble du cluster global. Soit $\omega^{(\text{loc})}$ la matrice des pondérations dans le cluster local et $\omega^{(\text{glob})}$ celle du cluster global. L'existence d'une fonction de transformation ϕ et d'un facteur λ s'exprime par :

$$\forall i, j \in \mathcal{C}_{\text{loc}}, \quad \omega_{\phi(i), \phi(j)}^{(\text{glob})} = \lambda \omega_{i,j}^{(\text{loc})} + \varepsilon_{i,j},$$

où $\varepsilon_{i,j}$ représente une erreur ou une perturbation qui tend à zéro dans l'idéal. Ce cadre mathématique, proche de la notion d'isomorphisme entre graphes pondérés, permet de caractériser l'auto-similarité par la présence d'une *structure résiliente* et récurrente. En outre, en couvrant le cluster global par des boîtes de taille ϵ (dans le sens de la méthode de box-counting utilisée pour calculer la dimension fractale), on obtient une relation du type :

$$N(\epsilon) \propto \epsilon^{-\dim_f},$$

où $N(\epsilon)$ est le nombre de boîtes nécessaires pour couvrir le cluster global et \dim_f la dimension fractale du réseau. Cette relation illustre que la structure du réseau est invariante par changement d'échelle, caractéristique des systèmes fractals.

2. Loi de Puissance dans la Distribution des Clusters

L'auto-similarité dans un SCN se traduit également par une distribution en loi de puissance des tailles de clusters. Si l'on note $s = |\mathcal{C}|$ la taille d'un cluster, on peut observer empiriquement que :

$$P(s) \sim s^{-\alpha},$$

ce qui signifie que, quelle que soit l'échelle, la probabilité d'obtenir un cluster de taille s décroît selon une loi de puissance. Ce comportement scale-free est une caractéristique essentielle des réseaux fractals, indiquant que le même mécanisme d'agrégation et de séparation s'applique indépendamment de la taille des clusters.

C. Interprétations dans le DSL et Conséquences pour l'Auto-organisation

1. Unicité du Mécanisme d'Auto-organisation

L'application uniforme de la règle DSL, telle que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

indique que la même dynamique opère sur tous les niveaux du réseau. Cette uniformité signifie que le processus d'agrégation (renforcement des liens forts) et d'inhibition (affaiblissement des liens faibles) se répète de manière identique, qu'on observe un petit groupe local ou l'ensemble global des agents. Ainsi, un micro-cluster se forme

localement par la même mécanique que celle qui gouverne la formation d'un macro-cluster. Le fait que la même règle puisse être appliquée de manière récursive dans le réseau est un gage de robustesse et de simplicité du modèle.

2. Avantages pour l'Apprentissage Multi-échelle

L'auto-similarité fractale dans le DSL offre plusieurs avantages majeurs :

- **Simplicité du modèle** : La possibilité de décrire le comportement global du réseau par une seule règle d'auto-organisation permet d'éviter la complexité inhérente à la gestion de multiples mécanismes distincts pour chaque échelle. Un modèle unique s'applique localement et se propage globalement par rescaling.
- **Robustesse** : La redondance des motifs d'organisation, reproduits à différentes échelles, assure que le système reste stable même en cas de perturbations locales. Ainsi, un micro-cluster qui se déstabilise peut être compensé par la stabilité d'un macro-cluster dont la structure est identique.
- **Facilité d'analyse** : Les lois de puissance associées à la distribution des tailles de clusters et la relation homothétique entre les niveaux permettent d'exploiter des outils mathématiques (dimension fractale, box-counting) pour prédire et contrôler l'évolution du réseau.
- **Adaptabilité** : Dans un environnement dynamique, la capacité de reproduire la même structure d'auto-organisation à différentes granularités permet d'adapter le modèle sans recalibrer l'ensemble des paramètres pour chaque niveau d'agrégation.

D. Conclusion

L'exemple d'un cluster local reproduisant la même structure que le cluster global illustre parfaitement le principe d'**invariance d'échelle** dans le cadre du DSL. Mathématiquement, on exprime cette invariance par l'existence d'un facteur d'échelle λ et d'un isomorphisme approché ϕ tel que

$$\omega_{\phi(i), \phi(j)}^{(\text{glob})} \approx \lambda \omega_{i,j}^{(\text{loc})},$$

ce qui signifie que la forme topologique et la distribution des liaisons observées localement se retrouvent à l'échelle globale, à une constante multiplicative près. Cette auto-similarité fractale est particulièrement intéressante pour l'apprentissage multi-échelle, car elle permet d'utiliser une unique règle d'auto-organisation pour stabiliser le comportement à la fois localement et globalement, simplifiant ainsi l'analyse, l'ajustement paramétrique et la robustesse du SCN.

En résumé, lorsque le même motif de liaison se répète à différentes échelles, cela indique que la dynamique sous-jacente (basée sur la synergie et l'inhibition) possède une propriété fractale. Cette invariance d'échelle offre un cadre conceptuel et mathématique puissant pour comprendre comment des interactions simples et locales, appliquées de manière récursive, peuvent donner lieu à une organisation hiérarchique complexe, cohérente et robuste dans un SCN. Cela constitue l'un des piliers théoriques qui justifie l'efficacité du DSL dans la modélisation des systèmes complexes et auto-organisés.

6.3.2.2. Conditions Mathématiques : Si la Distribution des Entités et leurs Synergies Respectent Certaines Lois de Puissance, un Aspect Fractal Peut Émerger

Dans l'étude des systèmes complexes, et plus particulièrement dans le cadre des Synergistic Connection Networks (SCN) appliqués au Deep Synergy Learning (DSL), il apparaît que la dynamique d'auto-organisation peut, sous certaines conditions, donner lieu à un comportement fractal. Ce phénomène se manifeste lorsque la distribution des pondérations $\omega_{i,j}$, la taille des clusters $|\mathcal{C}|$ ou d'autres indicateurs statistiques suivent des lois de puissance. Nous détaillons ici les fondements mathématiques de ces conditions, en insistant sur l'invariance d'échelle et les mécanismes qui conduisent à l'émergence d'un aspect fractal dans un SCN.

A. Notion de Lois de Puissance et d'Invariance d'Échelle

Une loi de puissance se caractérise par l'expression générale

$$\text{Prob}(X > x) \sim x^{-\alpha},$$

pour x suffisamment grand, où α est l'exposant caractéristique de la distribution. Cette relation signifie qu'il n'existe pas d'échelle caractéristique dans la variable X : la probabilité que X dépasse une valeur x décroît selon une fonction en puissance de x . Une propriété clé des lois de puissance est leur invariance d'échelle. En effet, pour tout facteur de redimensionnement $\lambda > 0$, on a :

$$\text{Prob}(\lambda X > x) \approx \lambda^{-\alpha} \text{Prob}(X > x).$$

Ce comportement signifie que, si l'on “zoomé” sur la distribution, la forme de la courbe demeure identique à une constante multiplicative près. Dans le contexte d'un SCN, si la distribution des valeurs de $\omega_{i,j}$, ou la répartition des tailles de clusters, suit une telle loi de puissance, alors le réseau est dit *invariant d'échelle*, une propriété fondamentale pour caractériser la fractalité.

B. Génération de Lois de Puissance par la Dynamique DSL

Dans un SCN, la dynamique d'auto-organisation est régie par une règle de mise à jour du type :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ représente le score de synergie entre les entités i et j , η est le taux d'apprentissage, et τ le coefficient de décroissance. Ce schéma de mise à jour présente plusieurs similitudes avec le mécanisme de renforcement préférentiel (« rich-get-richer ») étudié dans la théorie des réseaux complexes. En effet, lorsqu'un lien est légèrement supérieur à la moyenne, la dynamique tend à le renforcer, tandis que les liens faibles tendent à s'effacer. Ce processus de sélection naturelle des interactions conduit, sous certaines conditions, à l'émergence d'une distribution très hétérogène des pondérations.

Lorsque l'on ne contraint pas artificiellement la distribution (par exemple, par un seuil strict de saturation), la compétition entre les liens peut conduire à une répartition des valeurs de $\omega_{i,j}$ qui suit une loi de puissance. De même, dans l'agrégation hiérarchique des clusters – où les micro-clusters sont regroupés pour former des macro-clusters – si le même mécanisme de renforcement et d'inhibition s'applique à chaque niveau, on observe que la distribution des tailles de clusters, notée par exemple

$$\text{Prob}(|\mathcal{C}| > s) \sim s^{-\alpha},$$

reste invariante d'échelle. Cela signifie que le même schéma statistique apparaît tant à petite qu'à grande échelle, une signature typique d'un comportement fractal.

C. Conditions d'Invariance d'Échelle et Fractalité Statistique

Pour qu'un aspect fractal émerge dans un SCN, il faut que plusieurs conditions soient réunies :

369. Invariance de la Forme de la Distribution

Si l'on considère une variable X (qui peut être, par exemple, la force interne d'un cluster $\Omega(\mathcal{C}) = \sum_{i,j \in \mathcal{C}} \omega_{i,j}$ ou le degré d'un nœud $\deg(i) = \sum_j \omega_{i,j}$), l'invariance d'échelle se traduit par la propriété

$$\text{Prob}(\lambda X > x) \approx \lambda^{-\alpha} \text{Prob}(X > x),$$

ce qui indique que la distribution est une loi de puissance sans échelle caractéristique. Cette propriété doit être observée sur plusieurs ordres de grandeur, c'est-à-dire pour $x \in [x_{\min}, x_{\max}]$.

370. Répétition du Mécanisme d'Auto-organisation

La même règle de mise à jour des pondérations, appliquée de manière itérative, doit générer une dynamique qui se répète à toutes les échelles. Autrement dit, si l'on observe un micro-cluster \mathcal{C}_{loc} de petite taille et un macro-cluster $\mathcal{C}_{\text{glob}}$ beaucoup plus grand, il doit exister une transformation d'homothétie ϕ et un facteur λ tel que :

$$\omega_{\phi(i), \phi(j)}^{(\text{glob})} \approx \lambda \omega_{i,j}^{(\text{loc})} \quad \text{pour } i, j \in \mathcal{C}_{\text{loc}}.$$

Cette relation est au cœur de l'auto-similarité fractale et permet d'affirmer que le même schéma d'interconnexion se reproduit, simplement rescalé, d'un niveau à l'autre.

371. Aggregation et Box-Counting

Pour quantifier l'invariance d'échelle, on peut utiliser la méthode de box-counting. Si l'on recouvre le réseau par des boîtes de taille ϵ , le nombre de boîtes nécessaires $N(\epsilon)$ devrait suivre une loi de la forme :

$$N(\epsilon) \propto \epsilon^{-\dim_f},$$

où \dim_f est la dimension fractale du réseau. La constance de \dim_f à différentes échelles indique que la structure du réseau est auto-similaire.

D. Illustrations Concètes et Implications

Plusieurs indicateurs permettent d'observer ces comportements dans un SCN :

- **Distribution de la Force Interne des Clusters** : La somme des pondérations à l'intérieur d'un cluster, $\Omega(\mathcal{C})$, si elle suit une loi de puissance,

$$\text{Prob}(\Omega(\mathcal{C}) > x) \sim x^{-\alpha},$$
 témoigne d'une invariance d'échelle dans la façon dont les interactions se répartissent au sein des clusters.
- **Distribution des Degrés** : Si le degré d'un nœud, défini par $\deg(i) = \sum_j \omega_{i,j}$, suit une loi de puissance, cela indique que le SCN est *scale-free*, avec quelques nœuds hubs et une majorité de nœuds à faible degré. Ce comportement se retrouve généralement dans les réseaux fractals.
- **Répétition des Motifs** : La présence d'un motif structurel \mathcal{P} dans un micro-cluster, qui se retrouve dans un macro-cluster après application d'une transformation homothétique, confirme l'auto-similarité. Cette observation permet d'affirmer que la même logique d'agrégation et d'inhibition, codifiée par la dynamique DSL, opère de manière récurrente.

Ces constats mathématiques et statistiques confirment que, dans un SCN où la mise à jour des pondérations ne présente pas d'échelle caractéristique, l'auto-organisation produit des structures fractales. L'avantage est double : d'une part, il est possible d'analyser et de prévoir le comportement du réseau à partir d'un sous-ensemble représentatif ; d'autre part, cette invariance d'échelle assure une robustesse et une homogénéité dans l'apprentissage multi-échelle, facilitant ainsi l'extension du modèle à des systèmes de grande taille.

Conclusion

Pour résumer, l'émergence d'un aspect fractal dans un SCN repose sur des conditions mathématiques précises. Si la distribution des pondérations ω , des degrés, ou des forces internes des clusters suit une loi de puissance du type

$$\text{Prob}(X > x) \propto x^{-\alpha},$$

alors l'invariance d'échelle est manifeste. La dynamique DSL, appliquée de manière uniforme à toutes les échelles, engendre des processus de renforcement et d'inhibition qui conduisent à la formation de motifs synergiques identiques, qu'on observe localement ou globalement. Ce phénomène est formellement exprimé par l'existence d'un isomorphisme (ou quasi-isomorphisme) ϕ et d'un facteur d'échelle λ tels que :

$$\omega_{\phi(i),\phi(j)}^{(\text{glob})} \approx \lambda \omega_{i,j}^{(\text{loc})}.$$

La conservation de la même forme statistique sur plusieurs ordres de grandeur – illustrée par le comportement en loi de puissance et par des méthodes telles que le box-counting – démontre que la structure auto-organisée du SCN est fractale. Cet aspect fractal simplifie l'analyse du comportement multi-échelle, renforce la robustesse du système et offre des perspectives pour adapter dynamiquement les paramètres du DSL en fonction de la distribution observée. En d'autres termes, la capacité du DSL à produire des motifs auto-similaires à différentes granularités constitue l'un de ses atouts majeurs pour la modélisation des systèmes complexes, tant naturels qu'artificiels.

6.3.2.3. Mesures Fractales Possibles (Dimension Fractale d'un Graphe, etc.)

L'analyse de la structure fractale dans des systèmes complexes repose sur l'idée d'**invariance d'échelle**, c'est-à-dire que la même forme ou le même comportement se répète à différentes échelles de mesure. Dans le contexte d'un SCN (Synergistic Connection Network), cette approche permet de quantifier l'auto-similarité de la répartition des pondérations $\omega_{i,j}$ et, par extension, de la structure des clusters qui en résultent. Pour ce faire, plusieurs mesures fractales, issues de la géométrie fractale, ont été adaptées à l'étude des graphes pondérés et des réseaux complexes. Nous présentons ici en détail les principales méthodes, leurs fondements mathématiques et leur application au DSL (Deep Synergy Learning).

A. Rappels sur le Box-Counting en Géométrie Fractale

L'une des méthodes les plus classiques pour estimer la **dimension fractale** d'un ensemble $F \subset \mathbb{R}^d$ est la méthode du box-counting. Pour un tel ensemble, on définit $N(\varepsilon)$ comme le nombre minimal de boîtes (ou hypercubes) de côté ε nécessaires pour couvrir l'ensemble F . Si, pour $\varepsilon \rightarrow 0$, on observe que

$$N(\varepsilon) \sim \varepsilon^{-\dim_f},$$

alors la valeur \dim_f est appelée la **dimension fractale** (ou dimension de Minkowski–Bouligand) de l'ensemble F . Plus précisément, on définit

$$\dim_f = \lim_{\varepsilon \rightarrow 0} \frac{\ln N(\varepsilon)}{-\ln \varepsilon},$$

ce qui exprime le fait que la couverture de l'ensemble se fait suivant une loi de puissance. Dans le cas des objets géométriques classiques (ensemble de Cantor, flocon de Koch), cette méthode aboutit à des valeurs non entières, traduisant la complexité inhérente à ces structures.

B. Adaptation du Box-Counting à un SCN ou Graphe Pondéré

Dans un SCN, les entités sont représentées par des noeuds et les interactions par des liaisons pondérées $\omega_{i,j}$. Ces éléments ne sont pas nécessairement disposés dans un espace euclidien de manière canonique ; cependant, on peut définir une **métrique adaptée** à partir des pondérations. Par exemple, on peut définir la distance entre deux noeuds i et j par :

$$d(i,j) = \min_{\gamma(i \rightarrow j)} \sum_{(u,v) \in \gamma} \frac{1}{\omega_{u,v}},$$

où $\gamma(i \rightarrow j)$ désigne un chemin reliant i à j . L'inversion de la pondération permet d'interpréter un lien fort (grand ω) comme une distance courte. À partir de cette métrique, la méthode de box-counting peut être adaptée de la manière suivante :

372. Définition d'une Boîte dans le Graphe

Pour un nœud central c et un paramètre $\ell > 0$, on définit la "boîte" $B(c, \ell)$ comme l'ensemble des nœuds j tels que

$$B(c, \ell) = \{j \mid d(c, j) \leq \ell\}.$$

373. Recouvrement Minimal du Graphe

On cherche ensuite le nombre minimal $N(\ell)$ de tels blocs $B(c, \ell)$ nécessaire pour couvrir l'ensemble des nœuds du graphe. Si la relation

$$N(\ell) \sim \ell^{-\dim_f}$$

est vérifiée pour $\ell \rightarrow 0$, alors le graphe possède une **dimension fractale** \dim_f qui caractérise son auto-similarité. La pente de la courbe obtenue en traçant $\ln N(\ell)$ en fonction de $\ln(1/\ell)$ donne une estimation de \dim_f .

C. Méthodes Alternatives pour Quantifier la Fractalité

1. Dimension de Corrélation

La dimension de corrélation est une autre mesure employée dans l'analyse des attracteurs fractals, notamment via l'algorithme de Grassberger–Procaccia. Dans le contexte d'un SCN, on définit la fonction de corrélation $C(\ell)$ comme la probabilité qu'une paire de nœuds soit à une distance inférieure à ℓ :

$$C(\ell) = \frac{2}{N(N-1)} \sum_{i < j} \chi(d(i, j) \leq \ell),$$

où χ est la fonction indicatrice. Pour ℓ petit, on peut s'attendre à ce que

$$C(\ell) \sim \ell^{\dim_{\text{corr}}},$$

donnant ainsi une **dimension de corrélation** \dim_{corr} qui est souvent proche de la dimension fractale obtenue par box-counting. Cette méthode offre l'avantage de ne pas nécessiter un recouvrement explicite du graphe et s'appuie directement sur les distances entre paires de nœuds.

2. Analyse des Lois de Puissance

L'observation empirique des **lois de puissance** dans la distribution de certaines variables du réseau est également un indicateur d'auto-similarité. Par exemple, si l'on note $P(s)$ la probabilité qu'un cluster ait une taille s et que

$$P(s) \sim s^{-\alpha},$$

alors le réseau présente un comportement **scale-free**. De même, la distribution des degrés $\deg(i) = \sum_j \omega_{i,j}$ peut suivre une loi de puissance, ce qui est typique des réseaux fractals. Ces observations statistiques renforcent l'idée que, quelle que soit l'échelle, la structure du réseau ne présente pas d'échelle caractéristique, mais obéit à une dynamique de type "rich get richer".

D. Implications et Conditions pour l'Émergence de l'Aspect Fractal

Pour qu'un SCN présente un aspect fractal, il faut que la dynamique d'auto-organisation (le processus DSL) n'introduise pas d'échelle artificielle dans la distribution des pondérations. Cela signifie que les paramètres de mise à jour, tels que le taux d'apprentissage η et le coefficient de décroissance τ , doivent être réglés de manière à favoriser un comportement "échelle-libre". Dans un tel régime, les règles d'agrégation (renforcement des liens forts et inhibition

des liens faibles) s'appliquent de manière uniforme et génèrent des distributions de type power law pour des indicateurs tels que :

- La **force interne** d'un cluster, $\Omega(\mathcal{C}) = \sum_{i,j \in \mathcal{C}} \omega_{i,j}$,
- Le **degré** d'un nœud, $\deg(i) = \sum_j \omega_{i,j}$,
- La **taille des clusters**, mesurée en nombre d'entités.

Si, lors de l'agrégation des clusters, on observe que ces distributions restent invariantes à un facteur multiplicatif près, c'est-à-dire que pour une échelle de regroupement donnée, la forme de la distribution reste la même, on peut conclure à une invariance d'échelle dans le SCN.

L'apparition d'un cut-off naturel dans la distribution, c'est-à-dire une borne inférieure x_{\min} et une borne supérieure x_{\max} sur lesquelles la loi de puissance est valide, est également attendue dans des systèmes réels. Cela indique que la fractalité est présente sur plusieurs ordres de grandeur, même si elle n'est pas globale.

Conclusion

Les mesures fractales, telles que la dimension fractale obtenue par la méthode du box-counting ou la dimension de corrélation, fournissent des outils quantitatifs pour évaluer l'auto-similarité dans un SCN. Si la distribution des pondérations $\omega_{i,j}$, la force interne des clusters ou la taille des clusters suit une loi de puissance de la forme

$$\text{Prob}(X > x) \propto x^{-\alpha},$$

alors le système est dit invariant d'échelle. Dans un SCN dont la dynamique est régie par la même règle d'auto-organisation à toutes les échelles, cette invariance se traduit par le fait qu'un micro-cluster reproduit la même structure qu'un macro-cluster, à un facteur d'échelle près. Les conditions mathématiques ainsi établies, basées sur l'analyse des lois de puissance et sur la répartition des éléments du réseau via des méthodes de box-counting ou de corrélation, permettent de caractériser de manière rigoureuse l'aspect fractal d'un SCN. Ce comportement fractal offre un double avantage : il simplifie la modélisation multi-échelle, puisque le même mécanisme peut être appliqué de manière récurrente, et il renforce la robustesse de l'auto-organisation, garantissant ainsi une cohérence structurelle du réseau indépendamment de la taille de ses sous-groupes.

6.3.3. Modélisation et Indicateurs

Pour passer du **concept** d'auto-similarité (6.3.1–6.3.2) à une **quantification** de la fractalité dans un **SCN** (Synergistic Connection Network), on recourt à des **indicateurs** formels. Parmi eux, l'**approche par boîtes** (aussi appelée *box-counting*) est l'une des plus répandues pour estimer la **dimension fractale**. Elle s'adapte assez bien à un réseau (ou graphe), à condition de définir un **critère** de recouvrement en "boîtes" (ou "blocs") pertinent.

6.3.2.3. Mesures Fractales Possibles (Dimension Fractale d'un Graphe, etc.)

Dans cette section, nous exposons de manière détaillée et rigoureuse le cadre mathématique permettant de quantifier l'**auto-similarité** dans un **SCN** (Synergistic Connection Network). L'objectif est de mettre en œuvre des méthodes issues de la géométrie fractale – telles que la méthode du **box-counting**, la dimension de corrélation et l'analyse des lois de puissance – afin de caractériser la structure auto-similaire d'un réseau synergétique. Cette démarche s'inscrit dans le contexte du **Deep Synergy Learning** (DSL) et vise à démontrer que, si la distribution des pondérations ou des tailles de clusters suit une loi de puissance, alors le réseau présente une invariance d'échelle, qui est la marque d'un comportement fractal.

A. Définitions Fondamentales et Cadre Conceptuel

Soit un ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ constituant un SCN, où chaque paire d'entités (i, j) est reliée par une pondération $\omega_{i,j}$ qui reflète le degré de **synergie** entre elles. Pour appliquer une méthode de **box-counting** à ce réseau, il est indispensable de définir une **métrique** sur l'ensemble des nœuds. Une approche naturelle consiste à introduire une fonction κ qui associe à chaque lien une distance « locale » inversée, c'est-à-dire que l'on définit :

$$\kappa(\omega_{i,j}) = \frac{1}{\omega_{i,j}},$$

ce qui traduit l'intuition selon laquelle un lien fort (c'est-à-dire un $\omega_{i,j}$ élevé) correspond à une distance courte entre les entités. Par conséquent, la **distance** entre deux nœuds i et j peut être définie par la longueur du plus court chemin reliant ces deux nœuds :

$$d(i,j) = \min_{\gamma(i \rightarrow j)} \sum_{(\mu,\nu) \in \gamma} \frac{1}{\omega_{\mu,\nu}},$$

où $\gamma(i \rightarrow j)$ désigne un chemin reliant i à j dans le graphe du SCN. Cette distance $d(i,j)$ remplit le rôle de métrique dans notre approche et nous permet de transposer l'idée de couverture par des boîtes dans le contexte d'un graphe.

Pour une échelle donnée $\ell > 0$, nous définissons une **boîte** (ou une boule) centrée sur un nœud c par :

$$B(c, \ell) = \{j \in \{1, \dots, n\} \mid d(c, j) \leq \ell\}.$$

Le problème consiste alors à déterminer le **nombre minimal** de ces boîtes, noté $N(\ell)$, qui permet de recouvrir l'ensemble des nœuds du SCN. Ce nombre, en fonction de l'échelle ℓ , joue un rôle analogue au nombre de cubes nécessaires pour recouvrir un objet géométrique dans \mathbb{R}^d .

B. Procédure du Box-Counting et Estimation de la Dimension Fractale

La méthode du **box-counting** adaptée à un SCN s'articule autour de plusieurs étapes essentielles. D'une part, il faut établir une couverture du graphe par des boules de rayon ℓ . Pour une valeur d'échelle fixée, on choisit successivement des nœuds centraux, construit les boules correspondantes $B(c, \ell)$ et retire les nœuds ainsi couverts du graphe jusqu'à ce que tous les nœuds soient inclus dans au moins une boule. Le nombre minimal de boules obtenues dans cette procédure est noté $N(\ell)$.

Si, lorsque l'échelle ℓ tend vers zéro, on observe que

$$N(\ell) \sim \ell^{-\dim_f},$$

la **dimension fractale** du SCN est définie par la relation :

$$\dim_f = \lim_{\ell \rightarrow 0} \frac{\ln N(\ell)}{-\ln \ell}.$$

En pratique, on trace la courbe logarithmique $\ln N(\ell)$ en fonction de $\ln(1/\ell)$ et la pente de la droite de régression (sur un intervalle $[\ell_{\min}, \ell_{\max}]$) fournit une estimation de \dim_f . La linéarité de cette courbe dans cet intervalle est le témoignage de l'**invariance d'échelle** qui caractérise la structure fractale.

Il est à noter que la méthode du box-counting peut être complétée par d'autres approches, telles que la **dimension de corrélation**, où l'on définit la fonction

$$C(\ell) = \frac{2}{n(n-1)} \sum_{i < j} \chi(d(i,j) \leq \ell),$$

avec χ la fonction indicatrice, et l'on examine la dépendance de $C(\ell)$ en fonction de ℓ pour estimer la dimension de corrélation \dim_{corr} par la relation $C(\ell) \sim \ell^{\dim_{\text{corr}}}$. De même, l'analyse des **lois de puissance** dans la distribution des tailles de clusters ou des degrés des nœuds permet de détecter une invariance d'échelle. Par exemple, si la distribution des tailles de clusters satisfait

$$\text{Prob}(|\mathcal{C}| > s) \sim s^{-\alpha},$$

on identifie une caractéristique scale-free, qui renforce l'hypothèse d'auto-similarité dans l'agrégation des clusters.

C. Implications pour le Deep Synergy Learning (DSL)

L'application de ces méthodes fractales dans le cadre d'un SCN, et plus particulièrement du DSL, offre plusieurs avantages importants. Lorsque la même dynamique d'auto-organisation, exprimée par la règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

s'exerce à toutes les échelles, on obtient une auto-similarité qui se traduit par une invariance de la distribution des pondérations et des tailles de clusters. Cette invariance permet de simplifier la modélisation multi-échelle, car une même loi (ou un même algorithme) peut être utilisée pour décrire les interactions tant au niveau local que global. De plus, la robustesse du système est renforcée puisque l'auto-organisation, en se répétant à différentes granularités, tend à corriger les perturbations locales par une réorganisation à plus grande échelle. La dimension fractale \dim_f quantifiée par la méthode du box-counting constitue ainsi un indicateur précieux pour l'analyse du comportement global du SCN et pour le réglage des paramètres du DSL.

D. Conclusion

L'approche par boîtes (box-counting) appliquée à un SCN offre une méthode rigoureuse pour quantifier l'**auto-similarité** d'un réseau synergétique en estimant sa **dimension fractale**. En définissant une métrique adaptée $d(i,j)$ à partir des pondérations $\omega_{i,j}$ – par exemple, via l'inversion $1/\omega_{i,j}$ – et en recouvrant le graphe par des boules de rayon ℓ , il est possible de déterminer le nombre minimal $N(\ell)$ de boîtes nécessaires pour couvrir l'ensemble des nœuds. La relation

$$N(\ell) \sim \ell^{-\dim_f}$$

et la linéarité de la courbe $\ln N(\ell)$ versus $\ln(1/\ell)$ permettent d'extraire la dimension fractale \dim_f du réseau. L'émergence d'une telle invariance d'échelle, caractérisée par des lois de puissance dans la distribution de ω et des tailles de clusters, est une signature d'un comportement fractal dans le SCN. Pour le **Deep Synergy Learning**, cette propriété signifie que la même dynamique d'auto-organisation s'applique de manière récurrente à toutes les échelles, ce qui simplifie l'analyse multi-échelle, renforce la robustesse du système et offre des outils de réglage précis des paramètres d'apprentissage. En somme, le box-counting constitue un instrument essentiel pour mesurer quantitativement l'auto-similarité d'un SCN et pour démontrer que, indépendamment de l'échelle d'observation, la structure du réseau reste invariantée, attestant ainsi de sa nature fractale.

6.3.3.2. Représentation d'échelles multiples : la “scale invariance” si la structure se répète

Dans le cadre de l'analyse des **réseaux synergétiques**, il apparaît qu'un même **motif** d'organisation peut se reproduire à différentes échelles, ce que l'on désigne par le terme d'**invariance d'échelle** ou **scale invariance**. Cette propriété, intrinsèque aux systèmes fractals, se manifeste dans un SCN (Synergistic Connection Network) lorsque la même répartition des liaisons, la même configuration topologique, se retrouve de manière auto-similaire que ce soit au niveau local (micro-clusters) ou global (macro-clusters). L'objectif de cette section est de présenter rigoureusement comment la représentation d'échelles multiples permet de quantifier et d'exploiter cette invariance, et d'en démontrer les implications pour la dynamique du DSL (Deep Synergy Learning).

A. Représentation Multiscale du SCN

Considérons un SCN composé d'un ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ reliées par des pondérations $\omega_{i,j}$. La dynamique d'auto-organisation, régie par une règle de mise à jour telle que

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

s'exerce uniformément sur le réseau. Dès lors, il est naturel de considérer que, par des procédés d'**agrégation hiérarchique** ou de *coarse-graining* (voir notamment la section 6.2.2), le SCN peut être représenté à plusieurs niveaux de granularité. Ainsi, on introduit une suite d'agrégations

$$\omega^{(0)} \rightarrow \omega^{(1)} \rightarrow \omega^{(2)} \rightarrow \dots \rightarrow \omega^{(K)},$$

où $\omega^{(0)}$ représente la matrice de pondérations du réseau initial (au niveau micro) et $\omega^{(K)}$ correspond à celle obtenue après K niveaux d'agrégation, chacun étant obtenu par une opération de regroupement des entités en « super-nœuds ». À chaque niveau, la même dynamique d'auto-organisation est appliquée aux agrégats, de sorte que la **structure** (la répartition des liens et la topologie) reste, à un facteur d'échelle près, identique à celle du niveau précédent.

Pour formaliser cette idée, on suppose qu'il existe un **isomorphisme approché** ϕ et un **facteur de rescaling** $\lambda > 0$ tels que, pour tout couple d'entités i, j appartenant à un cluster au niveau ℓ , la relation

$$\omega_{\phi(i), \phi(j)}^{(\ell+1)} \approx \lambda \omega_{i,j}^{(\ell)}$$

soit vérifiée. Cette relation exprime que la *même* forme structurelle est retrouvée au niveau $\ell + 1$ (macro-niveau) à partir du niveau ℓ (micro-niveau) une fois les pondérations multipliées par un facteur d'échelle constant. En d'autres termes, la configuration topologique du SCN se reproduit de manière **auto-similaire** à travers les niveaux d'agrégation.

B. Détection et Quantification de la Scale Invariance

L'approche par boîtes (ou **box-counting**), présentée en 6.3.3.1, offre un cadre permettant de mesurer la **dimension fractale** du SCN. Dans un contexte multiscale, cette méthode s'applique non seulement au niveau initial mais également sur les versions agrégées du réseau. En recouvrant le réseau par des « boîtes » ou boules de rayon ℓ (définies via une métrique adaptée $d(i, j)$, par exemple $d(i, j) = \min_{\gamma(i \rightarrow j)} \sum_{(\mu, v) \in \gamma} \frac{1}{\omega_{\mu, v}}$), on détermine le nombre minimal $N(\ell)$ de boîtes nécessaires pour couvrir l'ensemble des entités. Si l'on trouve que

$$N(\ell) \sim \ell^{-\dim_f},$$

alors la pente de la droite obtenue en traçant $\ln N(\ell)$ en fonction de $\ln(1/\ell)$ donne la **dimension fractale** \dim_f du réseau. Le fait que cette relation reste valable à différents niveaux d'agrégation atteste de l'**invariance d'échelle** du SCN. De surcroît, une analyse statistique complémentaire basée sur la distribution en **lois de puissance** des tailles de clusters ou des degrés des nœuds renforce l'idée que le même schéma organisationnel se retrouve quelle que soit l'échelle.

La **dimension de corrélation**, obtenue par l'analyse de la fonction de corrélation

$$C(\ell) = \frac{2}{n(n-1)} \sum_{i < j} \chi(d(i, j) \leq \ell),$$

est une autre méthode qui permet d'évaluer l'auto-similarité du réseau. Si $C(\ell)$ satisfait une relation de la forme

$$C(\ell) \sim \ell^{\dim_{corr}},$$

alors \dim_{corr} fournit une estimation alternative de la dimension fractale. Ces approches, en s'appliquant de manière répétée sur les différentes couches d'agrégation du SCN, permettent de vérifier que la **structure** du réseau est effectivement **scale-invariant**.

C. Implications pour le Deep Synergy Learning (DSL)

L'existence d'une invariance d'échelle dans un SCN signifie que la même **règle d'auto-organisation** peut être appliquée de manière itérative à différents niveaux, du micro au macro, sans nécessiter d'ajustements spécifiques pour chaque palier. Si le mécanisme DSL (par exemple, la règle additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$$

) est le même à toutes les échelles, alors un micro-cluster se structure de la même manière qu'un macro-cluster, à un facteur de rescaling près. Cette **homogénéité** du mécanisme d'auto-organisation assure une **robustesse** intrinsèque, car toute perturbation locale peut être compensée par la réapparition du même motif à une échelle supérieure. Cette propriété fractale simplifie l'analyse du comportement multi-échelle du système, car la connaissance de la dynamique à une échelle permet de prédire le comportement à d'autres échelles grâce à des relations de type

$$\omega_{\phi(i), \phi(j)}^{(\ell+1)} \approx \lambda \omega_{i,j}^{(\ell)}.$$

Par ailleurs, l'obtention d'une dimension fractale non entière (par exemple, $\dim_f = 1.5$ ou 2.3) offre un indicateur quantitatif du degré de complexité et d'irrégularité du réseau. Cette mesure peut être utilisée pour adapter dynamiquement les paramètres du DSL, afin de maintenir la stabilité et la cohérence des structures émergentes.

D. Conclusion

L'approche par boîtes appliquée à un SCN permet de quantifier l'**invariance d'échelle** du réseau en définissant une métrique adaptée et en recouvrant l'ensemble des nœuds par des boules de rayon ℓ . Si le nombre de boules $N(\ell)$ satisfait la relation

$$N(\ell) \sim \ell^{-\dim_f},$$

alors la **dimension fractale** \dim_f du SCN se déduit de la pente de la droite obtenue en traçant $\ln N(\ell)$ versus $\ln(1/\ell)$. Cette relation, valable sur un intervalle d'échelles $[\ell_{\min}, \ell_{\max}]$, témoigne de la **scale invariance** du SCN, c'est-à-dire que la même structure d'interconnexion se répète, à un facteur multiplicatif près, du niveau micro au niveau macro. Cette propriété est fondamentale pour le DSL, car elle signifie que la même règle d'auto-organisation peut être appliquée de manière récurrente à toutes les échelles, simplifiant ainsi l'analyse multi-échelle, renforçant la robustesse du système et permettant d'adapter de manière universelle les paramètres d'apprentissage. En somme, la méthode du box-counting et ses dérivés offrent un cadre quantitatif puissant pour démontrer que l'organisation d'un SCN est fractale, c'est-à-dire qu'elle présente une **auto-similarité** persistante quel que soit le niveau d'observation, illustrant ainsi l'universalité et l'efficacité de la dynamique DSL dans la modélisation des systèmes complexes.

6.3.3.3. Exemples de phénomènes fractals dans la nature (p. ex. réseaux vasculaires) et analogies avec le DSL

La compréhension des **phénomènes fractals** dans la nature repose sur l'observation que la structure des systèmes biologiques et géophysiques se répète à différentes échelles, de manière hiérarchique et auto-similaire. Cette propriété d'**invariance d'échelle** se manifeste notamment dans les **réseaux vasculaires**, les **ramifications bronchiques** et les **systèmes racinaires**. Ces exemples offrent des analogies riches avec le **Deep Synergy Learning** (DSL) lorsque celui-ci se déploie en mode multi-niveau, permettant d'illustrer que la même dynamique d'auto-organisation peut s'appliquer tant au niveau micro qu'au niveau macro, assurant une cohérence structurelle du réseau.

A. Réseaux Vasculaires et Bronchiques : Une Ramification Auto-similaire



Dans le domaine biologique, le système cardiovasculaire présente une organisation fractale remarquable. L'**aorte** se subdivise en artères, lesquelles se transforment en artéries, avant de se ramifier en capillaires. À chaque palier de

ce qui implique qu'à chaque bifurcation, le diamètre des vaisseaux décroît d'un facteur constant, et que la distribution des rayons suit une loi de puissance. L'auto-similarité dans ce contexte se manifeste par la répétition d'un même motif de subdivision, de sorte que la structure globale demeure invariante d'échelle, à l'exception d'un facteur multiplicatif constant. Dans un **SCN fractal**, les pondérations $\omega_{i,j}$ peuvent être interprétées comme des "flux" ou intensités de connexion, et si la même règle de mise à jour (renforcement/inhibition) s'applique à chaque niveau, on observe que la structure d'un petit groupe de nœuds (micro-cluster) se retrouve, à un facteur d'échelle près, dans le réseau global (macro-cluster). Cette correspondance est formellement exprimée par la relation

$$r_{\text{parent}}^3 \approx r_{\text{enfant}_1}^3 + r_{\text{enfant}_2}^3,$$

où ϕ désigne une application de correspondance entre les nœuds du micro-cluster et ceux du macro-cluster, et λ représente le facteur de rescaling. L'existence d'un tel facteur souligne l'**invariance d'échelle** dans le réseau, analogue à la manière dont le système vasculaire conserve sa forme malgré la diminution progressive des diamètres des vaisseaux.

B. Systèmes Racinaires et Réseaux de Rivières

Les **systèmes racinaires** constituent un autre exemple marquant d'auto-similarité dans la nature. La racine principale se divise en racines secondaires, lesquelles se subdivisent à leur tour en radicelles. Mathématiquement, ce processus de division peut être modélisé par une relation de type

$$L_{\text{tot}} \sim \sum_{k=1}^m r_k^{-\beta},$$

où r_k représente le diamètre des branches à chaque niveau, et l'exposant β caractérise l'agencement fractal du système. Une structure similaire se retrouve dans les **réseaux de rivières**. Ces derniers sont organisés selon des règles telles que celles de **Horton-Strahler** ou de **Tokunaga**, qui décrivent comment chaque sous-affluent se structure de manière récurrente par rapport au cours d'eau principal. Le petit cours d'eau possède la même topologie que la rivière principale, mis à part un facteur d'échelle en termes de longueur et de débit. Cette récurrence topologique signifie que l'auto-organisation observée dans un SCN, par le biais de la mise à jour des pondérations, est comparable à ces processus naturels où l'agrégation et la division se font de manière régulière et prévisible.

C. Analogies Essentielles avec le DSL

L'analogie entre les phénomènes fractals naturels et le **Deep Synergy Learning** se fonde sur plusieurs aspects fondamentaux. D'une part, les **réseaux vasculaires** ou **systèmes racinaires** présentent une **absence d'échelle caractéristique** ; toutes les échelles participent de manière équivalente à la formation de la structure globale. De même, dans un SCN où la dynamique DSL ne privilégie aucune échelle – c'est-à-dire qu'il n'existe pas de seuil arbitraire imposant une taille fixe pour un cluster – la distribution des pondérations et la formation des clusters suivent une **loi de puissance**. D'autre part, l'**auto-organisation** dans ces systèmes se caractérise par une **hiérarchie récurrente**. Un petit groupe local qui se structure selon un certain motif se retrouve, lorsqu'on l'aggrave, à l'échelle macro, en reproduisant le même schéma, comme si l'ensemble du réseau était le reflet d'une même dynamique, simplement dilatée par un facteur constant. Cette **self-similarity** se traduit dans le DSL par l'application uniforme de la règle d'actualisation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

à toutes les échelles, ce qui garantit que le même motif de renforcement et d'inhibition apparaît du micro au macro. Par ailleurs, la **robustesse** inhérente à une structure fractale – où l'absence d'un point critique unique permet à l'ensemble du système de se réorganiser même en cas de perturbation locale – se transpose directement dans le DSL, rendant le modèle plus résilient face aux variations et aux imprévus.

D. Conclusion

Les exemples de phénomènes fractals dans la nature, tels que les **réseaux vasculaires**, les **ramifications bronchiques** et les **systèmes racinaires**, illustrent de manière concrète la notion d'**invariance d'échelle**. Chaque subdivision ou bifurcation dans ces systèmes se reproduit à un facteur constant, donnant lieu à des lois de puissance qui décrivent l'ensemble de la structure. En analogie avec le **Deep Synergy Learning**, si la dynamique d'auto-organisation s'applique uniformément à travers des niveaux hiérarchiques (du micro-cluster au macro-cluster), alors le SCN présente un comportement fractal caractérisé par une auto-similarité des motifs de connexion. Ce comportement, qui se traduit par des relations telles que

$$\omega_{\phi(i),\phi(j)}^{(\text{glob})} \approx \lambda \omega_{i,j}^{(\text{loc})},$$

illustre que le même processus d'agrégation et d'inhibition opère indépendamment de l'échelle, garantissant une **robustesse** et une **universalité** du modèle. L'analogie avec les systèmes naturels souligne ainsi l'intérêt de concevoir des mécanismes d'auto-organisation capables de reproduire des structures hiérarchiques et fractales, ce qui constitue l'un des atouts majeurs du DSL dans la modélisation et la compréhension des réseaux complexes.

6.3.4. Avantages et Limitations

Après avoir exploré les principes de la fractalité (6.3.1 et 6.3.2) et comment la *quantifier* (6.3.3), il convient d'évaluer ce que **signifie** concrètement, pour un **SCN** (Synergistic Connection Network), d'être (ou de tendre vers) un **système fractal**. La fractalité n'est pas une finalité **systématique** dans le DSL (Deep Synergy Learning), mais plutôt un **cas particulier** susceptible d'offrir certains avantages tout en présentant des limites.

6.3.4.1. Avantage : la fractalité suggère une “unité” ou une cohérence à toutes les échelles

Dans l'étude des réseaux complexes, la notion de **fractalité** apparaît comme une caractéristique essentielle, notamment parce qu'elle suggère l'absence d'une taille caractéristique imposée aux **clusters** ou aux **liens**. Cette propriété d'**invariance d'échelle** se manifeste lorsque, quelle que soit la granularité d'observation – du niveau **micro** jusqu'au niveau **macro** – les motifs d'organisation se répètent, de manière auto-similaire, à un facteur d'échelle près. En d'autres termes, la même structure de distribution des **pondérations** $\omega_{i,j}$ et la même logique d'auto-organisation se retrouvent, que l'on “zoome” sur un petit sous-groupe ou que l'on “dézoomé” sur l'ensemble du réseau. Ce phénomène, qui constitue l'un des piliers théoriques de la modélisation fractale, présente plusieurs avantages tant sur le plan mathématique que sur le plan opérationnel dans le cadre du **Deep Synergy Learning (DSL)**.

Soit un réseau synergétique $G = (V, E)$ où chaque nœud représente une entité \mathcal{E}_i et chaque arête (i, j) est associée à une pondération $\omega_{i,j}$ qui reflète le degré de **synergie** entre les entités i et j . La dynamique d'auto-organisation dans le DSL est gouvernée par une règle de mise à jour, souvent exprimée sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η est le **taux d'apprentissage** et τ est le **coefficent de décroissance**. Lorsque cette même règle est appliquée de manière uniforme à toutes les échelles du réseau, elle produit des motifs récurrents, ce qui conduit à une structure fractale. En effet, dans un tel réseau, la distribution des **clusters** ne présente pas de taille caractéristique ; au contraire, un *micro-cluster* local peut présenter exactement la même topologie qu'un *macro-cluster*, à un facteur d'échelle près. Formellement, on peut exprimer cette idée par l'existence d'un isomorphisme approché ϕ et d'un facteur $\lambda > 0$ tels que, pour tout i, j appartenant à un cluster local \mathcal{C}_{loc} , la relation suivante est satisfaite :

$$\omega_{\phi(i),\phi(j)}^{(\text{glob})} \approx \lambda \omega_{i,j}^{(\text{loc})}.$$

Cette équation illustre que, si l'on agrège les entités du niveau micro pour obtenir un niveau macro, la *forme* des liaisons se conserve, seule l'échelle change.

Du point de vue **mathématique**, cette invariance d'échelle se manifeste par la présence de **lois de puissance** dans la distribution des tailles de clusters ou des degrés des nœuds. Par exemple, si l'on note $s = |\mathcal{C}|$ la taille d'un cluster, on peut observer que la probabilité qu'un cluster dépasse une taille s se comporte selon

$$\text{Prob}(s > x) \sim x^{-\alpha},$$

ce qui indique que le même mécanisme d'agrégation s'applique indépendamment de l'échelle considérée. De même, une analyse via la méthode du **box-counting** (voir la section 6.3.3.1) permet d'extraire une dimension fractale \dim_f du réseau en montrant que

$$N(\ell) \sim \ell^{-\dim_f},$$

où $N(\ell)$ représente le nombre minimal de boîtes de rayon ℓ nécessaires pour recouvrir l'ensemble des nœuds. La constance de \dim_f sur un intervalle d'échelles indique que le réseau est **auto-similaire**.

D'un point de vue **ingénierique**, le fait que la même logique d'auto-organisation (les mêmes paramètres η et τ dans la dynamique DSL, ainsi que les mêmes mécanismes d'inhibition ou de saturation) se répète de manière identique du niveau micro au niveau macro confère une **unité** au système. Cette **cohérence** structurelle se traduit par une simplification de l'implémentation, puisqu'un unique ensemble de règles s'applique à toute la hiérarchie. Par ailleurs, en cas de perturbation locale, le système peut se reconfigurer en s'appuyant sur les mêmes schémas fractals à d'autres niveaux, renforçant ainsi sa **robustesse** et son **adaptabilité**. La capacité à analyser un petit sous-ensemble du réseau et à extrapoler ses propriétés à l'ensemble du SCN constitue un avantage considérable pour l'analyse et la conception des systèmes d'auto-organisation.

En somme, la fractalité suggère que le réseau, quelle que soit son échelle d'observation, présente une **unité** intrinsèque ; la **même forme** de distribution des liens et la même logique d'auto-organisation se retrouvent aussi bien dans un micro-cluster que dans un macro-cluster, modulo un simple changement d'échelle par un facteur λ . Cette propriété renforce la **cohérence** du DSL et facilite l'analyse multi-échelle, tout en simplifiant la mise en œuvre et le réglage des paramètres du système. L'universalité de la dynamique DSL, qui se répète à tous les niveaux, est ainsi l'une des raisons majeures pour lesquelles les réseaux fractals sont particulièrement recherchés dans la modélisation des systèmes complexes et auto-organisés.

Conclusion.

Les **réseaux fractals** offrent un sentiment de continuité et d'unité à toutes les échelles, car la même loi d'auto-organisation – caractérisée par des mécanismes de renforcement et d'inhibition appliqués de manière homogène – se répète de manière auto-similaire. Cette invariance d'échelle facilite l'analyse mathématique et la modélisation multi-échelle, renforce la robustesse du système et simplifie la mise en œuvre d'un DSL en permettant une réutilisation universelle des mêmes règles, quelle que soit la granularité d'observation.

6.3.4.2. Limitation : tous les systèmes DSL ne sont pas fractals ; c'est un cas particulier où les synergies multi-niveau se recoupent fortement

Dans l'étude des **réseaux synergétiques** et du **Deep Synergy Learning (DSL)**, il est tentant d'identifier des structures fractales lorsque la même dynamique d'auto-organisation se répète à différentes échelles. Toutefois, il apparaît que l'émergence d'un comportement fractal n'est pas universelle au sein des systèmes DSL, mais se manifeste seulement dans des conditions particulières où les synergies multi-niveau s'alignent de façon très homogène. Cette limitation doit être examinée à la lumière de plusieurs aspects théoriques et pratiques, et il est essentiel d'en préciser les conditions, afin de ne pas extrapoler de manière abusive l'idée de fractalité à l'ensemble des systèmes DSL.

Une première condition requise pour qu'un SCN puisse exhiber une structure fractale est que la **fonction d'agrégation** utilisée dans la dynamique d'auto-organisation, souvent notée Ψ dans le cadre des opérations de coarse-graining (cf. section 6.2.2), soit appliquée de manière quasi uniforme à tous les niveaux. Il faut que la règle d'actualisation des pondérations, typiquement donnée par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

reçoive exactement le même traitement que l'on applique à des sous-ensembles du réseau, de sorte que la distribution des valeurs $\omega_{i,j}$ à chaque niveau d'agrégation soit comparable, modulo un simple facteur d'échelle. Dès lors, si les **paramètres** η et τ ne varient pas ou varient proportionnellement d'un niveau à l'autre, la dynamique risque de se maintenir, produisant ainsi une auto-similarité au sens strict. En revanche, si ces paramètres sont ajustés indépendamment à chaque palier – par exemple, en introduisant un taux d'apprentissage local η_{loc} distinct d'un taux global η_{glob} – l'uniformité de la règle d'auto-organisation est rompue, et le réseau ne peut plus présenter la récurrence des motifs caractéristiques d'un système fractal.

Un deuxième facteur réside dans la **nature des données** et la **distribution initiale** des synergies. Dans de nombreux systèmes DSL, les entités sont hétérogènes, et la répartition des pondérations $\omega_{i,j}$ peut être influencée par des contraintes externes ou par des mécanismes de saturation qui imposent une échelle caractéristique. Par exemple, lorsqu'un seuil de saturation est appliqué pour éviter que les pondérations ne croissent indéfiniment, ou lorsque des règles spécifiques de formation des clusters induisent une taille moyenne prédéfinie, la distribution des $\omega_{i,j}$ ne suit plus une loi de puissance pure. Cette rupture de la loi de puissance empêche l'apparition d'un comportement véritablement **scale-free**, et par conséquent, la fractalité du système ne s'exprime que de manière partielle ou localisée.

Un troisième aspect concerne la **diversité topologique** inhérente au réseau. Même si l'on applique une règle DSL uniforme, la topologie du SCN peut être influencée par des facteurs externes ou par des propriétés intrinsèques des interactions entre entités. Par exemple, dans un système multimodal, certains sous-groupes d'entités se formeront naturellement selon des motifs distincts en fonction de leurs caractéristiques spécifiques, tandis que d'autres adopteront une organisation plus homogène. Cette hétérogénéité dans l'**organisation** empêche la réapparition d'un motif unique à toutes les échelles, et seule une partie du réseau pourra être qualifiée de fractale, tandis que d'autres portions présenteront des distributions plus ordinaires.

En résumé, la fractalité dans un SCN n'émerge que dans le cas particulier où (i) la même **fonction d'agrégation** Ψ est appliquée de manière uniforme à tous les niveaux, (ii) les paramètres d'actualisation η et τ restent constants ou sont ajustés de manière proportionnelle, et (iii) la distribution initiale des synergies et des données est suffisamment homogène pour ne pas introduire de coupures d'échelle. Lorsque ces conditions ne sont pas réunies – par exemple, en présence de **paramètres divergents** entre les niveaux, de contraintes de saturation strictes ou de données hétérogènes – le SCN ne présente pas de comportement fractal global, mais seulement une **fractalité partielle** ou même l'absence d'auto-similarité.

Cette limitation est essentielle à comprendre, car elle indique que l'apparition d'une **structure fractale** dans un système DSL est un cas particulier, illustrant l'unité et la robustesse de la dynamique d'auto-organisation, mais qui n'est pas généralisable à tous les réseaux synergétiques. D'un point de vue théorique, cela signifie que les outils mathématiques permettant de quantifier la fractalité, tels que le box-counting, la dimension de corrélation ou l'analyse des lois de puissance, ne s'appliquent que lorsque le réseau ne présente pas de contraintes d'échelle imposées par la nature des données ou par les mécanismes d'inhibition et de saturation. D'un point de vue opérationnel, il convient de reconnaître que, dans la pratique, de nombreux systèmes DSL ne sont que partiellement fractals et que leur analyse doit intégrer cette hétérogénéité afin d'éviter des interprétations trop simplistes de la dynamique d'auto-organisation.

Conclusion.

La fractalité, en tant que manifestation d'une invariance d'échelle et d'une auto-similarité des motifs d'interconnexion, apparaît dans un SCN uniquement lorsque les mécanismes d'auto-organisation se recoupent fortement et de manière homogène à tous les niveaux. En revanche, si les paramètres de mise à jour varient d'un niveau à l'autre, ou si des contraintes externes imposent une échelle caractéristique aux interactions, l'auto-similarité fractale s'en trouve rompue. Ainsi, bien que la présence d'une structure fractale offre de nombreux avantages, elle constitue un cas particulier dans l'univers DSL, et ne peut être considérée comme généralisée à l'ensemble des systèmes multi-niveaux, qui présenteront souvent une **fractalité partielle** ou l'absence complète d'auto-similarité à cause de la diversité des conditions initiales et des mécanismes locaux d'agrégation et d'inhibition.

6.3.4.3. Applicabilité : ex. grands réseaux hétérogènes, IA inspirée du cerveau

Dans le contexte des systèmes complexes, l'idée d'**invariance d'échelle** se retrouve naturellement dans de nombreux phénomènes naturels et inspire la modélisation de systèmes artificiels tels que le **Deep Synergy Learning (DSL)**. Cette section examine de manière approfondie l'applicabilité des concepts fractals à de grands réseaux hétérogènes et à des approches d'**IA inspirée du cerveau**, en mettant en évidence les avantages mais également les limites de l'hypothèse d'auto-similarité dans ces domaines.

A. Grands Réseaux Hétérogènes

Dans des environnements comportant des millions d'entités et une multiplicité de types de relations – par exemple dans les **réseaux sociaux** ou les systèmes multi-domaines – aucune taille de cluster n'est imposée a priori. En l'absence de contraintes externes fortes, la dynamique d'auto-organisation, pilotée par une règle DSL uniforme, permet l'émergence de clusters de toutes tailles. D'un point de vue mathématique, cette liberté se traduit souvent par des distributions de taille de clusters qui suivent une **loi de puissance** :

$$\text{Prob}(|\mathcal{C}| > s) \sim s^{-\alpha},$$

ce qui signifie qu'il n'existe pas de taille caractéristique dominante dans le réseau. L'**absence de scale** se reflète ainsi dans une distribution continue et heavy-tail des agrégations, indiquant que le même mécanisme d'agrégation – notamment le renforcement des liens forts et l'inhibition des liens faibles – opère de manière homogène sur toutes les échelles. Par conséquent, dans ces grands réseaux hétérogènes, le DSL, lorsqu'il est laissé relativement libre de ses paramètres, peut générer ou révéler une **structure fractale**. Cette propriété offre l'avantage majeur de la **robustesse** et de la **flexibilité** de l'organisation : l'arrivée de nouveaux nœuds, ou la disparition de certains, ne perturbe pas l'équilibre global puisque la loi de puissance continue de s'appliquer, garantissant ainsi une auto-organisation sans point critique fixe.

B. IA Inspirée du Cerveau

La biologie neuronale offre un exemple emblématique d'**organisation multi-échelle**. Le cerveau humain, par exemple, présente une structure hiérarchique allant des micro-colonnes corticales aux aires fonctionnelles et jusqu'aux lobes. Des études empiriques suggèrent que les connexions neuronales, tant au niveau des synapses que dans la configuration des réseaux neuronaux, peuvent exhiber des propriétés d'auto-similarité et d'invariance d'échelle. Mathématiquement, de telles structures se caractérisent par des mesures comme la **dimension fractale** ou par des lois de puissance dans la distribution des degrés de connexion, lesquelles indiquent que les motifs de connectivité se répètent à toutes les échelles.

Dans une approche DSL inspirée du cerveau, la règle d'auto-organisation – par exemple

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

– s'applique de manière uniforme du niveau micro (groupes de neurones ou micro-colonnes) jusqu'au niveau macro (aires corticales interconnectées). La réutilisation des mêmes mécanismes d'apprentissage et de plasticité synaptique, associés à des schémas de connectivité auto-similaire, confère au réseau une **robustesse** comparable à celle observée dans le cortex. Un tel système permet, par analogie avec les réseaux vasculaires ou bronchiques, de maintenir une cohérence structurelle malgré l'hétérogénéité des modules fonctionnels, facilitant ainsi la transmission d'informations et l'adaptation aux perturbations.

C. Synthèse et Implications

La fractalité, lorsqu'elle se manifeste dans un SCN, offre un cadre conceptuel puissant pour l'analyse de grands réseaux hétérogènes ainsi que pour la conception d'architectures d'**IA inspirée du cerveau**. Dans ces domaines, l'**absence d'une taille caractéristique imposée aux clusters** – traduite mathématiquement par des lois de puissance telles que

$$\text{Prob}(|\mathcal{C}| > s) \sim s^{-\alpha}$$

– et la récurrence de la même dynamique d'auto-organisation à toutes les échelles, permettent d'envisager un modèle robuste et évolutif. L'universalité de la règle DSL assure que, même en présence d'une hétérogénéité extrême, les motifs d'interconnexion se reproduisent de façon auto-similaire, garantissant une répartition efficace des ressources et

une coordination harmonieuse. Toutefois, il convient de noter que la fractalité est souvent un cas particulier qui se révèle pleinement lorsque les conditions d'homogénéité des paramètres et d'absence de contraintes externes prédominantes sont réunies. Dans de nombreux systèmes réels, des facteurs tels que des seuils de saturation ou des différences de paramètres entre les niveaux peuvent limiter l'étendue de cette invariance d'échelle. Néanmoins, l'inspiration tirée des structures fractales observées dans la nature – des réseaux vasculaires aux systèmes racinaires – offre des perspectives prometteuses pour concevoir des SCN capables de s'auto-organiser de manière à la fois flexible et résiliente, en reproduisant des motifs d'organisation qui transcendent les échelles d'observation.

D. Conclusion

L'applicabilité des concepts fractals dans les grands réseaux hétérogènes et dans les systèmes d'IA inspirés du cerveau repose sur l'observation que, lorsque les mécanismes d'auto-organisation sont uniformes et qu'aucune échelle caractéristique n'est imposée, le SCN peut exhiber des lois de puissance et une invariance d'échelle. Cette propriété, qui se traduit par la répétition auto-similaire de la structure du réseau à tous les niveaux – du micro-cluster au macro-cluster – confère au système une robustesse et une adaptabilité remarquables. Par analogie avec des systèmes naturels tels que les réseaux vasculaires, les systèmes racinaires ou les bassins-versants, un DSL fractal permet de concevoir des architectures qui s'auto-organisent de manière universelle, offrant ainsi des perspectives d'ingénierie pour des systèmes complexes capables de gérer l'hétérogénéité et de s'adapter aux variations multi-échelles.

6.4. Interactions Multi-Niveau et Coordination

Le **DSL** (Deep Synergy Learning) opère de manière particulièrement riche lorsqu'il est déployé sur un **SCN** (Synergistic Connection Network) organisé en **plusieurs niveaux** (micro, méso, macro). Dans ce cadre, les **flux** ascendants (bottom-up) et descendants (top-down) assurent une **coordination** essentielle entre les niveaux, les entités (ou petits clusters) **remontent** leur organisation vers les paliers supérieurs, tandis que les paliers supérieurs **rétroagissent** pour influencer ou “piloter” la configuration des liens au palier inférieur. La section 6.4.1 récapitule ces deux flux de base et souligne l'importance de la cohérence entre niveaux.

6.4.1. Bottom-Up vs. Top-Down

Dans un **SCN** multi-niveau, on distingue au moins deux grands axes de **communication** :

- 374. **Flux ascendants (bottom-up)** : depuis les entités brutes (niveau micro), on agrège progressivement des clusters vers un niveau macro (ou intermédiaire).
- 375. **Flux descendants (top-down)** : le macro-niveau, disposant d'une vue plus globale, impose un **feedback** (contrainte ou renforcement) sur les pondérations ω au niveau inférieur.

6.4.1.1. Rappel des flux ascendants (entités → cluster local → cluster macro)

Les **flux ascendants** dans un **SCN** (Synergistic Connection Network) multi-échelle désignent le mécanisme par lequel on **construit** progressivement, à partir d'entités élémentaires au niveau micro, des **clusters** toujours plus vastes au fil des niveaux supérieurs, jusqu'à atteindre un niveau macro. Cette démarche correspond à un processus de *coarse-graining* itératif, où l'on consolide des groupes de bas niveau en “super-nœuds” plus globaux, optimisant ainsi la représentation et la gestion de la complexité.

On considère un ensemble d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$, il s'agit du **niveau 0** ou micro-niveau. Les **pondérations** $\omega_{i,j}^{(0)}$ décrivent les affinités (ou synergies) entre ces entités. Dans un **DSL** (Deep Synergy Learning), ces pondérations évoluent selon une règle itérative, par exemple de type additive :

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_0 [S_0(i,j) - \tau_0 \omega_{i,j}^{(0)}(t)],$$

avec η_0, τ_0 pour le niveau micro. À un instant donné, certains groupes d'entités peuvent présenter une cohésion interne suffisamment élevée (somme des $\omega_{i,j}$ importante).

Dès lors que l'on détecte un **cluster** $\mathcal{C}_\alpha^{(0)}$ (quelques entités fortement reliées), on mesure par exemple :

$$\Omega(\mathcal{C}_\alpha^{(0)}) = \sum_{i,j \in \mathcal{C}_\alpha^{(0)}} \omega_{i,j}^{(0)}.$$

Un **seuil** θ_0 (cohésion minimale) permet de décider si ce regroupement est *actif*. Chaque cluster local $\mathcal{C}_\alpha^{(0)}$ acquiert une identité propre, possiblement appelée “super-nœud de niveau 1”, notée $\mathcal{N}_\alpha^{(1)}$.

Une fois qu'on a constitué plusieurs clusters $\{\mathcal{C}_1^{(0)}, \dots, \mathcal{C}_r^{(0)}\}$ au niveau 0, chacun se transforme en **super-nœud** $\{\mathcal{N}_1^{(1)}, \dots, \mathcal{N}_r^{(1)}\}$. Les pondérations entre ces super-nœuds s'obtiennent via une **fonction** d'agrégation Ψ appliquée aux $\omega_{i,j}^{(0)}$ des entités qu'ils contiennent :

$$\omega_{\alpha,\beta}^{(1)} = \Psi \left(\left\{ \omega_{i,j}^{(0)} \mid i \in \mathcal{C}_\alpha^{(0)}, j \in \mathcal{C}_\beta^{(0)} \right\} \right).$$

Une fois ce niveau 1 établi, on reproduit une **même** dynamique DSL (ou adaptée) pour $\omega_{\alpha,\beta}^{(1)}$. Ainsi, si plusieurs super-nœuds de niveau 1 s'attirent fortement, ils peuvent se fondre en un **cluster** $\mathcal{C}_\gamma^{(1)}$ de taille supérieure, que l'on réifie au niveau 2, et ainsi de suite.

Ce **flux ascendant** se répète itérativement. Au niveau k , on identifie des *clusters* $\{\mathcal{C}_\alpha^{(k)}\}$. Chaque $\mathcal{C}_\alpha^{(k)}$ devient un super-nœud $\mathcal{N}_\alpha^{(k+1)}$. Les pondérations $\omega_{\alpha,\beta}^{(k+1)}$ entre super-nœuds s'agrègent via Ψ . Ainsi, on converge vers des **macro-nœuds** de plus en plus globaux, caractérisant le niveau K (ultime palier, voire unique macro-nœud).

On peut schématiser la montée d'échelle ainsi :

$$\omega^{(k+1)} = \Gamma_k(\omega^{(k)}),$$

où Γ_k agrège les **clusters** de $\omega^{(k)}$. On définit :

376. Des **clusters** $\mathcal{C}_\alpha^{(k)} \subseteq \{\mathcal{N}_1^{(k)}, \dots\}$.

377. Les pondérations $\omega_{\alpha,\beta}^{(k+1)}$ via

$$\omega_{\alpha,\beta}^{(k+1)} = \Psi\left(\left\{\omega_{\alpha',\beta'}^{(k)}, \mid \alpha' \in \mathcal{C}_\alpha^{(k)}, \beta' \in \mathcal{C}_\beta^{(k)}\right\}\right).$$

Ici, $\mathcal{C}_\alpha^{(k)}$ désigne un regroupement au **niveau** k . Après agrégation, on obtient un ensemble de “super-nœuds” $\{\mathcal{N}_\alpha^{(k+1)}\}$. Si le SCN se **stabilise**, on pourra aboutir à un nombre réduit de super-nœuds (voire un unique macro-nœud).

Ce **flux ascendant** s'accompagne d'un **gain** en :

1) Réduction de la taille

Si au niveau 0 il existe $O(n^2)$ liens, le niveau 1 n'en comporte plus que $O(m^2)$ (si $m \ll n$), libérant une partie de la charge de calcul et facilitant un traitement plus rapide au palier supérieur.

2) Hiérarchie fonctionnelle

Chaque **cluster local** se spécialise (regroupe des entités proches), puis devient un super-nœud dans l'étape supérieure, reflétant un **rôle** plus large. La progression micro→macro donne une **vision** d'ensemble en successive agrégation.

3) Couplage avec le flux descendant

Les macro-nœuds, une fois construits, peuvent influer sur les règles locales (chap. 6.4.1.2). Ce couplage (bottom-up + top-down) renforce l'auto-organisation hiérarchique.

Conclusion

Le **flux ascendant** (entités → cluster local → cluster macro) constitue la pierre angulaire d'un **SCN** multi-niveau. Il **agrège** les entités élémentaires ou les super-nœuds déjà formés en **structures** de plus en plus étendues, tout en assurant la **cohérence** interne à chaque échelle. D'un point de vue **mathématique**, cela se traduit par un *coarse-graining* itératif : on isole des clusters, on en fait des “super-nœuds”, puis on recalcule les pondérations agrégées $\omega^{(k+1)}$. D'un point de vue **opérationnel**, cela **réduit** considérablement la complexité et prépare le terrain à un **contrôle** top-down (6.4.1.2) qui viendra affiner ou réorienter la dynamique au niveau micro.

6.4.1.2. Étude Mathématique Approfondie : Flux Descendants (Cluster Macro → Entités) et Contraintes sur les Pondérations

La construction hiérarchique de *clusters* en flux ascendant (cf. 6.4.1.1) ne suffit pas à elle seule pour assurer la cohérence globale d'un **SCN** (Synergistic Connection Network) à plusieurs niveaux. Il est souvent nécessaire que les

niveaux supérieurs — où sont agrégés de larges super-nœuds ou clusters macro — puissent **redescendre** des signaux ou contraintes vers les liaisons $\omega_{i,j}$ du niveau inférieur, afin de réorienter ou de stabiliser la dynamique locale. Cette section expose une formulation analytique de ce **flux descendant**, montre comment l'inclure dans l'équation de mise à jour du **DSL** et discute son rôle dans la cohérence globale.

A. Cadre Mathématique et Définition du Flux Descendant

On considère un **SCN** multi-niveau indexé par $k = 0, 1, \dots, K$. Le **niveau 0**, ou micro-niveau, rassemble les **entités élémentaires** $\{\mathcal{E}_i\}$. À chaque palier k , on définit un ensemble de *super-nœuds* $\{\mathcal{N}_\alpha^{(k)}\}$. On note $\omega_{\alpha,\beta}^{(k)}$ les liaisons entre ces super-nœuds au niveau k .

Le **flux descendant** part du niveau k (macro ou intermédiaire) pour imposer un **feedback** sur le niveau $k - 1$. Si $\mathcal{N}_\alpha^{(k)}$ se décompose en un ensemble de super-nœuds du niveau $k - 1$, noté $\mathcal{C}_\alpha^{(k-1)} \subseteq \{\mathcal{N}_\gamma^{(k-1)}\}$, alors le niveau k peut envoyer un **terme** correctif dans la mise à jour des pondérations $\omega_{i,j}^{(k-1)}$.

Dans un DSL purement local, on a généralement

$$\omega_{i,j}^{(k-1)}(t+1) = \omega_{i,j}^{(k-1)}(t) + \eta_{k-1} [S_{k-1}(i,j) - \tau_{k-1} \omega_{i,j}^{(k-1)}(t)],$$

où η_{k-1}, τ_{k-1} sont des paramètres (taux d'apprentissage, facteur d'oubli) et $S_{k-1}(i,j)$ décrit la synergie locale entre les super-nœuds (ou entités) i, j au niveau $k - 1$.

Pour tenir compte du **flux descendant**, on **rajoute** un terme $\gamma_{\text{td}} h_{i,j}^{(k)}(t)$ dans cette équation :

$$\omega_{i,j}^{(k-1)}(t+1) = \omega_{i,j}^{(k-1)}(t) + \eta_{k-1} [S_{k-1}(i,j) - \tau_{k-1} \omega_{i,j}^{(k-1)}(t)] + \gamma_{\text{td}} h_{i,j}^{(k)}(t).$$

Le **signal descendant** $h_{i,j}^{(k)}(t)$ est défini par le niveau k . Il reflète par exemple la volonté de **renforcer** ou **d'affaiblir** certains liens $\omega_{i,j}$ au niveau $k - 1$ pour rendre cohérente une macro-structure pressentie au palier k . Le coefficient $\gamma_{\text{td}} > 0$ règle l'intensité globale de cette rétroaction.

Exemples de Signaux Descendants

- **Renforcement** : si le niveau k décide de fusionner deux super-nœuds $\mathcal{N}_\alpha^{(k)}, \mathcal{N}_\beta^{(k)}$, il peut fixer $h_{i,j}^{(k)}(t) = \delta^+$ (un scalaire positif) pour tous les liens (i,j) situés “sous” ces deux macro-nœuds. Ainsi, la mise à jour $\omega_{i,j}^{(k-1)}$ devient positivement biaisée et encourage leur croissance.
- **Séparation** : si le macro-niveau veut scinder un bloc trop hétérogène, il peut imposer un $h_{i,j}^{(k)}(t) = \delta^- < 0$ sur les liens (i,j) internes, incitant la dynamique locale à **affaiblir** $\omega_{i,j}$.

B. Formulation Analytique du Feedback : Un Modèle

Soit un niveau k avec super-nœuds $\mathcal{N}_\alpha^{(k)}$. Chacun $\mathcal{N}_\alpha^{(k)}$ recouvre un ensemble $\mathcal{C}_\alpha^{(k-1)} \subseteq \{\mathcal{N}_\gamma^{(k-1)}\}$. De la même manière, un super-nœud $\mathcal{N}_\gamma^{(k-1)}$ recouvre lui-même des entités d'un niveau antérieur, jusqu'au micro-niveau si on poursuit le dépliage. Alors

$$h_{i,j}^{(k)}(t) = \theta_{\alpha,\beta}^{(k)}(t) \quad \text{dès que } i \in \mathcal{N}_\alpha^{(k)}, j \in \mathcal{N}_\beta^{(k)}.$$

En d'autres termes, le signal descendant agit sur *toutes* les paires (i,j) dont les entités relèvent des mêmes macro-nœuds α et β au niveau k .

On obtient

$$\Delta \omega_{i,j}^{(k-1)}(t) = \eta_{k-1} [S_{k-1}(i,j) - \tau_{k-1} \omega_{i,j}^{(k-1)}(t)] + \gamma_{\text{td}} \theta_{\alpha,\beta}^{(k)}(t).$$

Même si la synergie locale $S_{k-1}(i,j)$ est faible, un *terme macro* positif $\theta_{\alpha,\beta}^{(k)}$ peut maintenir ou renforcer $\omega_{i,j}$. Inversement, un signal négatif peut annihiler $\omega_{i,j}$. Le niveau macro **façonne** ainsi la structure locale pour parvenir à un objectif d'ensemble.

C. Existence de Contraintes et Cohérence Globale

Dans de nombreux scénarios, imposer par le niveau macro un *minimum* ou un *maximum* à la somme des $\omega_{i,j}$ sur un bloc est réécrivable sous forme d'un *multiplicateur de Lagrange*. Pour un cluster macro $\mathcal{M}_\alpha^{(k)}$, on peut vouloir

$$\sum_{i \in \mathcal{M}_\alpha^{(k-1)}, j \in \mathcal{M}_\beta^{(k-1)}} \omega_{i,j}^{(k-1)} \geq T_{\alpha,\beta}.$$

Si on transcrit ceci en une maximisation (ou minimisation) du potentiel DSL soumis à cette contrainte, la **solution optimale** introduit un $\lambda_{\alpha,\beta} > 0$, qui s'avère *équivalent* à un signal descendant dans $\Delta \omega_{i,j}^{(k-1)}$. En somme, la rétroaction top-down se formalise comme la réponse d'un problème constraint cherchant à préserver (ou atteindre) un lien global entre \mathcal{M}_α et \mathcal{M}_β .

Pour intégrer cette contrainte dans un problème de minimisation (ou de maximisation) d'un **potentiel DSL** $V(\omega)$, on forme la fonction Lagrangienne

$$\mathcal{L}(\omega, \lambda_{\alpha,\beta}) = V(\omega) - \lambda_{\alpha,\beta} \left(\sum_{i \in \mathcal{M}_\alpha^{(k-1)}, j \in \mathcal{M}_\beta^{(k-1)}} \omega_{i,j}^{(k-1)} - T_{\alpha,\beta} \right)$$

où $\lambda_{\alpha,\beta}$ est le multiplicateur de Lagrange associé à cette contrainte, avec $\lambda_{\alpha,\beta} > 0$ pour une contrainte d'inégalité. L'optimisation consiste à trouver ω et $\lambda_{\alpha,\beta}$ tels que les dérivées partielles de \mathcal{L} par rapport à ces variables soient nulles, c'est-à-dire

$$\frac{\partial \mathcal{L}}{\partial \omega_{i,j}^{(k-1)}} = 0 \quad \text{pour tout } i, j \quad \text{et} \quad \frac{\partial \mathcal{L}}{\partial \lambda_{\alpha,\beta}} = 0.$$

La dérivée par rapport à $\lambda_{\alpha,\beta}$ impose directement la contrainte

$$\sum_{i \in \mathcal{M}_\alpha^{(k-1)}, j \in \mathcal{M}_\beta^{(k-1)}} \omega_{i,j}^{(k-1)} = T_{\alpha,\beta},$$

dans le cas limite où la contrainte est active. La dérivée par rapport à $\omega_{i,j}^{(k-1)}$ fournit une condition d'optimalité qui lie la variation du potentiel $V(\omega)$ aux pondérations elles-mêmes et à $\lambda_{\alpha,\beta}$.

Calculons cette dérivée. La fonction \mathcal{L} se décompose en deux parties : le potentiel $V(\omega)$ et le terme de contrainte multiplié par $\lambda_{\alpha,\beta}$. Ainsi, pour un lien donné, la dérivée partielle s'écrit :

$$\frac{\partial \mathcal{L}}{\partial \omega_{i,j}^{(k-1)}} = \frac{\partial V(\omega)}{\partial \omega_{i,j}^{(k-1)}} - \lambda_{\alpha,\beta} \frac{\partial}{\partial \omega_{i,j}^{(k-1)}} \left(\sum_{p \in \mathcal{M}_\alpha^{(k-1)}, q \in \mathcal{M}_\beta^{(k-1)}} \omega_{p,q}^{(k-1)} - T_{\alpha,\beta} \right).$$

Notons que la somme

$$\sum_{p \in \mathcal{M}_\alpha^{(k-1)}, q \in \mathcal{M}_\beta^{(k-1)}} \omega_{p,q}^{(k-1)}$$

est linéaire par rapport à chaque variable $\omega_{i,j}^{(k-1)}$. Par conséquent, la dérivée partielle de cette somme par rapport à $\omega_{i,j}^{(k-1)}$ est simplement 1 (lorsque $i \in \mathcal{M}_\alpha^{(k-1)}$ et $j \in \mathcal{M}_\beta^{(k-1)}$); de plus, la dérivée de la constante $T_{\alpha,\beta}$ est nulle. On obtient donc :

$$\frac{\partial \mathcal{L}}{\partial \omega_{i,j}^{(k-1)}} = \frac{\partial V(\omega)}{\partial \omega_{i,j}^{(k-1)}} - \lambda_{\alpha,\beta}.$$

Pour que cette dérivée soit nulle, il faut que

$$\frac{\partial V(\omega)}{\partial \omega_{i,j}^{(k-1)}} = \lambda_{\alpha,\beta}.$$

ce qui implique que l'augmentation de la pondération $\omega_{i,j}^{(k-1)}$ doit être compensée par un signal descendant proportionnel à $\lambda_{\alpha,\beta}$. En d'autres termes, le multiplicateur $\lambda_{\alpha,\beta}$ agit comme un **signal rétroactif top-down** qui ajuste les mises à jour locales $\Delta \omega_{i,j}^{(k-1)}$ afin de satisfaire la contrainte globale imposée sur le cluster.

Ce formalisme montre que la rétroaction top-down dans le DSL peut être interprétée comme la solution optimale d'un problème d'optimisation sous contrainte. Le **multiplicateur de Lagrange** $\lambda_{\alpha,\beta}$ introduit une pénalité ou une récompense qui guide la mise à jour des liens pour atteindre ou préserver un niveau de cohésion global, exprimé par le seuil $T_{\alpha,\beta}$. Autrement dit, si les interactions entre les clusters doivent dépasser un certain seuil pour garantir la formation d'un super-cluster cohérent, le multiplicateur $\lambda_{\alpha,\beta}$ se positionne de manière à influencer les mises à jour locales, assurant ainsi la **rétroaction** nécessaire pour maintenir la structure hiérarchique du réseau.

Une fois ce **terme** $\theta_{\alpha,\beta}^{(k)}$ inséré, on peut étudier la stabilité de la dynamique couplée (macro + micro). On obtiendra en général un **point fixe hiérarchique** si le flux descendant n'est ni trop fort ni en contradiction permanente avec la synergie locale. L'analyse linéaire (ou non-linéaire) autour d'un état stationnaire montrerait comment ce *terme descendant* modifie les conditions d'équilibre.

D. Conséquences Dynamiques : Stabilisation ou Oscillation

1) Stabilisation Accélérée

Lorsque le niveau macro repère que deux blocs $\mathcal{M}_\alpha^{(k)}, \mathcal{M}_\beta^{(k)}$ devraient logiquement se rapprocher (ou se fusionner) selon un macro-critère, il envoie un **feedback positif** $\theta_{\alpha,\beta}^{(k)}(t) > 0$. Cela force la consolidation au niveau $k - 1$ (voir ex. 5.1 dans 6.4.1.2). La **convergence** peut alors se faire plus vite, le macro-niveau évitant des tâtonnements au palier inférieur.

2) Oscillations ou Conflits

Une rétroaction trop massive ou contradictoire engendre possiblement des **oscillations**. Le **niveau micro** cherche à suivre sa *synergie intrinsèque*, pendant que le macro-niveau force un *schéma* incompatible. On peut ainsi observer un **cycle** si le feedback inverse le sens local à un moment inopportun, tant que la synergie micro n'a pas encore atteint un état stable. Ce se traduit par une valeur propre (ou un module > 1) dans le **jacobien** du système couplé, signe d'un régime oscillatoire.

E. Exemple Pratique de Fusion Macro

Considérons deux **macro-clusters** $\mathcal{M}_\alpha^{(k)}, \mathcal{M}_\beta^{(k)}$ au niveau k . On souhaite les "unir" pour constituer un plus grand bloc, mais au palier $k - 1$, leurs entités (i, j) ne sont pas encore suffisamment reliées. Le macro-niveau définit alors :

$$\theta_{\alpha,\beta}^{(k)}(t) = \delta^+ > 0,$$

et $\theta_{\alpha',\beta'}^{(k)} = 0$ pour tout autre couple (α', β') . Automatiquement, la formule

$$\Delta \omega_{i,j}^{(k-1)}(t) = \eta_{k-1} [S_{k-1}(i,j) - \tau_{k-1} \omega_{i,j}(t)] + \gamma_{\text{td}} \delta^+$$

se retrouve > 0 pour (i,j) appartenant à $\mathcal{M}_\alpha^{(k-1)} \times \mathcal{M}_\beta^{(k-1)}$. En itérant, ces liens $\omega_{i,j}$ se renforcent jusqu'à atteindre un point stable élevé, *verrouillant* (lock-in) la fusion macro souhaitée.

Pour analyser l'état d'équilibre, supposons que la dynamique converge vers une valeur stable $\omega_{i,j}^*$. À l'équilibre, nous avons

$$\omega_{i,j}^* = \omega_{i,j}^* + \eta_{k-1} [S_{k-1}(i,j) - \tau_{k-1} \omega_{i,j}^*] + \gamma_{\text{td}} \delta^+.$$

En simplifiant, nous obtenons

$$\eta_{k-1} [S_{k-1}(i,j) - \tau_{k-1} \omega_{i,j}^*] + \gamma_{\text{td}} \delta^+ = 0.$$

Isolons $\omega_{i,j}^*$:

$$S_{k-1}(i,j) - \tau_{k-1} \omega_{i,j}^* = -\frac{\gamma_{\text{td}} \delta^+}{\eta_{k-1}},$$

ce qui conduit à

$$\omega_{i,j}^* = \frac{S_{k-1}(i,j)}{\tau_{k-1}} + \frac{\gamma_{\text{td}} \delta^+}{\eta_{k-1} \tau_{k-1}}.$$

Cette expression montre que l'équilibre atteint par la pondération $\omega_{i,j}^{(k-1)}$ est supérieur à celui que l'on aurait obtenu sans la rétroaction macro (c'est-à-dire sans le terme $\gamma_{\text{td}} \delta^+$). Le terme supplémentaire $\frac{\gamma_{\text{td}} \delta^+}{\eta_{k-1} \tau_{k-1}}$ est positif, ce qui force la pondération à converger vers une valeur élevée. En conséquence, lorsque l'on itère la mise à jour, les liens entre les entités des sous-clusters qui constituent les macro-clusters $\mathcal{M}_\alpha^{(k)}$ et $\mathcal{M}_\beta^{(k)}$ se renforcent progressivement jusqu'à ce qu'ils atteignent ce niveau élevé, assurant ainsi la fusion verrouillée des macro-clusters. Ce phénomène de **lock-in** garantit que, dès lors que le signal macro est positif, la rétroaction descendante ajuste la dynamique locale de manière à fusionner de manière cohérente les clusters concernés.

Ainsi, la **rétroaction top-down** s'avère indispensable pour qu'un SCN multi-niveau *ne se cantonne pas* à un simple auto-apprentissage local, mais bénéficie d'une **vision** plus large, garantissant une **cohérence** entre les micro-liens et les *objectifs* ou configurations macro. Cette connexion "macro → micro" rend possible une véritable **hiérarchie** où *chaque* palier est actif, et non seulement un agrégateur passif (flux ascendant).

6.4.1.3. Rôle crucial de la cohérence pour éviter le conflit entre niveaux

Les **flux ascendants** (bottom-up) et **descendants** (top-down) (cf. 6.4.1.1 et 6.4.1.2) assurent la double direction de la hiérarchie dans un **SCN** (Synergistic Connection Network) à plusieurs échelles. Cependant, l'existence simultanée de ces deux flux n'est pas exempte de difficultés. Un **conflit** peut survenir si les décisions ou la logique d'auto-organisation d'un **niveau** (micro ou macro) viennent s'opposer de manière répétée à celles d'un autre. Pour maintenir la **stabilité** et permettre une véritable coordination multi-niveau, on a besoin d'une **cohérence**, c'est elle qui garantit que les modifications imposées en haut (macro) et la dynamique locale en bas (micro) ne se **détruisent** pas mutuellement.

A. Notion de "Conflit" Entre Niveaux

Dans la formulation du **DSL** (Deep Synergy Learning) au niveau micro (ou local), des entités $\{\mathcal{E}_i\}$ se regroupent en clusters via la règle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$, appuyée par des seuils ou conditions de cohésion. Le flux ascendant envoie au niveau supérieur l'information "tel sous-ensemble forme un cluster local stable".

Le **niveau macro** peut, inversement, vouloir briser ou fusionner ces clusters pour un objectif global. S'il identifie qu'un ensemble local \mathcal{C} est "mal aligné" avec le schéma plus vaste, il envoie un signal descendant pour casser la cohésion $\sum_{i,j \in \mathcal{C}} \omega_{i,j}$. Si cette décision contrarie trop fortement la synergie locale $S(i,j)$, un **conflit** naît. On a alors, par exemple, un **macro-niveau** dictant $\theta_{\text{desc}} < 0$ sur tous les liens de \mathcal{C} , tandis que localement, $\omega_{i,j}$ continue d'augmenter sous l'effet d'une forte affinité.

Sans un **alignement** suffisant, le micro-niveau peut "verrouiller" un cluster alors que le macro-niveau tente de le disloquer, causant des **oscillations** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \gamma_{\text{topdown}} h_{i,j}^{(\text{macro})}(t).$$

Si γ_{topdown} est trop grand et s'oppose en permanence à la synergie $S(i,j)$, on peut assister à un "tir à la corde" où $\omega_{i,j}$ ne converge pas, passant successivement au-dessus et au-dessous du seuil critique.

B. Formulation Mathématique du Conflit et de la Cohérence

Dans un SCN hiérarchique, on définit les pondérations :

$$\omega_{\alpha,\beta}^{(k)}(t+1) = \omega_{\alpha,\beta}^{(k)}(t) + \eta_k [S_k(\alpha, \beta) - \tau_k \omega_{\alpha,\beta}^{(k)}(t)] + \Delta_{\text{desc}}^{(k)}(\omega^{(k+1)}),$$

pour le niveau k . La fonction $\Delta_{\text{desc}}^{(k)}$ correspond au flux descendant venant de $\omega^{(k+1)}$. Parallèlement,

$$\omega_{\alpha,\beta}^{(k+1)}(t) = \Gamma_k (\omega^{(k)}(t)) \quad (\text{agrégation ascendante}).$$

Si la *direction* imposée par $\Delta_{\text{desc}}^{(k)}$ contredit la logique locale $S_k(\alpha, \beta)$, il y a risque de boucle contradictoire.

Un moyen de **limiter** la discorde est de postuler une **contrainte** du type

$$\|\Delta_{\text{desc}}^{(k)}(\omega^{(k+1)})\| \leq \alpha \|\eta_k [S_k(\cdot) - \tau_k \cdot]\|,$$

avec $\alpha < 1$. Cela impose que le flux descendant ne soit pas plus grand (en norme) que la mise à jour locale. Sur un plan conceptuel, cela **force** le macro-niveau à *respecter* la logique micro au lieu de la balayer, prévenant les conflits.

Nous démontrons ci-dessous, de manière rigoureuse, en quoi cette contrainte garantit la cohérence de la dynamique.

Pour chaque paire (α, β) considérée, la mise à jour totale au niveau k est la somme du terme local et du flux descendant :

$$U_{\text{total}}^{(k)} = \eta_k [S_k(\alpha, \beta) - \tau_k \omega_{\alpha,\beta}^{(k)}(t)] + \Delta_{\text{desc}}^{(k)}(\omega^{(k+1)}).$$

Supposons que la norme du terme local soit notée

$$\|U^{(k)}\| = \|\eta_k [S_k(\alpha, \beta) - \tau_k \omega_{\alpha,\beta}^{(k)}(t)]\|.$$

La contrainte imposée se traduit par

$$\|\Delta_{\text{desc}}^{(k)}\| \leq \alpha \|U^{(k)}\| \quad \text{avec } \alpha < 1.$$

Par l'inégalité triangulaire, nous avons

$$\|U_{\text{total}}^{(k)}\| \geq \|U^{(k)}\| - \|\Delta_{\text{desc}}^{(k)}\| \geq \|U^{(k)}\| (1 - \alpha).$$

Comme $1 - \alpha > 0$, il s'ensuit que le signe (et par conséquent la direction) du terme total est dominé par celui du terme local. En d'autres termes, la rétroaction descendante ne peut renverser la logique de la mise à jour locale si sa contribution reste inférieure à une fraction déterminée de celle-ci. Cette propriété garantit que le mécanisme de mise à jour au niveau k continue à évoluer dans la direction dictée par $S_k(\alpha, \beta)$, évitant ainsi la formation de boucles contradictoires.

Nous pouvons analyser ce phénomène en considérant deux cas extrêmes. D'une part, si la rétroaction descendante est parfaitement nulle, la mise à jour est entièrement dictée par la logique locale, et le système converge vers l'équilibre défini par $S_k(\alpha, \beta) = \tau_k \omega_{\alpha, \beta}^{(k)}(t)$. D'autre part, si la rétroaction descendante devient trop imposante (c'est-à-dire que $\|\Delta_{\text{desc}}^{(k)}\|$ dépasse la limite fixée par la contrainte), la direction du signal total pourrait être inversée par rapport à celle du terme local, conduisant potentiellement à une oscillation ou à une instabilité du système. Le choix de $\alpha < 1$ est donc essentiel pour forcer la hiérarchie à respecter la logique locale, en empêchant le niveau macro de submerger la dynamique micro.

Dans ce contexte, la contrainte agit comme un **filtre** ou un **modulateur** qui assure que l'influence descendante reste subordonnée à l'influence locale. Cela permet d'obtenir une convergence stable du système et d'éviter les comportements indésirables, tels que des cycles oscillatoires ou un comportement chaotique, qui pourraient résulter d'une rétroaction trop forte ou mal synchronisée.

Si, malgré tout, on laisse γ_{topdown} ou $\Delta_{\text{desc}}^{(k)}$ devenir très imposant, le système peut :

- **Osciller** (cycle),
- **Stagner** dans un compromis inutile,
- Produire un **comportement chaotique** si la rétroaction n'est pas phasée.

C. Stratégies d'Évitement du Conflit

Une **solution** est d'**ordonner** clairement les phases d'apprentissage local et les phases de correction macro. Par exemple, on laisse le micro-niveau faire θ_1 itérations, puis le macro-niveau intervient, puis on reboucle. Cela évite que le feedback macro ne se déclenche en même temps que les dynamiques locales, réduisant les risques d'oscillation.

On peut calibrer η_{micro} et γ_{topdown} de sorte que le niveau macro ne renverse pas violemment la cohésion locale. On peut également intégrer un *filtrage* :

$$h_{i,j}^{(\text{macro})}(t+1) = \lambda h_{i,j}^{(\text{macro})}(t) + (1 - \lambda) \tilde{h}_{i,j}^{(\text{macro})}(t),$$

où \tilde{h} est le signal brut. Ainsi, on lisse la correction descendante, évitant des secousses brusques.

Dans certains *algorithmes DSL*, on peut surveiller la **norme** d'évolution $\|\Delta \omega\|$. Si on détecte une amplitude de mises à jour trop forte signifiant un conflit, on restreint temporairement la rétroaction top-down ou on revoit les paramètres d'agrégation ascendante pour calmer la discorde.

Considérons un SCN hiérarchique dans lequel la mise à jour des pondérations au niveau local (micro-niveau) est effectuée selon la dynamique classique

$$\omega(t+1) = \omega(t) + \eta_{\text{micro}} [S_{\text{local}}(\omega(t)) - \tau_{\text{micro}} \omega(t)],$$

où S_{local} représente le signal de synergie issu des interactions locales. Supposons que l'on laisse le micro-niveau opérer pendant θ_1 itérations consécutives, ce qui conduit à une certaine convergence ou stabilisation locale. Après ces θ_1 itérations, un niveau macro intervient en fournissant un signal de rétroaction, que nous notons $\Delta\omega_{\text{topdown}}$. L'ordonnancement séquentiel consiste alors à définir une alternance temporelle, telle que :

$$\begin{aligned} \text{Pour } t \in [t_0, t_0 + \theta_1]: \quad \omega(t+1) &= \omega(t) + \eta_{\text{micro}} [S_{\text{local}}(\omega(t)) - \tau_{\text{micro}} \omega(t)], \\ \text{Pour } t = t_0 + \theta_1: \quad \omega(t+1) &= \omega(t) + \Delta\omega_{\text{topdown}}, \end{aligned}$$

puis le cycle se répète. Cette phase de séparation temporelle garantit que le micro-niveau atteint d'abord un état quasi-stationnaire avant que le macro-niveau n'intervienne, évitant ainsi que le signal top-down ne vienne perturber simultanément la dynamique locale.

La stabilité du système dépend fortement du calibrage des paramètres η_{micro} et γ_{topdown} (ce dernier étant le coefficient associée à la rétroaction macro). Pour empêcher le signal macro de renverser violemment la cohésion locale, il convient d'exiger que, pour tout instant t , la norme du signal de rétroaction $\|\Delta\omega_{\text{topdown}}(t)\|$ soit limitée par rapport à la norme de la mise à jour locale. Formellement, on impose la contrainte

$$\|\Delta\omega_{\text{topdown}}(t)\| \leq \alpha \|\eta_{\text{micro}} [S_{\text{local}}(\omega(t)) - \tau_{\text{micro}} \omega(t)]\|, \quad \text{avec } \alpha < 1.$$

Cette inégalité assure que, quelle que soit l'amplitude du signal macro, sa contribution reste strictement inférieure à celle du mécanisme local. En d'autres termes, même en présence d'un feedback top-down, la mise à jour totale

$$\Delta\omega(t) = \eta_{\text{micro}} [S_{\text{local}}(\omega(t)) - \tau_{\text{micro}} \omega(t)] + \Delta\omega_{\text{topdown}}(t)$$

vérifie, d'après l'inégalité triangulaire,

$$\begin{aligned} \|\Delta\omega(t)\| &\geq \|\eta_{\text{micro}} [S_{\text{local}}(\omega(t)) - \tau_{\text{micro}} \omega(t)]\| - \|\Delta\omega_{\text{topdown}}(t)\| \geq (1 - \alpha) \\ &\|\eta_{\text{micro}} [S_{\text{local}}(\omega(t)) - \tau_{\text{micro}} \omega(t)]\|. \end{aligned}$$

Ainsi, la direction de la mise à jour reste dominée par le signal local, ce qui empêche le macro-niveau d'imposer un changement radical dans la cohésion locale.

Afin d'éviter des variations brusques du signal macro, il est judicieux d'intégrer un mécanisme de filtrage qui lisse la rétroaction descendante. Pour ce faire, on définit un signal filtré $h_{i,j}^{(\text{macro})}(t)$ qui se met à jour selon la relation de lissage exponentiel

$$h_{i,j}^{(\text{macro})}(t+1) = \lambda h_{i,j}^{(\text{macro})}(t) + (1 - \lambda) \tilde{h}_{i,j}^{(\text{macro})}(t),$$

où $\tilde{h}_{i,j}^{(\text{macro})}(t)$ représente le signal top-down brut, et $\lambda \in (0,1)$ est le coefficient de lissage. Ce procédé agit comme un **passe-bas** qui atténue les fluctuations rapides du signal macro, assurant que la correction descendante se fasse de manière progressive. Par conséquent, la mise à jour effective devient

$$\Delta\omega_{\text{topdown,eff}}(t) = \gamma_{\text{topdown}} h_{i,j}^{(\text{macro})}(t),$$

ce qui garantit que les variations brusques du feedback sont atténuées par le facteur $(1 - \lambda)$.

En complément des stratégies précédentes, il est possible d'introduire un mécanisme de surveillance de la norme d'évolution des pondérations, notée $\|\Delta\omega(t)\|$. L'objectif est de détecter en temps réel des mises à jour excessivement importantes, qui signaleraient un conflit entre le signal local et le feedback macro. Si, à un instant donné, l'on observe que

$$\|\Delta\omega(t)\| > \varepsilon_{\text{critique}},$$

où $\varepsilon_{\text{critique}}$ est un seuil fixé en fonction des caractéristiques du système, alors une procédure de contrôle est déclenchée, pouvant consister en la réduction temporaire de γ_{topdown} ou en l'ajustement des paramètres d'agrégation ascendante. Ce mécanisme adaptatif permet de ramener la dynamique dans la zone de stabilité définie par la contrainte ci-dessus.

D. Intérêt Fondamental de la Cohérence

Convergence Efficace. Une cohérence multi-niveau bien gérée garantit une **convergence** plus rapide. Le flux descendant consolide ou oriente ce qui remonte au lieu de le déstabiliser. Au final, moins d'itérations suffisent pour qu'un **cluster** local se confirme ou se dissolve selon les vœux macro.

Robustesse. Sans cohérence, un petit aléa peut dégénérer en conflit entre étages, produisant des effets chaotiques ou des réorganisations incessantes. Avec cohérence, on obtient une **résilience** : si un bloc local est perturbé, le macro-niveau agit pour le replacer, sans entrer dans une lutte contradictoire.

Cas d'École : Applications Cognitives ou Robotiques

- **Cognitif** : Le *macro-concept* (aire cérébrale supérieure) incite des groupes de neurones (niveau micro) à se réorganiser, mais ne détruit pas systématiquement leurs associations *stables* ; c'est un **compromis**.
- **Robotique** : Une flotte de robots se coordonne (niveau micro), le commandant (macro) donne une directive globale ; la cohérence évite l'état de confusion où les robots réagissent localement à un plan que l'autorité contredit.

Conclusion

La **cohérence** multi-niveau est *essentielle* pour empêcher que le flux ascendant (clusters locaux) et le flux descendant (contraintes macro) ne se **neutralisent** ou ne **s'entre-détruisent**. Sur le plan **mathématique**, elle requiert un **équilibrage** des forces (paramètres DSL) et, parfois, un **ordonnancement** temporel (phases alternées, filtrage de signaux). Sur le plan **ingénierie**, cela se traduit par :

- Une **régulation** de la puissance top-down,
- Une **synchronisation** ou un **batch** d'itérations locales avant le feedback macro,
- Un **contrôle** d'oscillation détectant un conflit persistant et adaptant la rétroaction.

De cette façon, le **système** multi-niveau demeure *collaboratif* : le **niveau local** opère librement son auto-organisation, et le **niveau macro** vient affiner ou orienter ce processus sans engendrer de blocages ou de cycles conflictuels. Cette cohérence préserve la **dynamique** du DSL de la discorde, assurant un **fonctionnement** hiérarchique *harmonieux* et *robuste*.

6.4.2. Communication Synergiques entre Niveaux

Au sein d'un **SCN** (Synergistic Connection Network) multi-niveau, la **communication** entre paliers (micro, méso, macro) ne se limite pas à de simples flux d'informations, elle s'inscrit dans la **dynamique** d'auto-organisation, où les liens ω (pondérations) sont mis à jour de manière **synchrone** ou **asynchrone** suivant la règle DSL. La notion de "communication synergiques" évoque spécifiquement la façon dont les **clusters** détectés (ou émergents) à un niveau sont **transmis** sous forme d'agrégation vers le niveau supérieur, et comment, réciproquement, ce niveau supérieur valide ou rétroagit sur ces clusters. Dans la section 6.4.2.1, nous examinons particulièrement l'idée selon laquelle un micro-cluster **stabilisé** (au niveau local) peut être "validé" ou "agrégé" au palier macro, illustrant la prise en compte hiérarchique des informations.

6.4.2.1. Si un micro-cluster se stabilise, le niveau macro peut le "valider" ou l'agréger

La dynamique **multi-échelle** au sein d'un **SCN** (Synergistic Connection Network) permet à des **micro-clusters** (niveau local) de se former, puis de "remonter" progressivement vers des niveaux supérieurs (intermédiaire ou macro), conformément à la logique du **flux ascendant** (cf. 6.4.1.1). Une fois qu'un micro-cluster \mathcal{C}_α est jugé **stable** au niveau 0 (ou local), il revient au **niveau macro** (ou intermédiaire) de **valider** cette stabilisation, puis, si nécessaire, de **l'agréger** sous forme d'un super-nœud. Le texte qui suit expose les bases mathématiques de cette validation et en souligne les bénéfices et précautions.

A. Stabilisation au Niveau Micro et Constitution d'un Cluster

Un **micro-cluster** $\mathcal{C}_\alpha \subseteq \{\mathcal{E}_i\}$ se définit généralement comme un groupe d'entités dont les **pondérations** internes $\{\omega_{i,j}^{(0)}\}$ sont élevées. En pratique, on introduit un **critère** de cohésion tel que

$$\Omega(\mathcal{C}_\alpha) = \sum_{i,j \in \mathcal{C}_\alpha} \omega_{i,j}^{(0)} \geq \theta_{loc},$$

où θ_{loc} est un **seuil** local. La dynamique DSL (par exemple de type additive) agit au niveau 0 :

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_0 [S_0(i,j) - \tau_0 \omega_{i,j}^{(0)}(t)],$$

faisant **croître** les liens $\omega_{i,j}^{(0)}$ entre entités localement synergiques, jusqu'à ce qu'un sous-ensemble \mathcal{C}_α dépasse θ_{loc} . On dit alors que \mathcal{C}_α est un **cluster local stabilisé** (voir aussi 6.4.1.1 pour le flux ascendant).

B. Rôle du Niveau Macro : Valider ou Refuser la Stabilisation

Le **niveau macro** (ou supérieur) reçoit l'information qu'un **cluster local** \mathcal{C}_α s'est formé. Deux possibilités se présentent :

378. Le niveau macro **valide** la stabilisation, c'est-à-dire qu'il *accepte* ce micro-cluster comme un bloc fonctionnel.

379. Le niveau macro **rejette** (ou "modère") cette stabilisation, par exemple s'il estime que \mathcal{C}_α interfère avec un **objectif** ou une **cohérence globale** (cf. 6.4.1.3 sur la cohérence).

Dans la première hypothèse, le macro-niveau **agrège** \mathcal{C}_α en un **super-nœud** $\mathcal{N}_\alpha^{(1)}$. On procède à une fonction d'agrégation Ψ (6.2.2) pour définir les liaisons entre $\mathcal{N}_\alpha^{(1)}$ et les autres super-nœuds $\{\mathcal{N}_\beta^{(1)}\}$ déjà reconnus :

$$\omega_{\alpha,\beta}^{(1)} = \Psi(\{\omega_{i,j}^{(0)} \mid i \in \mathcal{C}_\alpha, j \in \mathcal{C}_\beta\}).$$

Le cluster local \mathcal{C}_α devient donc un **nœud de niveau 1**. L'information micro est ainsi "condensée" pour que l'échelle macro puisse la manipuler avec un **nombre réduit** de nœuds. Cette **validation** confère au cluster local un statut "officiel" à l'échelle supérieure.

Si le niveau macro **refuse** la stabilisation locale, il peut renvoyer un **feedback** descendant (6.4.1.2) pour affaiblir les liens $\omega_{i,j}^{(0)}$ du cluster \mathcal{C}_α . Par exemple, un **signal** $\theta_\alpha^{(\text{macro})} < 0$ s'ajoute dans la mise à jour :

$$\Delta \omega_{i,j}^{(0)}(t) = \eta_0 [S_0(i,j) - \tau_0 \omega_{i,j}^{(0)}(t)] + \gamma_{\text{top-down}} \theta_\alpha^{(\text{macro})}.$$

Cela **empêche** le cluster de persister localement et force sa reconfiguration, évitant un conflit durable.

C. Bénéfices de la Validation par le Macro-Niveau

La "validation" d'un micro-cluster par le macro-nœud apporte plusieurs **avantages** dans un SCN multi-échelle :

380. **Réduction de la complexité** : On n'a pas à gérer toutes les entités $\{\mathcal{E}_i\}$ individuellement au niveau macro. Une fois validé, \mathcal{C}_α est résumé par $\mathcal{N}_\alpha^{(1)}$.

381. **Stabilisation hiérarchique** : Le micro-cluster profite de l'**appui** du macro-niveau. En retour, le macro-niveau sait qu'il peut compter sur l'existence de ce bloc local "fiable" pour bâtir des structures plus vastes.

382. **Rapidité de convergence** : Plutôt que d'attendre un consensus global, on **valide** localement et on "remonte" ces blocages fonctionnels. Ainsi, la mise en place d'une structure macro se nourrit immédiatement de la dynamique micro.

D. Exemple : Fusion de Micro-Clusters Réunis

Dans un système DSL robotique (6.1, 6.2), supposons que deux micro-clusters \mathcal{C}_α et \mathcal{C}_β se forment dans des zones proches. Le **macro-niveau** repère qu'ils ont un **objectif** semblable (cohérence spatiale, même mission), et décide de "valider" chacun individuellement, puis de **fusionner** ces deux super-nœuds $\mathcal{N}_\alpha^{(1)}$ et $\mathcal{N}_\beta^{(1)}$ en un mégacluster $\mathcal{N}_{\alpha,\beta}^{(2)}$. Cette validation en chaîne illustre comment le flux ascendant (entités → micro-clusters) s'enchaîne au flux macro (validation → super-nœuds).

Sans l'étape de "confirmation" macro, les micro-clusters pourraient rester dans un état incertain, ou, à l'inverse, un flux descendant contradictoire viendrait contester leur existence sans se manifester explicitement. La validation rend la hiérarchie plus explicite et plus stable.

E. Conclusion

Lorsqu'un **micro-cluster** se **stabilise**, le **niveau macro** a la faculté de le **valider** (et donc de l'**agréger** sous forme d'un super-nœud). Cette reconnaissance ascendante :

383. **Finalise** l'auto-organisation locale en l'inscrivant dans l'échelle supérieure,

384. **Allège** la structure macro, qui n'opère plus sur des entités microscopiques dispersées, mais sur des blocs agrégés,

385. **Réduit** la probabilité de conflits ou de remises en cause ultérieures, puisque le niveau macro "assume" l'existence de ce cluster.

Ce processus forme l'un des points essentiels de la **communication multi-niveau** : le micro détecte et consolide un groupe, puis le macro "scelle" ou "rejette" ce groupe. Avec cette logique, un **SCN** multi-échelle (chap. 6.2) arrive à bâtrir une **hiérarchie** de clusters cohérente et évolutive, tout en laissant place à la flexibilité du flux descendant si l'ensemble validé localement se révèle inadapté à plus long terme.

6.4.2.2. Mécanismes de filtrage : ne pas transmettre toute la synergie brute vers le haut, résumer ou agréger

La **multi-échelle** dans un **SCN** (Synergistic Connection Network) induit une série de **flux ascendants** (chap. 6.4.1) où des entités, d'abord organisées localement (niveau 0), s'agrègent ou se stabilisent sous forme de clusters, puis sont transmises à l'échelle supérieure (méso, macro). Cette **transmission** d'informations soulève immédiatement une question de **volume**. Si l'on transmet la **totalité** des pondérations $\omega_{i,j}$ ou liaisons internes à chaque cluster, le palier supérieur se retrouve surchargé et perd l'avantage de la hiérarchie. D'où la nécessité d'un **filtrage** ou d'une **agrégation** qui, au lieu de transmettre toutes les "synergies brutes", n'en relaie qu'un **résumé** concis et pertinent.

La présente section clarifie les principes formels de ce **filtrage**. D'un point de vue **mathématique**, il s'agit d'une opération Ψ réduisant la dimension du réseau ; d'un point de vue **opérationnel**, cela limite la **masse** d'informations circulant vers le haut, assurant robustesse et économie.

A. Motivation du Filtrage et de l'Agrégation

Si au niveau local n entités forment potentiellement $O(n^2)$ liaisons, transmettre $\omega_{i,j}^{(0)}$ pour toutes les paires (i, j) vers le niveau 1 annulerait l'intérêt d'une **architecture hiérarchique**. Le filtrage permet de **résumer** la masse des liens en un plus petit nombre de **super-pondérations** ou **indicateurs**, de sorte que le niveau macro n'opère que sur $O(m^2)$ valeurs, où $m \ll n$.

Exemple : si $n = 10^5$, on peut réduire à $m = 10^3$ super-nœuds, puis passer de 10^{10} liaisons à 10^6 .

Les dynamiques DSL (ex. la mise à jour additive) génèrent souvent beaucoup de **liens faibles** ou transitoires. Les transmettre « bruts » enliserait le niveau macro dans des détails superflus. Un **filtrage** (seuil, élagage) écarte les liaisons jugées peu contributrices, protégeant l'échelle supérieure contre des variations mineures.

Un cluster local \mathcal{C}_α peut avoir $O(k^2)$ liens internes. En les condensant via une fonction Ψ (moyenne, somme, etc.), on gagne une "seule" valeur $\omega_\alpha^{(1)}$ pour décrire la cohésion interne de \mathcal{C}_α , ce qui **lissee** les aléas locaux. Mathématiquement, la variance de liens internes se "contracte" dans une moyenne, rendant la **signalétique** plus stable aux fluctuations.

B. Principes Formels du Filtrage/Agrégation

Un mécanisme simple :

$$\tilde{\omega}_{i,j}^{(0)} = \begin{cases} \omega_{i,j}^{(0)}, & \text{si } \omega_{i,j}^{(0)} \geq \theta, \\ 0, & \text{sinon.} \end{cases}$$

On ne conserve que les liaisons $\omega_{i,j}$ supérieures à un seuil θ . Ce filtrage peut s'appliquer avant de construire le super-nœud, assurant qu'on n'intègre dans \mathcal{C}_α que les liens réellement “forts”.

Une fois un cluster $\mathcal{C}_\alpha \subseteq \{\mathcal{E}_i\}$ déterminé, la liaison $\omega_{\alpha,\beta}^{(1)}$ entre deux clusters $\mathcal{C}_\alpha, \mathcal{C}_\beta$ se calcule par une **fonction**

$$\omega_{\alpha,\beta}^{(1)} = \Psi\left(\left\{\omega_{i,j}^{(0)} \mid i \in \mathcal{C}_\alpha, j \in \mathcal{C}_\beta\right\}\right).$$

Les formes de Ψ peuvent être :

- **Somme** : $\omega_{\alpha,\beta} = \sum_{i \in \mathcal{C}_\alpha, j \in \mathcal{C}_\beta} \omega_{i,j}.$
- **Moyenne** : $\omega_{\alpha,\beta} = \frac{1}{|\mathcal{C}_\alpha||\mathcal{C}_\beta|} \cdot \sum_{i,j} \omega_{i,j}.$
- **Max ou quantile** : $\omega_{\alpha,\beta} = \max_{i \in \mathcal{C}_\alpha, j \in \mathcal{C}_\beta} \omega_{i,j}.$

Plutôt que de donner un seul nombre, on peut transmettre un **vecteur** de statistiques : {min, max, moyenne, écart-type, ...} décrivant la distribution interne. Le niveau macro reçoit alors une signature plus riche, au prix d'une “explosion” toutefois modérée (quelques indicateurs au lieu de centaines ou milliers de liens).

C. Processus Algorithmique : du Micro au Macro

1. Détection de Clusters Locaux

On commence par identifier $\{\mathcal{C}_\alpha^{(0)}\}_{\alpha=1,\dots,r}$ au niveau micro : chaque $\mathcal{C}_\alpha^{(0)}$ est un **bloc** de forte cohésion $\Omega(\mathcal{C}_\alpha^{(0)}) \geq \theta_{\text{loc}}$.

2. Filtrage

On applique un seuil ou un algorithme d’élargissement pour ne **retenir** que les liens importants au sein de \mathcal{C}_α ou entre \mathcal{C}_α et \mathcal{C}_β .

3. Construction du Super-Nœud

On agrège \mathcal{C}_α en un *super-nœud* $\mathcal{N}_\alpha^{(1)}$. On calcule $\omega_{\alpha,\beta}^{(1)}$, $\forall \beta$, grâce à Ψ .

4. Transmission

On envoie vers le niveau macro le **résultat** : $\{\omega_{\alpha,\beta}^{(1)}\}_{\alpha,\beta}$ ou un *résumé* statistique des liaisons internes et externes. Le macro-niveau n'a pas besoin de toutes les $\omega_{i,j}^{(0)}$, seulement de ce “filtrat”.

D. Impact Pratique : Économie et Stabilité

Allègement. Avec le filtrage, le **niveau macro** ne manie plus $O(n^2)$ poids bruts, mais $O(r^2)$ super-pondérations ($r =$ nombre de clusters $\leq n$). L'économie est dramatique si $r \ll n$.

Robustesse. Les *petites* fluctuations des liens $\omega_{i,j}^{(0)}$ qui n'atteignent pas le seuil θ ne remontent pas. On évite de bouleverser la structure macro pour un “micro-bruit” local. Cela procure une forme de **tolérance** aux perturbations.

Possibilité de Réactualisation. Le filtrage ne doit pas être *irréversible* : un lien *faible* aujourd’hui peut devenir *fort* demain si la synergie $S_0(i,j)$ évolue. Certaines implémentations DSL réévaluent régulièrement les *liens filtrés*, pour ne pas rater l'opportunité d'un futur cluster.

E. Conclusion

Dans un **SCN** multi-niveau, la **transmission** des informations du **niveau micro** vers le **niveau macro** requiert une **synthèse** soignée. Les *mécanismes de filtrage* (seuil, élagage, agrégation Ψ , résumé statistique) jouent un rôle **essentiel** :

386.Ils **limitent l'inondation** de liens bruts,

387.Ils **stabilisent** la représentation macro en n'envoyant que les structures (clusters, liens forts) réellement significatives,

388.Ils **permettent** d'organiser une *communication ascendante* agile et parcimonieuse, rendant le **DSL** multi-niveau robuste et efficace.

Ainsi, on concrétise la philosophie **hiérarchique** (chap. 6.2) : le micro-niveau s'occupe du *détail*, le macro-niveau d'une *vue d'ensemble*, chacun communiquant *juste ce qu'il faut* pour maintenir la cohérence globale (6.4.1.3) et la réactivité locale.

6.4.2.3. Exemples Concrets : Agent Local en Robotique, Superviseur Macro

Il est souvent utile d'illustrer la **communication synergiques** (section 6.4.2) entre un **niveau micro** et un **niveau macro** dans un contexte de **Deep Synergy Learning**. Deux cas d'usage sont présentés : d'une part, un **agent local** en robotique (niveau micro) et, d'autre part, un **superviseur macro** responsable d'une coordination globale. Dans ces deux scénarios, on voit comment un *flux ascendant* condense l'information essentielle (évacuant les détails inutiles) et comment un *flux descendant* fournit un **contrôle** ou une **influence** réciproque sur les liaisons du niveau inférieur.

A. Agent Local en Robotique

Considérons un système multi-agent de type robotique, où chaque **robot** est modélisé comme une **entité** \mathcal{E}_i . Les pondérations $\omega_{i,j}^{(0)}$ relient les paires de robots i et j au **niveau 0**, c'est-à-dire le niveau micro. L'**objectif** du **Deep Synergy Learning** est de permettre une auto-organisation locale ; chaque robot peut adapter ses connexions selon une **règle** de plasticité. Une forme linéaire-additive simple est donnée par

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_{\text{loc}}[S_{\text{loc}}(i,j) - \tau_{\text{loc}} \omega_{i,j}^{(0)}(t)],$$

où $\eta_{\text{loc}} > 0$ est un **taux d'apprentissage** et $\tau_{\text{loc}} > 0$ un **terme de décroissance**. La **fonction de synergie** $S_{\text{loc}}(i,j)$ dépend par exemple de la **distance** entre robots, de leur **similarité** de tâches ou de toute autre mesure favorisant la coopération.

Lorsque ce processus d'**auto-organisation** local aboutit à un **micro-cluster**, noté $\mathcal{C}_\alpha \subset \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$, on peut agréger \mathcal{C}_α en un **super-nœud** $\mathcal{N}_\alpha^{(1)}$. Au **niveau 1**, on ne conserve pas la totalité des liaisons individuelles $\omega_{i,j}^{(0)}$, mais une **agrégation** $\omega_{\alpha,\beta}^{(1)}$ calculée typiquement via

$$\omega_{\alpha,\beta}^{(1)} = \Psi(\{\omega_{i,j}^{(0)} \mid i \in \mathcal{C}_\alpha, j \in \mathcal{C}_\beta\}),$$

où Ψ est une **opération de synthèse**, par exemple une somme, une moyenne pondérée ou un maximum. De la sorte, le **niveau macro** ne voit plus qu'une poignée d'entités $\mathcal{N}_\alpha^{(1)}$, correspondant chacune à un ensemble local cohésif de robots.

Le **flux ascendant** (micro vers macro) se retrouve donc filtré ; on élimine la majorité des détails et on concentre la communication sur un petit nombre de "super-nœuds" issus de l'auto-organisation des agents locaux. Cette logique **multi-niveau** permet de limiter le coût, au lieu de gérer $O(n^2)$ liaisons élémentaires, le **niveau 1** ne manipule que $O(k^2)$ super-liaisons, où k est le nombre de clusters.

B. Superviseur Macro

Dans un second exemple, on suppose l'existence d'un **superviseur macro**, un **niveau supérieur** qui reçoit les **super-nœuds** du niveau 1 (ou un niveau intermédiaire plus haut encore). Ce superviseur gère la **tâche globale** (répartition de zones à explorer, contraintes de mission, etc.) et peut renvoyer un *feedback* vers le micro-niveau pour orienter le renforcement ou l'inhibition de certaines connexions.

Une façon de modéliser ce *feedback descendant* est d'enrichir l'équation de mise à jour micro par un **terme d'influence** Δ_{down} . On peut alors écrire

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_{\text{loc}}[S_{\text{loc}}(i,j) - \tau_{\text{loc}} \omega_{i,j}^{(0)}(t)] + \gamma_{\text{macro}} \Delta_{\text{down}},$$

où γ_{macro} est un **facteur** réglant l'intensité de l'intervention du **superviseur**. Le terme Δ_{down} peut être, par exemple, un **signal** incitant à renforcer les liaisons entre deux sous-groupes \mathcal{C}_α et \mathcal{C}_β si le niveau supérieur décide de fusionner leurs efforts.

Dans ce cas, l'**auto-organisation** demeure largement locale (les agents mettent à jour leurs poids selon une synergie locale S_{loc}), mais un **contrôle** partiel par la structure de plus haut niveau permet d'anticiper les besoins de la mission globale (objectif externe imposé, ou simple consigne d'alignement).

C. Avantages de la Communication Synergiques

L'**agent local** peut s'organiser entre ses pairs de manière autonome, créant ainsi des **micro-clusters** robustes et adaptatifs. Cette information "micro" est alors **remontée au superviseur macro** sous forme condensée (agrégation en super-nœuds), évitant une surcharge $O(n^2)$. Le **superviseur** peut à son tour moduler la configuration du niveau inférieur via un *flux descendant*, soit pour soutenir certaines synergies, soit pour en restreindre d'autres.

Ces mécanismes hiérarchisés assurent à la fois la **scalabilité** (le macro-niveau n'interagit qu'avec un nombre réduit de super-nœuds) et la **réactivité** (chaque groupe local s'auto-organise rapidement selon η_{loc}). Les pondérations finales $\omega_{i,j}^{(0)}$ émanent donc d'une **double influence** : la synergie "réelle" estimée localement et l'**orientation** imposée par le macro-niveau.

Lorsque ce cadre s'applique à l'**industrie** robotique ou à un **système** multi-agent plus générique, on préserve un fort degré de **décentralisation** tout en maintenant un pilotage global cohérent. La communication **synergiques** entre niveaux assure en effet un bon compromis entre l'autonomie des sous-groupes et la coordination stratégique imposée par le sommet de la hiérarchie.

Cette capacité à se reconfigurer, tant localement que globalement, correspond à l'esprit du **Deep Synergy Learning**. Plusieurs *couches* ou *niveaux* interagissent via des **flux** réciproques, filtrés et agrégés, sans pour autant que l'un de ces niveaux ne subisse la totalité des détails issus des autres. Le résultat est un **réseau** où l'intelligence se répartit des agents les plus élémentaires jusqu'aux superviseurs d'échelle plus élevée, ce qui favorise la **robustesse** et l'**adaptation** face aux changements de contexte.

6.4.3. Synchronisation et Clustering

Dans la perspective d'un **DSL** (Deep Synergy Learning) déployé sur un **SCN** (Synergistic Connection Network) multi-niveau, la question de la **synchronisation** entre niveaux et celle du **clustering** progressif se posent en termes concrets :

- *Comment garantir que les divers paliers (micro, méso, macro) ne sont pas en perpétuel décalage ?*
- *Comment gérer la coexistence de clusters à échelles différentes, et quand décider de la stabilisation ou de la fusion de plusieurs regroupements ?*

La section 6.4.3.1 met l'accent sur le rôle d'"échelles intermédiaires" (semi-globales), agissant comme des **étapes** cruciales dans la stabilisation hiérarchique du SCN.

6.4.3.1. Les Clusters d'Échelle Intermédiaire (Semi-Globale) comme Étapes de Stabilisation

Introduction. Le passage d'un niveau **micro** (petits groupes ou entités isolées) à un niveau **macro** (super-nœuds ou regroupements englobants) peut s'avérer trop rapide ou trop complexe si l'on ne tient pas compte de *paliers intermédiaires*. Dans de nombreux schémas multi-niveau, ces paliers prennent la forme de **clusters semi-globaux**, c'est-à-dire des regroupements de taille moyenne. Ces entités intermédiaires facilitent la stabilisation et la cohérence hiérarchique, tout en régulant la montée progressive vers des macro-structures. Les sections qui suivent détaillent le rôle de ces clusters semi-globaux, les mécanismes de synchronisation inter-paliers et divers cas d'utilisation.

A. Échelle Intermédiaire et Stabilisation Progressive

Dans une architecture multi-niveau, un **micro-niveau** peut produire des micro-clusters $\{\mathcal{C}_\alpha^{(0)}\}$. On agrège ensuite ces micro-clusters en un **niveau 1**, où l'on considère des super-nœuds $\mathcal{N}_\alpha^{(1)}$. Lorsque l'échelle totale du système est grande, il est souvent nécessaire de continuer à agréger ces super-nœuds pour former un **niveau 2**, aboutissant à des macro-clusters $\{\mathcal{N}_\beta^{(2)}\}$ de dimension plus élevée. Dans ce contexte, le niveau 1 agit comme un **tampon** ou une **étape pivot**. Il consolide des groupes de taille moyenne, sans prétendre atteindre immédiatement la structure macro complète.

La stabilisation progressive s'explique en termes **d'équilibre dynamique**. Les pondérations $\omega^{(k)}$ au palier k suivent une règle de mise à jour de type

$$\omega_{a,b}^{(k)}(t+1) = \omega_{a,b}^{(k)}(t) + \eta_k [S^{(k)}(a,b) - \tau_k \omega_{a,b}^{(k)}(t)],$$

où η_k et τ_k sont respectivement le **taux d'adaptation** et le **terme de décroissance** associés au niveau k . Il n'est pas nécessaire d'augmenter subitement l'échelle de ces pondérations pour passer directement du micro au macro. Les paliers intermédiaires laissent le temps à chaque niveau d'atteindre un quasi-équilibre local, évitant un chaos dû à un trop grand saut d'échelle.

B. Clusters Semi-Globaux et Régulation de la Complexité

Un cluster semi-global représente un groupe de dimension intermédiaire, par exemple quelques dizaines ou centaines de nœuds sur un total de milliers. Sur le plan **dynamique**, ces regroupements agissent comme un filtre. Les micro-clusters du niveau inférieur $\mathcal{N}_\alpha^{(1)}$ peuvent fusionner ou se scinder pour former des ensembles plus vastes, mais seulement si leur **synergie** demeure assez élevée lors des itérations d'**auto-organisation**. Les clusters semi-globaux ainsi créés ne sont ni trop petits ni trop grands, ce qui facilite leur réorganisation interne et leur éventuelle fusion en un macro-niveau ultérieur.

La compétition ou la collaboration entre clusters semi-globaux peut être modélisée par des pondérations $\omega_{\alpha,\beta}^{(1)}$. Par exemple, si $\mathcal{N}_\alpha^{(1)}$ et $\mathcal{N}_\beta^{(1)}$ exhibent une forte compatibilité, la mise à jour

$$\omega_{\alpha,\beta}^{(1)}(t+1) = \omega_{\alpha,\beta}^{(1)}(t) + \eta_1 [S_1(\alpha,\beta) - \tau_1 \omega_{\alpha,\beta}^{(1)}(t)]$$

peut conduire, à terme, à l'union de ces deux regroupements si cette compatibilité dépasse un certain seuil. Inversement, de faibles valeurs de synergie pousseront à la désactivation ou au relâchement des liens. Les clusters semi-globaux les plus stables constitueront alors le socle pour la **phase suivante** d'agrégation.

C. Synchronisation des Paliers et Ordonnancement Temporel

Les différents niveaux ne doivent pas être mis à jour simultanément à chaque instant, sous peine d'induire des oscillations ou conflits entre le micro et le macro. Une stratégie courante de **synchronisation** consiste à laisser le micro-niveau évoluer pendant un certain nombre d'itérations, puis à figer ses résultats, à construire les clusters de niveau supérieur et à itérer à ce niveau intermédiaire. Ce découpage temporel peut se formaliser par un calendrier en étapes :

389. Le **micro-niveau** (niveau 0) applique la règle de mise à jour durant T_0 itérations, produisant des groupes $\{\mathcal{C}_\alpha^{(0)}\}$.

390. On agrège ces groupes en super-nœuds $\{\mathcal{N}_\alpha^{(1)}\}$. Les pondérations $\omega_{\alpha,\beta}^{(1)}$ sont initialisées par une opération de synthèse (somme, moyenne, etc.) sur les liens intergroupes de niveau 0.

391. Le **niveau intermédiaire** exécute alors la mise à jour pendant T_1 itérations, aboutissant à la formation ou la stabilisation de clusters semi-globaux.

392. Enfin, si nécessaire, un passage au **macro-niveau** (niveau 2) peut se faire après un nombre d'itérations suffisant pour que le niveau 1 se soit **stabilisé**.

Dans cette séquence, chaque échelon (k) transmet ses informations à $(k + 1)$ lorsque le système local est parvenu à un état suffisamment cohérent. Cela évite de multiplier des allers-retours entre des niveaux qui n'ont pas eu le temps de converger.

D. Cas d'Utilisation et Exemples Concrets

Les **clusters semi-globaux** apparaissent dans un large éventail de systèmes multi-niveau. Dans le domaine des réseaux distribués (cloud et edge), le palier intermédiaire peut correspondre à des **data centers régionaux** qui rassemblent les ressources de plusieurs sites. En robotique, le niveau 1 regroupe des sous-équipes locales pour former des unités de taille moyenne, avant de s'associer en un métagroupe macro si une mission globale l'exige. Dans des approches inspirées de la **cognition** ou du **cerveau**, des sous-réseaux peuvent émerger en tant qu'aires corticales locales et coopérer à plus grande échelle par paliers successifs.

Dans tous ces scénarios, la dynamique d'échelle intermédiaire offre une **stabilisation hiérarchique**. Plutôt que de confronter immédiatement l'ensemble des entités à une vision macro, on laisse les synergies locales s'organiser en regroupements robustes. Les paliers semi-globaux constituent ainsi une étape de maturation indispensable, limitant le risque de configurations fragiles ou mal ajustées lorsqu'on atteindra le niveau englobant.

Conclusion

Les **clusters d'échelle intermédiaire** fournissent un mode de **stabilisation progressive** entre l'échelle micro et l'échelle macro. Sur le plan mathématique, cette stabilisation s'exprime par la mise à jour en paliers successifs des pondérations $\omega^{(k)}$, permettant à chaque regroupement d'affiner sa cohésion avant de s'intégrer dans un ensemble plus vaste. Les groupes semi-globaux ne sont ni trop restreints, ni trop imposants, ce qui leur confère la souplesse nécessaire pour évoluer, fusionner ou se scinder au gré de la dynamique d'**auto-organisation**. Ils agissent comme un **filtre** ou un **tampon** essentiel, assurant la montée en échelle sans basculer dans la confusion ni l'explosion combinatoire. Cette hiérarchie en paliers, déjà illustrée dans le chapitre 6.4, émerge de manière naturelle dans les architectures multi-niveau du **Deep Synergy Learning** et se montre très adaptée à des domaines aussi variés que la robotique, les systèmes distribués et l'IA cognitive.

6.4.3.2. Coordination Latérale entre Clusters de Niveaux Proches : Comment Interagir avec le Cluster Voisin

Introduction. Dans un **Deep Synergy Learning** multi-niveau, il est courant de focaliser la communication sur l'axe vertical, que ce soit vers le haut (bottom-up) ou vers le bas (top-down). Toutefois, au sein d'un **même** palier de la hiérarchie (par exemple, un niveau intermédiaire ou semi-global), plusieurs **clusters** peuvent coexister. La question se pose alors de savoir comment ces clusters, qui partagent la même échelle, interagissent **directement** entre eux sans nécessairement faire appel au niveau supérieur ou au niveau micro. Cette **coordination latérale** s'avère cruciale pour résoudre d'éventuels conflits locaux, permettre des fusions partielles ou faciliter des échanges de ressources à l'échelle intermédiaire. Les paragraphes suivants détaillent la nécessité et les modalités de cette interaction horizontale, ainsi que les bénéfices et implications sur la dynamique globale.

A. Raison d'Être de la Coordination Latérale

Au sein d'un palier donné, il est possible que plusieurs super-nœuds $\mathcal{N}_\alpha^{(k)}$ et $\mathcal{N}_\beta^{(k)}$ se retrouvent dans une situation de **complémentarité** ou de **compétition**. Des groupes voisins peuvent occuper des régions adjacentes (dans un contexte de robotique) ou partager des caractéristiques similaires (dans une application de clustering de données), et se retrouvent incités à établir des mécanismes de dialogue direct. L'idée est de gérer en local leurs **relations mutuelles**, plutôt que de solliciter le niveau macro pour chaque ajustement mineur. Cela accélère la réaction à des défis communs et décharge la couche supérieure, qui se concentre alors sur des stratégies plus globales.

S'il n'existe pas de **coordination latérale**, toute communication entre clusters du même palier devrait être relayée à travers le niveau $k + 1$. Un tel détour crée un surcoût, perturbe la réactivité locale et contribue à la surcharge d'information chez le macro-nœud. La mise en place de **liens horizontaux**, avec leurs pondérations $\omega_{\alpha,\beta}^{(k)}$, constitue un moyen efficace de rendre ces échanges plus directs, plus rapides et mieux adaptés à la dimension semi-globale.

B. Formalisation : Liens Latéraux au Même Palier

Au niveau k , chaque super-nœud $\mathcal{N}_\alpha^{(k)}$ est potentiellement connecté aux autres super-nœuds $\mathcal{N}_\beta^{(k)}$. On peut donc définir une **matrice** $\omega^{(k)}$ dont les éléments $\omega_{\alpha,\beta}^{(k)}$ sont sujets à une règle de plasticité du même type que celle utilisée aux niveaux micro. Une loi de mise à jour classique est :

$$\omega_{\alpha,\beta}^{(k)}(t+1) = \omega_{\alpha,\beta}^{(k)}(t) + \eta_k [S_k(\alpha, \beta) - \tau_k \omega_{\alpha,\beta}^{(k)}(t)].$$

Le **taux d'adaptation** η_k et la **constante de décroissance** τ_k peuvent être spécifiques au palier k . La *fonction de synergie* $S_k(\alpha, \beta)$ correspond alors à un **score** reflétant la compatibilité entre les regroupements $\mathcal{N}_\alpha^{(k)}$ et $\mathcal{N}_\beta^{(k)}$. Ce score peut être dérivé de la somme ou de la moyenne des liens inter-entités au niveau $k - 1$, de la proximité spatiale, de la complémentarité de compétences, ou de toute autre donnée permettant d'évaluer l'intérêt de coopérer.

L'existence de ces pondérations $\omega_{\alpha,\beta}^{(k)}$ transforme le palier k en un **réseau** de super-nœuds, qui s'auto-organise selon le même principe DSL : les paires de clusters les plus synergiques renforcent leurs liens, les autres s'étiolent. Les super-nœuds peuvent alors se **rapprocher**, se **fusionner** ou maintenir une simple relation d'information mutuelle. Cette auto-organisation horizontale rend la couche k plus autonome et plus stable face à des phénomènes locaux.

C. Collaboration, Concurrence et Frontières

D'un point de vue dynamique, deux super-nœuds $\mathcal{N}_\alpha^{(k)}$ et $\mathcal{N}_\beta^{(k)}$ peuvent :

393. **Fusionner** s'ils constatent que leur synergie latérale $\omega_{\alpha,\beta}^{(k)}$ croît jusqu'à dépasser un certain seuil, ou si leurs entités $\{\mathcal{E}_i\}$ montrent un recouvrement élevé.

394. **Coexister** paisiblement en maintenant une frontière clairement définie, peut-être par une relation modérée (ni trop basse, ni trop élevée).

395. **Entrer en compétition** pour des ressources partagées ou si leurs objectifs se recouvrent, ce qui peut conduire à une réduction de $\omega_{\alpha,\beta}^{(k)}$ via la dynamique de décroissance.

Un **exemple** concret est celui d'un niveau intermédiaire regroupant des équipes robotisées. Deux super-nœuds voisins peuvent être naturellement amenés à échanger des robots, des informations ou à se répartir des sous-zones d'exploration. Dès lors, la pondération $\omega_{\alpha,\beta}^{(k)}$ augmente si le partage se révèle mutuellement bénéfique, reflétant leur **coopération**. Dans le cas contraire, si des **conflits** surgissent (redondance de ressources, incompréhensions de mission), la valeur de $\omega_{\alpha,\beta}^{(k)}$ tend à diminuer, signifiant un cloisonnement plus net.

D. Avantages Généraux et Implications

Stabilisation latérale. Autoriser cette coordination horizontale évite que chaque super-nœud fonctionne en vase clos. Un cluster semi-global peut résoudre localement ses ajustements avec un voisin, limitant ainsi les remontées de demandes vers le niveau macro. Cette **stabilisation latérale** contribue à ce que la couche k soit plus robuste et cohérente avant de se projeter vers une agrégation de plus grande échelle.

Réduction de la surcharge au niveau supérieur. Dès lors que les clusters parviennent à trouver un consensus sur leur frontière commune, ils n'ont pas besoin de mobiliser le palier macro ($k + 1$). Le niveau supérieur est ainsi déchargé de multiples querelles de détail et peut se consacrer à la vision plus large et à la stratégie globale.

Complexité algorithmique. La gestion des liens latéraux $\omega_{\alpha,\beta}^{(k)}$ introduit une **matrice** de taille proportionnelle au nombre de super-nœuds au palier k . Bien que cela représente un coût en plus, ce coût reste très inférieur à l'échelle micro si le palier k agrège déjà plusieurs entités dans un même nœud. Il s'agit donc d'un **mini-réseau** plus léger à manipuler que le réseau initial complet.

Convergence globale. Sur le plan mathématique, la mise à jour simultanée des liaisons **verticales** (entre niveaux différents) et **horizontales** (à l'intérieur d'un même niveau) s'apparente à un **double mécanisme** d'auto-organisation. Le niveau k peut stabiliser la structure dans sa couche, tandis que la structure agrégée (ou filtrée) est progressivement transmise à $(k + 1)$. Une telle gestion hiérarchique, associée à la coordination latérale, favorise une convergence plus fluide et évite les incohérences ou oscillations qui surviendraient si tout devait être uniformément décidé au niveau macro.

Conclusion

La **coordination latérale** constitue un prolongement naturel de la logique multi-niveau dans un **DSL**. En s'appuyant sur des **pondérations horizontales** $\omega_{\alpha,\beta}^{(k)}$, elle permet à des super-nœuds **situés au même palier** d'interagir, de coopérer ou de se séparer selon les principes d'auto-organisation habituels : renforcement des synergies fortes, décroissance des liaisons peu utiles. Ce mécanisme évite de surcharger la couche macro (niveau $k + 1$) et instaure un degré supplémentaire de **liberté** : les regroupements semi-globaux peuvent évoluer ou fusionner en fonction de leurs besoins communs.

En pratique, cette interaction latérale se révèle précieuse pour gérer la **complémentarité** ou la **compétition** entre clusters de même échelle, tout en stabilisant la configuration avant de l'envoyer vers un palier supérieur. Elle contribue ainsi à la robustesse et à la scalabilité des **réseaux** construits sous l'angle d'un **Deep Synergy Learning** multi-niveau, que ce soit en robotique, en gestion distribuée ou dans des architectures d'intelligence artificielle neuronales.

6.4.3.3. Détection et gestion de conflits : si deux macro-nœuds aspirent la même entité

Dans la dynamique **multi-niveau** d'un **SCN** (Synergistic Connection Network), il peut arriver qu'au **niveau macro**, deux super-nœuds $\mathcal{N}_\alpha^{(k)}$ et $\mathcal{N}_\beta^{(k)}$ se disputent le **même** ensemble local, c'est-à-dire qu'une ou plusieurs entités $i \in \{\mathcal{E}_i\}$ (au niveau micro) se retrouvent "attirées" par les deux macro-clusters. Ce phénomène fait naître un **conflit** : lequel des macro-nœuds va réellement absorber (ou contrôler) l'entité i ? Si la **synergie** (ω) vers chacun est comparable, on risque des oscillations ou une incohérence. Cette section présente **comment** détecter ce type de conflit (section A) et **comment** le gérer mathématiquement (section B), afin d'assurer la stabilité globale du **DSL** (Deep Synergy Learning).

A. Détection de Conflits : deux macro-nœuds visant la même entité

Au **niveau 0** (micro), une entité \mathcal{E}_i peut avoir un ensemble de pondérations $\{\omega_{i,j}\}$ la reliant aux entités $\mathcal{C}_\alpha^{(0)}$ ou $\mathcal{C}_\beta^{(0)}$.

Après agrégation (chap. 6.2.2), on forme au niveau k deux **macro-nœuds** $\mathcal{N}_\alpha^{(k)}$ et $\mathcal{N}_\beta^{(k)}$. Chacun d'eux englobe un certain sous-ensemble d'entités (ou super-nœuds du palier $k - 1$).

Il se peut qu'une *même* entité \mathcal{E}_i (au palier micro) apparaisse partiellement dans la “zone” de $\mathcal{N}_\alpha^{(k)}$ et de $\mathcal{N}_\beta^{(k)}$. Par exemple, ses liens $\omega_{i,j}$ avec les membres de \mathcal{N}_α sont élevés, mais $\omega_{i,j}$ avec certains membres de \mathcal{N}_β le sont aussi.

Si la *même* entité \mathcal{E}_i est “aspirée” par $\mathcal{N}_\alpha^{(k)}$ et $\mathcal{N}_\beta^{(k)}$, la structure du SCN risque une incohérence : l'entité i ne doit pas se trouver **simultanément** dans deux macro-nœuds distincts (sauf si le modèle l'autorise, mais dans la plupart des schémas, on priviliege une partition ou un recouvrement contrôlé).

Sur le plan **mathématique**, on peut poser l'**indicateur** $\zeta(i, \alpha)$ qui vaut 1 si $i \in \mathcal{N}_\alpha$ et 0 sinon. Un conflit se détecte si $\zeta(i, \alpha) = \zeta(i, \beta) = 1$ pour $\alpha \neq \beta$.

Souvent, la **détection** se fait en observant la **somme des pondérations** entre i et chacun des macro-nœuds :

$$\Omega(i, \alpha) = \sum_{j \in \mathcal{N}_\alpha} \omega_{i,j}^{(0)}, \quad \Omega(i, \beta) = \sum_{j \in \mathcal{N}_\beta} \omega_{i,j}^{(0)}.$$

Si $\Omega(i, \alpha) \approx \Omega(i, \beta)$ et les deux dépassent un certain seuil, un conflit d’“aspiration” se produit.

B. Gestion Mathématique des Conflits : Choix, Partage, ou Inhibition

Une entité i ne peut appartenir *exclusivement* qu'à un **macro-nœud**. Lorsqu'on détecte un conflit ($\Omega(i, \alpha) \approx \Omega(i, \beta)$), on applique une **règle** de résolution :

$$\zeta(i, \alpha) = 1, \quad \zeta(i, \beta) = 0 \quad \text{ou} \quad \zeta(i, \alpha) = 0, \quad \zeta(i, \beta) = 1,$$

selon lequel des deux dépasse le plus nettement le seuil, ou selon un *tirage aléatoire pondéré* (si Ω est quasi identique).

On peut introduire un **terme** Δ_{conflict} qui “coupe” la liaison la plus faible, par ex. :

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) - \gamma(\cdot),$$

pour les liens orientés vers le macro-nœud qu'on *rejette*.

Dans certains modèles **non** exclusifs, on autorise un recouvrement des macro-nœuds (une entité peut être dans \mathcal{N}_α et \mathcal{N}_β) si c'est *logiquement* possible (ex. un robot fait partie de 2 équipes).

Mais cette *co-appartenance* peut introduire de la **redondance** et compliquer la logique d'agrégation. On implémente alors un **poids d'appartenance** $\zeta(i, \alpha) \in [0,1]$ qui mesure à quel degré i contribue à \mathcal{N}_α . S'il y a un conflit, on *répartit* ζ sur plusieurs macro-nœuds.

Troisième méthode : si le système **macro** (niveau k) considère que \mathcal{N}_α a déjà assez d'entités, il impose une **inhibition** ($\Delta_{\text{down}}^{(\alpha)} < 0$) sur les liaisons reliant i à \mathcal{N}_α , incitant i à se replier vers \mathcal{N}_β .

Le flux descendant du macro-niveau α vers l'entité i (au niveau micro) est :

$$\Delta_{\text{down}}^{(\alpha)}(i) = -\varepsilon \quad \text{si conflit.}$$

Cela **affaiblit** la synergie $\omega_{i,j}$ pour $j \in \mathcal{N}_\alpha$, poussant \mathcal{E}_i à basculer vers \mathcal{N}_β .

C. Stabilisation et Résolution Finale

Au fil des itérations, une entité i se **fixe** dans un macro-nœud \mathcal{N}_α ou un autre, au fur et à mesure que les **liens** (micro-liaisons $\omega_{i,j}$) s'ajustent en réponse aux signaux top-down.

Quand la **convergence** s'opère, le conflit cesse : i est pleinement (ou majoritairement) intégré à un cluster macro.

S'il n'y a pas de **règle** claire pour briser l'égalité $\Omega(i, \alpha) \approx \Omega(i, \beta)$, on peut voir i osciller entre \mathcal{N}_α et \mathcal{N}_β . D'où l'intérêt d'un *petit* mécanisme de **brisure de symétrie** (ex. un ajout aléatoire, un tirage stochastique) ou d'un *facteur d'hystérésis* (on ne bascule pas si la différence est $< \delta$) pour éviter le “flip-flop”.

Optionnellement, on peut définir une **énergie** locale $E_i(\alpha, \beta)$ associée au fait que i appartient à \mathcal{N}_α et \mathcal{N}_β . Si ce double appartenance coûte cher (haute énergie), le système finit par minimiser l'énergie en choisissant un *unique* cluster macro pour i .

D. Résumé et Impact

Quand deux macro-nœuds aspirent la même entité, cela révèle souvent une **ambiguïté** ; l'entité partage une forte synergie avec deux groupes différents. C'est fréquent si le **réseau** n'impose pas une partition stricte ou si la distribution initiale rend l'entité “hybride”.

La **gestion** de ce conflit est donc un aspect **naturel** de la coordination multi-niveau en DSL.

Une première approche consiste à appliquer un **choix excluant**, qu'il soit forcé ou stochastique, afin que l'entité finisse par appartenir à un seul macro-nœud. Une autre possibilité repose sur le **partage**, si le modèle autorise l'overlapping, ce qui implique l'introduction d'une pondération d'appartenance $\zeta(i, \alpha) \in [0,1]$. Enfin, une troisième stratégie repose sur l'**inhibition**, où l'entité est repoussée hors d'un macro-nœud par un mécanisme de feedback négatif.

En combinant un *terme top-down* et une *loi DSL* locale, on garantit qu'un conflit se **résoudra** généralement, tant que les paramètres (η, τ, γ) sont choisis pour **éviter** les oscillations permanentes. Des petits ajouts de *bruit* ou de *random break ties* aident aussi à la *brisure de symétrie*.

Conclusion

La **détection** et la **gestion** de conflits, lorsque *deux macro-nœuds* se disputent la *même* entité (ou sous-ensemble) au niveau micro, est un **phénomène courant** dans un **SCN** hiérarchisé :

- **Détection** : on repère que $\Omega(i, \alpha) \approx \Omega(i, \beta)$ pour l'entité i ,
- **Gestion** : on impose un **choix** (ex. inhiber l'un des liens), un **partage** (overlapping), ou un **feedback top-down** forçant le basculement,
- **Stabilité** : on évite les oscillations en introduisant une loi de “résolution” (briser l'égalité, paramétriser la feedback).

Sur le plan **mathématique**, ces mécanismes s'insèrent dans la mise à jour DSL combinant *renforcement local* et *rétroaction macro*. Ainsi, le système multi-niveau parvient à **assigner** chaque entité à un cluster macro (ou éventuellement à plusieurs, si le modèle l'autorise) de manière *cohérente*, évitant les incohérences structurelles qui mineraient la robustesse et la clarté de l'organisation globale.

6.5. Dynamique et Algorithmes Multi-Échelle

Dans de nombreux **SCN** (Synergistic Connection Networks), l'organisation **multi-niveau** n'est pas juste un concept statique, elle émerge et se peaufine *dynamiquement* grâce à l'application d'**algorithmes** spécifiques. La section 6.5.1 décrit l'**agrégation progressive** (bottom-up), principe selon lequel on forme des super-nœuds de plus en plus grands (micro → méso → macro). Ensuite, 6.5.2 verra la **division** (top-down), et 6.5.3 une **approche hybride**. Enfin, 6.5.4 discutera des aspects algorithmiques concrets et des paramètres clés.

6.5.1. Agrégation Progressive (Bottom-Up)

Lorsqu'on parle de **DSL** (Deep Synergy Learning) à plusieurs paliers, l'une des logiques classiques consiste à **agréger** progressivement les entités de bas en haut, d'abord on détecte des **micro-clusters**, puis on fusionne ces micro-clusters pour former des **super-nœuds**, et ainsi de suite jusqu'au "macro-nœud" ou niveau global.

6.5.1.1. Déetecter des Micro-Clusters, en Créer un "Super-Nœud" ; puis Déetecter des Super-Nœuds Cohérents pour un "Macro-Nœud", etc.

Introduction. Dans une architecture **Deep Synergy Learning** multi-niveau, l'un des mécanismes les plus naturels pour construire une **hiérarchie** consiste à procéder de manière bottom-up ; on commence par détecter, au **niveau micro**, des **micro-clusters** suffisamment cohérents. On agrège ensuite chacun de ces groupes en un super-nœud pour le palier suivant, puis on répète le même principe de détection/agrégation afin de former des *macro-clusters* ou un *macro-nœud* unique. Les paragraphes qui suivent détaillent la démarche et les fondements mathématiques associés.

A. Détection de Micro-Clusters au Niveau 0

On se place au **niveau 0**, celui des entités élémentaires $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Chaque paire (i, j) est associée à un poids $\omega_{i,j}^{(0)}$ et suit une **règle DSL** inspirée de la plasticité synaptique. Par exemple, une règle additive canonique peut s'écrire :

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_0 [S_0(i, j) - \tau_0 \omega_{i,j}^{(0)}(t)],$$

où η_0 est un **taux d'apprentissage**, τ_0 un **terme de décroissance**, et $S_0(i, j)$ une **fonction de synergie** mesurant la similarité ou l'utilité mutuelle des entités \mathcal{E}_i et \mathcal{E}_j . Après un certain nombre d'itérations, on voit émerger des **groupes** (micro-clusters) dont les pondérations internes $\omega_{i,j}^{(0)}$ restent élevées et se stabilisent, tandis que les liens extérieurs faiblissent ou s'annulent.

Lorsqu'un **sous-ensemble** $\mathcal{C}_\alpha^{(0)} \subset \{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ présente une **cohésion** suffisamment forte, on peut le reconnaître comme un **micro-cluster** stable. À partir de ce moment, on peut réduire l'information qu'il contient à un **super-nœud** $\mathcal{N}_\alpha^{(1)}$, de telle sorte que le **niveau 0** se voit partiellement "condensé" pour constituer le **niveau 1**.

B. Construction d'un Super-Nœud

Une fois que l'on a détecté un micro-cluster $\mathcal{C}_\alpha^{(0)}$, on crée un **super-nœud** $\mathcal{N}_\alpha^{(1)}$. Pour former l'ensemble des liaisons $\omega_{\alpha,\beta}^{(1)}$ au **niveau 1**, on réalise une **agrégation** des poids $\omega_{i,j}^{(0)}$. Une manière générique de procéder consiste à choisir une fonction Ψ et à poser :

$$\omega_{\alpha,\beta}^{(1)} = \Psi(\{\omega_{i,j}^{(0)} \mid i \in \mathcal{C}_\alpha^{(0)}, j \in \mathcal{C}_\beta^{(0)}\}).$$

La fonction Ψ peut être une **moyenne**, une **somme**, un **maximum**, ou toute autre mesure appropriée pour condenser les poids $\omega_{i,j}^{(0)}$ liant le cluster $\mathcal{C}_\alpha^{(0)}$ au cluster $\mathcal{C}_\beta^{(0)}$. Cette opération de filtrage ou de synthèse garantit que l'on passe d'un nombre potentiellement élevé de liaisons au niveau micro ($O(n^2)$) à un nombre bien plus faible de liaisons au niveau 1. De plus, cette **réduction** souligne uniquement les liens les plus significatifs entre les clusters du niveau inférieur.

C. Détection de Super-Nœuds Cohérents

Au **niveau 1**, on se retrouve avec un ensemble de super-nœuds $\{\mathcal{N}_1^{(1)}, \dots, \mathcal{N}_{m_1}^{(1)}\}$ et une matrice de pondérations $\omega_{\alpha,\beta}^{(1)}$. On peut appliquer la **même** règle DSL (ou une variante) pour faire évoluer ces poids :

$$\omega_{\alpha,\beta}^{(1)}(t+1) = \omega_{\alpha,\beta}^{(1)}(t) + \eta_1 [S_1(\alpha, \beta) - \tau_1 \omega_{\alpha,\beta}^{(1)}(t)].$$

Ici, η_1 et τ_1 sont des paramètres d'apprentissage fixés pour le niveau 1, et $S_1(\alpha, \beta)$ est la nouvelle fonction de synergie, reflétant la proximité ou la compatibilité entre super-nœuds $\mathcal{N}_\alpha^{(1)}$ et $\mathcal{N}_\beta^{(1)}$. Cette dynamique auto-organisée permet de repérer à ce **deuxième palier** des regroupements $\{\mathcal{C}_p^{(1)}\}$ que l'on convertit ensuite en super-nœuds $\mathcal{N}_p^{(2)}$ pour le **niveau 2**, et ainsi de suite. En itérant le processus, on peut progressivement atteindre un **macro-niveau** (niveau K) où il ne reste plus qu'un petit nombre de super-nœuds de grande taille, voire un seul **macro-nœud** global.

D. Progression jusqu'au Macro-Nœud

La répétition de l'agrégation forme une **cascade** :

396. Le niveau 0 détecte des micro-clusters $\mathcal{C}_\alpha^{(0)}$.

397. Les micro-clusters sont condensés en super-nœuds $\mathcal{N}_\alpha^{(1)}$.

398. Le niveau 1 détecte à son tour des clusters $\mathcal{C}_p^{(1)}$ parmi les super-nœuds déjà existants.

399. Ces regroupements sont convertis en super-nœuds $\mathcal{N}_p^{(2)}$.

400. Le niveau 2 poursuit le même mouvement, etc.

Ce **principe bottom-up** s'arrête lorsque le nombre de super-nœuds restants est faible (par exemple, un unique macro-nœud englobant tout), ou lorsque certains **critères** d'arrêt (seuil de similarité, contrainte de taille, etc.) sont atteints. La complexité $O(n^2)$ potentiellement nécessaire pour mettre à jour toutes les liaisons au niveau micro se résorbe au profit d'un nombre plus restreint de liaisons $O(m^2)$ au fur et à mesure que $m \ll n$.

Conclusion

Le processus de **détection de micro-clusters** et la création de **super-nœuds** (puis de **macro-nœuds**) constitue l'une des approches les plus directes pour construire une hiérarchie dans un **Deep Synergy Learning**. En partant des pondérations au **niveau micro**, on détecte les sous-ensembles d'entités fortement synergiques, on les agrège en super-nœuds et l'on répète la même logique de **mise à jour** et d'**auto-organisation** sur les nouveaux niveaux. Cette **progression bottom-up** confère une structure multi-niveau organique et réduit progressivement la complexité, tout en maintenant la **cohérence** du réseau. Les micro-clusters apparaissent comme le socle de base, suivi par des regroupements de plus en plus larges, jusqu'à un **macro-nœud** unique, si tel est l'objectif final. C'est précisément cette capacité à gérer les données ou les agents "depuis la racine" jusqu'à l'échelle la plus large qui fait la force du **DSL** en environnement hétérogène ou massivement distribué.

6.5.1.2. Avantages : Construction Organique, Moins de Paramétrage Initial

Introduction. La stratégie **bottom-up** d'agrégation progressive, décrite précédemment pour bâtir une hiérarchie de super-nœuds (section 6.5.1.1), comporte des avantages significatifs lorsque l'on souhaite structurer un **Synergistic Connection Network (SCN)** en plusieurs échelles de manière flexible. Contrairement aux approches imposant dès le départ un découpage (top-down) ou un nombre fixe de partitions (clustering global), l'agrégation **bottom-up** offre une **construction organique** qui épouse la dynamique locale du **DSL** et réclame un **paramétrage** minimal. Les développements ci-après mettent en évidence les principaux bénéfices, tout en situant cette démarche par rapport à d'autres méthodes.

A. Construction Organique

La première caractéristique marquante de l'agrégation **bottom-up** tient au fait qu'elle est en phase directe avec la **dynamique** locale du Deep Synergy Learning. Au **niveau micro**, on met en œuvre la règle DSL :

$$\omega_{i,j}^{(0)}(t+1) = \omega_{i,j}^{(0)}(t) + \eta_0 [S_0(i,j) - \tau_0 \omega_{i,j}^{(0)}(t)].$$

Cette équation favorise l'émergence de **micro-clusters** $\{\mathcal{C}_\alpha^{(0)}\}$ qui se forment dès lors que certaines paires ou triplets d'entités présentent une synergie récurrente. Contrairement à une coupe arbitraire, ces groupes se **stabilisent** naturellement sous l'effet des renforcements et des décroissances sélectives de $\omega_{i,j}$. L'agrégation progressive prend alors ces micro-clusters comme points de départ et les élève au palier supérieur sous forme de super-nœuds, sans brusquer la progression de l'auto-organisation locale.

Une fois les micro-clusters détectés, on ne "sauter" pas immédiatement à un macro-niveau unique, on laisse la possibilité de créer des **niveaux intermédiaires** (méso-niveaux), répétant la mise à jour DSL. Les fusions successives reproduisent une logique de **croissance progressive**, proche de mécanismes observés dans des contextes biologiques ou sociologiques, où de petits groupements se consolident en sous-communautés avant d'aboutir, le cas échéant, à des structures englobantes plus vastes.

Cette architecture bottom-up se révèle extrêmement **adaptative**. Si de nouveaux liens $\omega_{i,j}$ apparaissent (ou se renforcent) au niveau micro, cela provoque la création (ou la reconfiguration) de clusters. Ces changements locaux se répercutent naturellement aux paliers supérieurs, lesquels peuvent également réajuster leurs super-nœuds ou fusionner des groupes si la synergie l'exige. La **souplesse** qui en découle est particulièrement cruciale dans des environnements évolutifs ou soumis à des flux de données continus.

B. Moins de Paramétrage Initial

Contrairement aux approches top-down, où il faut généralement définir un **nombre de partitions** dès le début (ex. dire "on veut 5 clusters" ou "on découpe en 3 niveaux de taille égale"), l'agrégation **bottom-up** se limite à un **faible** nombre de paramètres.

Les seuls éléments imposés concernent la **règle de mise à jour** (taux η_0 , décroissance τ_0 , etc.) et la **fonction de synergie** $S_0(i,j)$. Éventuellement, on peut définir un **seuil** pour reconnaître qu'un groupe local est assez cohésif, mais sans imposer la forme ou la taille exacte de chaque cluster.

Il n'est pas nécessaire de fixer a priori le nombre exact de clusters, ni même de préciser la profondeur de la hiérarchie. Les niveaux intermédiaires surgissent **spontanément** dès lors que des sous-ensembles \mathcal{C}_α se distinguent et sont jugés suffisamment stables pour être agrégés. La hiérarchie finale – pouvant s'arrêter à 1, 2, 3 ou plus de niveaux – est ainsi **découverte** plutôt qu'imposée.

Lorsque le partitionnement est dicté en amont, on court le danger de forcer des **découpages artificiels**, mal adaptés à la structure réelle des synergies $\omega_{i,j}$. L'approche **bottom-up** évite ces erreurs en laissant la dynamique et les degrés de cohésion déterminer les regroupements. Les grands clivages dans le réseau émergent **naturellement** s'ils sont effectivement présents, tandis que les liens de force intermédiaire aboutissent à des fusions partielles ou restent séparés.

C. Implications Pratiques

La mise en place d'une agrégation **bottom-up** se traduit par une séquence d'étapes :

401. Boucles d'Itération au Niveau Micro.

Le réseau exécute la règle DSL pendant un certain nombre d'itérations, renforce les liens synergiques et affaiblit les autres.

402. Détection/Actualisation des Micro-Clusters.

On identifie les groupes ayant atteint une cohésion suffisante.

403. Construction du Palier Supérieur.

Chaque micro-cluster devient un super-nœud, et l'on agrège les poids $\omega^{(0)}$ pour former $\omega^{(1)}$.

404. Mise à Jour au Niveau Méso.

Le même procédé d'**auto-organisation** se poursuit, cette fois entre super-nœuds, jusqu'à former de nouveaux regroupements $\mathcal{C}_\alpha^{(1)}$.

405. Prolongement ou Arrêt.

On répète les paliers tant qu'il reste un intérêt à poursuivre l'agrégation. S'il ne subsiste qu'un petit nombre de super-nœuds ou un unique **macro-nœud**, on stoppe la progression.

Cette stratégie présente une **scalabilité** naturelle : chaque palier réduit le nombre de nœuds en agrégeant ceux qui sont trop fortement liés, ce qui évite d'opérer en permanence à l'échelle du réseau entier. Par ailleurs, elle autorise une **synchronisation ascendante** : on attend que le niveau micro se stabilise avant de promouvoir la structure au palier suivant, maintenant ainsi une cohérence hiérarchique plus solide.

Conclusion

L'agrégation **bottom-up** offre à la fois une **construction organique** totalement alignée sur la dynamique locale du DSL et un **paramétrage réduit** à l'essentiel (paramètres de la règle DSL, éventuellement un critère de détection de clusters). Les clusters émergent librement, fusionnent ou s'ignorent selon les synergies perçues dans le réseau, et la hiérarchie finale se forme pas à pas, en respectant la **réalité** des liens $\omega_{i,j}$. Les avantages de ce procédé – flexibilité, adaptabilité, minimisation des découpages arbitraires – en font un choix particulièrement attrayant pour gérer des structures **massivement distribuées** ou pour modéliser des processus d'**auto-organisation** complexes où l'on ne souhaite pas imposer de partitions rigides a priori.

6.5.1.3. Inconvénients : Besoin d'Algorithmes “Greedy” ou Heuristiques, Potentiels Merges Successifs

La démarche **bottom-up** (agrégation progressive) exposée dans les sections précédentes (6.5.1.1 et 6.5.1.2) apporte une flexibilité et une réduction du paramétrage initial, mais n'est pas exempte de limites. Elle repose souvent sur des **algorithmes** de fusion successifs, de type *greedy* ou **heuristiques**, et peut ainsi s'avérer délicate dans certaines circonstances. Les paragraphes qui suivent soulignent en particulier deux grandes familles d'inconvénients : l'impossibilité de garantir la fusion optimale (nécessitant des heuristiques), et le risque de **merges** itératifs pouvant conduire à des configurations sous-optimales par “erreur cumulative”.

A. Besoin d'Algorithmes Greedy ou Heuristiques

Dans un **SCN** (Synergistic Connection Network) de grande taille, trouver la **meilleure** manière de fusionner les micro-clusters est un problème qui, sous de nombreuses variantes (community detection, clustering hiérarchique optimal, etc.), est **NP-difficile**. Autrement dit, l'exploration exhaustive de toutes les possibilités de partitions ou de merges devient **inabordable** à mesure que le nombre d'entités croît. Vouloir choisir à chaque étape la fusion localement “idéale” – ou, plus encore, vouloir optimiser globalement l'agencement des fusions – se heurte à une **explosion** combinatoire. Cela rend quasi impossible, en pratique, l'obtention d'une **solution exacte** dès lors que n dépasse quelques dizaines ou centaines.

Pour contourner cette complexité, on recourt typiquement à des **procédures** itératives, dites *greedy merges*, où l'on :

- Cherche les deux clusters \mathcal{C}_α et \mathcal{C}_β qui présentent la **plus forte synergie** (ou dont la fusion entraîne la **meilleure réduction de coût** local),
- Fusionne ces deux ensembles en un **super-cluster** $\mathcal{C}_{\alpha \cup \beta}$,

- Met à jour la structure, puis répète l'opération jusqu'à ce qu'on atteigne un niveau jugé suffisant (arrêt par critère de taille, de niveau, ou de cohésion).

De nombreux schémas d'**agrégation hiérarchique** (single linkage, complete linkage, average linkage, etc.) s'appuient sur cette philosophie, en substituant un **score** de proximité ou de similarité entre clusters. Dans un **SCN**, on peut utiliser les **pondérations** $\omega_{\alpha,\beta}$ pour déterminer la force de rapprochement : plus $\omega_{\alpha,\beta}$ est élevée, plus il est tentant de fusionner les clusters α et β . Il s'agit cependant d'un **choix local**, qui ne garantit pas de maximiser la cohésion globale du réseau (ou de minimiser toute fonction de coût plus étendue).

B. Risques de Merges Successifs et Erreurs Cumulatives

Parce qu'on s'appuie sur une suite de **décisions locales** (fusionner deux clusters \mathcal{C}_α et \mathcal{C}_β si leur lien est jugé suffisant), il y a un risque d'**erreurs** successives qui finissent par s'accumuler dans la configuration finale.

La règle *greedy* peut pousser à fusionner rapidement deux clusters dont la synergie est pour l'instant dominante, mais qui pourraient en réalité être mieux agencés (ou scindés) si l'évolution du réseau se prolongeait. On peut voir cela comme un *verrouillage* : une fois la fusion effectuée, le nouveau super-cluster $\mathcal{C}_{\alpha \cup \beta}$ est rarement redécomposé au sein du même algorithme bottom-up. Des "fusions hâties" de ce type peuvent enfermer la hiérarchie dans un état sous-optimal.

Dans la plupart des implémentations, l'**agrégation progressive** ne prévoit pas de "**dé-fusion**" ou de *split* durant le même cycle d'ascension. Il faudrait recourir à des *extensions* plus complexes (chap. 6.5.2 sur des divisions possibles, ou des mécanismes top-down correctifs) pour rétablir une répartition plus adaptée.

Sans cette possibilité de correction, chaque petit choix local vient s'inscrire dans la structure globale, risquant de créer une **erreur cumulative** au fil des paliers.

Si l'on associe à la partition du réseau une **énergie** (ou "coût") \mathcal{E} à minimiser, tout algorithme de fusion local vise à réduire $\Delta\mathcal{E}$ sur l'instant. Or, un optimum local n'est pas nécessairement un optimum global, et, en multipliant les "petites optimisations" locales, on peut aboutir à un état final nettement plus élevé en \mathcal{E} que la **configuration** la plus favorable.

C. Conséquences Pratiques et Possibles Solutions

Ces inconvénients ne rendent pas l'agrégation *bottom-up* inopérante, mais soulignent la nécessité de précautions ou de *mécanismes complémentaires*.

Il est fréquent d'introduire un **feedback descendant** (cf. 6.4) au cours duquel un niveau supérieur peut suggérer la re-scission d'un cluster si la macro-analyse repère des incohérences. Cela donne un algorithme plus complet, où la construction *bottom-up* est **assouplie** par des corrections top-down.

Dans des architectures DSL complexes, ce double flux (ascendant et descendant) permet de retarder ou d'inverser certaines unions malheureuses.

S'il n'existe pas de mécanisme de "re-diviser", on peut subir d'importants effets d'inertie. Une fusion inappropriée à la première étape se répercute dans les paliers ultérieurs, menant parfois à une structure loin de l'optimum. À l'inverse, si on autorise trop facilement la division, il y a un risque d'oscillations (fusions défusions récurrentes).

Des **seuils** ou paramètres de stabilisation (hystérésis, temporalité de mise à jour) aident à limiter ces phénomènes.

En dépit de ces limites, un algorithme *greedy* de fusion hiérarchique reste souvent **très rapide** et **simple** à mettre en œuvre. Il donne des résultats satisfaisants dans la majorité des scénarios pratiques, pourvu qu'on accepte une configuration "raisonnablement bonne" plutôt que *parfaitement optimale*.

Conclusion

Le **bottom-up** en agrégation progressive (cf. 6.5.1.1–6.5.1.2), bien qu'utile pour **construire** et **réduire** un **SCN** sur plusieurs paliers, souffre de **deux** inconvénients majeurs :

406. Recours quasi obligatoire aux heuristiques ou règles *greedy*.

Les fusions successives impliquent un choix local, puisque la solution globale est **NP-difficile**. Les algorithmes de type single linkage, average linkage, ou fusion par “plus forte synergie” n’assurent donc pas un optimum global.

407. Risque d’erreurs cumulatives.

Les “merges” successifs peuvent verrouiller la hiérarchie dans une configuration sous-optimale si aucune procédure de vérification ou de *dé-fusion* n’est envisagée. Cela appelle, en pratique, des **approches hybrides** mixant *bottom-up* et corrections top-down pour se prémunir des unions précipitées.

Ainsi, la **conclusion** s’impose : l’agrégation **bottom-up**, tout en **facilitant** la construction organique et **réduisant** le paramétrage, doit s’accompagner de **gardes-fous** (soit via un niveau supérieur actif, soit via des mécanismes de division) pour que les “merges” successifs ne figent pas le réseau en un état peu satisfaisant. Dans un cadre **DSL** plus large, on considère souvent une **multi-directionnalité** (ascendant et descendant) afin de compenser ces faiblesses tout en conservant les bénéfices de la fusion progressive.

6.5.2. Division ou Zoom (Top-Down)

En **complément** de la logique d’agrégation (*bottom-up*) décrite en (6.5.1), certains **algorithmes** multi-niveau adoptent ou complètent l’approche **top-down**, où l’on **part** d’un **macro-cluster** global (ou d’un ensemble restreint de grands clusters) pour ensuite **segmentation** s’il apparaît que la **cohésion** interne est insuffisante. La section 6.5.2.1 introduit la notion d’une **approche fractale descendante**, qui renverse la perspective : plutôt que de fusionner progressivement, on “zooome” (divise) progressivement.

6.5.2.1. Approche Fractale Descendante : Partir d’un Cluster Global et le Segmenter si des Sous-Groupes se Forment

Contrairement à la logique **bottom-up**, qui agrège progressivement des micro-clusters vers des super-nœuds (section 6.5.1), la démarche **top-down** se propose de partir d’un **grand** ensemble englobant toutes les entités et d’y opérer des **divisions** successives dès que l’on détecte des “failles” ou des “sous-groupes” cohésifs. Cette approche se qualifie parfois de “fractale descendante” parce qu’elle reproduit à chaque échelle la même procédure de scission, révélant des **structures** de plus en plus fines. Les paragraphes suivants présentent le principe général, illustrent le lien avec la notion de fractalité, puis discutent les avantages et inconvénients de cette segmentation top-down.

A. Principe d’une Division/Zoom du Macro vers le Micro

La stratégie **top-down** consiste à partir d’un **unique** cluster (macro) qui inclut la totalité des entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. On considère alors ce bloc global comme un “super-nœud” $\mathcal{N}^{(\text{global})}$. À ce stade, si l’on estime (via une fonction de cohésion, d’hétérogénéité ou de modularité) que ce vaste ensemble est **trop** hétérogène, on opère une **division** : on sépare $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ en plusieurs **sous-clusters** plus homogènes.

On ne part pas d’entités dispersées, mais d’un bloc unique englobant tout. Cela peut correspondre à la situation où l’on sait d’emblée qu’il existe un *objectif commun* ou qu’un macro-niveau coiffe l’ensemble des agents (par exemple, un centre de commande).

Dès lors qu’un critère d’hétérogénéité (ou de “faible synergie interne”) est détecté, on *scinde* le cluster macro en deux (ou plus) sous-blocs. Chaque sous-bloc \mathcal{C}_α , considéré comme un cluster local, peut faire l’objet du **même** examen, se divisant à son tour si nécessaire.

À chaque niveau de division, la **même règle** s’applique : si un sous-ensemble est encore hétérogène, on le scinde de nouveau. Ce procédé descend petit à petit vers des regroupements de taille moyenne (méso-niveau), puis éventuellement de petite taille (micro-niveau), tant que la cohésion interne n’est pas jugée satisfaisante.

B. Allusion à une Logique Fractale

Le terme “fractale” se réfère au fait que la même *loi* de division se reproduit à chaque échelle, dévoilant une **auto-similarité** structurale. On peut définir une **fonction** d’hétérogénéité \mathcal{H} mesurant dans quelle mesure un ensemble est “mélangé”. Si \mathcal{H} s’avère trop élevée pour le cluster global, on le coupe en deux blocs, puis on évalue \mathcal{H} dans chacun de ces blocs, et ainsi de suite :

$$\text{si } \mathcal{H}(\mathcal{C}) > \theta > \Rightarrow > \text{division de } \mathcal{C} > \text{en sous-blocs } \mathcal{C}_\alpha, \dots, \mathcal{C}_\beta.$$

Cette itération (ou récursion) rappelle l’idée qu’en “zoomant” à un niveau plus fin, on applique la **même** loi de division, ce qui renvoie au concept d'**auto-similarité fractale** : le “motif” de segmentation se répète à chaque échelle jusqu’à atteindre un seuil de granularité.

C. Exemple d’Illustration : Cortex, Réseaux, etc.

Cortex. Si on envisage un ensemble de neurones occupant une zone cérébrale globale, on peut constater que certains neurones forment un sous-ensemble très interconnecté, ce qui justifie une **scission** dans la structure globale. On isole ce sous-groupe en tant qu’aire plus restreinte, puis on continue le même examen en son sein.

Réseaux (Graphes). Sur un graphe englobant toutes les entités, la détection de **communautés** peut s’effectuer en repérant des coupes ou des **faiblesses** dans la matrice de pondérations ω . Chaque fois qu’on découvre un sous-ensemble de nœuds mieux reliés entre eux qu’au reste du graphe, on scinde. Les sous-ensembles sont ensuite traités comme des graphes à part entière, dans lesquels on peut dénicher des sous-communautés supplémentaires.

Analogie fractale. Dans certains systèmes, cette hiérarchie descendante révèle des *patterns* identiques à différentes échelles, renforçant l’aspect fractal. Les mêmes principes de “cohésion interne” et de “faibles connexions externes” se répètent du niveau macro jusqu’au niveau micro.

D. Schéma Mathématique d’une Division Descendante

On note $\mathcal{N}_{\text{global}}^{(K)}$ le cluster unique contenant tous les indices. On évalue sa “cohérence” ou son hétérogénéité $\mathcal{H}(\mathcal{N}_{\text{global}}^{(K)})$. Si cette valeur reste en deçà d’un seuil θ_K , on décide de **conserver** le bloc entier. Sinon, on le segmente en plusieurs sous-ensembles.

À chaque découpage, un cluster $\mathcal{N}_\alpha^{(k)}$ est subdivisé en deux (ou plusieurs) sous-clusters $\mathcal{N}_{\alpha_1}^{(k-1)}, \mathcal{N}_{\alpha_2}^{(k-1)}, \dots$. On mesure ensuite $\mathcal{H}(\mathcal{N}_{\alpha_i}^{(k-1)})$ pour voir si on doit à nouveau “descendre” d’un cran. Cette itération se poursuit tant que l’on découvre des divisions nécessaires ou que l’on n’a pas atteint la taille souhaitée (niveau micro).

Le processus s’interrompt lorsque tous les *blocs* résultants satisfont un critère de cohésion, ou bien lorsqu’on considère qu’ils sont déjà suffisamment fins (niveau minimal). On obtient alors une hiérarchie descendante où chaque palier résulte de scissions itérées.

E. Avantages d’une Approche Top-Down

Le fait de commencer par un bloc global procure une **vision** macroscopique d’ensemble. Au lieu d’assembler lentement des micro-éléments, on *contrôle* tout de suite la partition du grand ensemble, ce qui peut être avantageux si le système est déjà régi par un cadre commun ou un gestionnaire central.

Dans une démarche descendante, on peut décider à chaque étape comment scinder le bloc (en deux, en trois, etc.) en fonction des **informations** dont on dispose (score de variance, synergie interne, etc.). On évite le risque d’avoir un trop grand nombre de micro-clusters non désirés au départ.

Si la structure du réseau présente une forme d'**auto-similarité**, la segmentation top-down la fait apparaître naturellement : on “zoome” dans le grand ensemble et retrouve la même loi de division. Cela peut être particulièrement intéressant dans des contextes où l’échelle macro est connue pour être hétérogène, mais où l’on s’attend à des *patterns* récurrents à des niveaux plus fins.

F. Limites et Paramétrages

Malgré son intérêt, cette approche **top-down** soulève plusieurs questions :

408. Choix du Critère de Division.

Il faut définir comment on décide qu'un bloc est "trop hétérogène". Cette définition implique la mise en place d'une **fonction \mathcal{H}** (ou d'un critère d'énergie \mathcal{E}) et d'un **seuil θ** . Ces paramètres doivent être ajustés de manière à éviter soit l'absence de division (si θ est trop élevé), soit une segmentation excessive (si θ est trop bas).

409. Décisions de Découpage.

Une fois qu'on a décreté qu'un cluster se scinde, comment répartir les entités ? Faut-il un simple **bipartitionnement** ou bien une **multipartition** ? Les algorithmes de segmentation peuvent être aussi complexes que ceux employés en "cut" de graphes, nécessitant parfois des heuristiques (ex. minimisation d'une coupe, maximisation de la modularité locale).

410. Risque de Sur-Segmentations.

Si l'on n'y prend pas garde, on peut créer trop de sous-blocs. Certains systèmes peuvent présenter des liaisons d'intensité moyenne et non pas de fortes discontinuités, ce qui conduit à scinder trop finement et à rater la cohérence multi-niveau. Comme en bottom-up, des algorithmes heuristiques peuvent conduire à des configurations globalement sous-optimales.

Conclusion

L'**approche fractale descendante** (ou **top-down**) consiste à partir d'un **super-nœud** global, représentant l'ensemble du réseau ou des entités, et à **segmenter** progressivement ce bloc si l'hétérogénéité interne l'exige. À chaque division, la **même** règle est appliquée pour détecter des sous-groupes, jusqu'à ce que tous les clusters résiduels soient suffisamment cohésifs.

411. Principe. Le *zoom* du macro vers le micro reproduit la même dynamique à chaque échelle, dans un esprit fractal où la segmentation se répète si l'ensemble considéré reste trop "mélangé".

412. Forces. On bénéficie d'une **vision globale** dès le départ, et on garde un contrôle direct sur la découpe en nombre de sous-blocs.

413. Faiblesses. Le critère de division et les paramètres associés (seuil d'hétérogénéité, type de bipartition ou multipartition, etc.) peuvent s'avérer délicats à régler. On risque la **sur-segmentation** ou, au contraire, l'absence de division fine quand on emploie des heuristiques inadaptées.

Au final, cette méthode top-down apporte une solution complémentaire à la logique bottom-up (section 6.5.1). Dans les architectures **DSL** multi-niveau, il est courant de **mélanger** ces deux perspectives (voir section 6.5.3), permettant ainsi à la structure de se réorganiser à la fois par fusion **ascendante** et par subdivision **descendante**, assurant une hiérarchie ajustée à toutes les échelles.

6.5.2.2. Systèmes "Macro → Micro" : Si la Synergie Interne n'est pas Assez Homogène, On Scinde

Dans la dynamique **top-down** (ou descendante) de la construction multi-niveau, il est possible de partir d'un **ensemble englobant** (un ou plusieurs gros clusters) et de **scinder** ceux-ci dès que leur cohésion interne est jugée insuffisante. Cette logique se fonde sur l'idée de "zoomer" progressivement : on ne crée pas de micro-clusters dès le départ, mais on initialise un **bloc global** et on le subdivise au fur et à mesure que la synergie interne se révèle inhomogène. Les sections suivantes détaillent le principe, illustrent la mécanique de division, et exposent les avantages comme les contraintes pratiques de ce mode "macro → micro".

A. Principe : Partir d'un (ou de quelques) Grand(s) Bloc(s)

Une caractéristique fondamentale de la démarche top-down est qu'elle **commence** par un **très gros cluster** — parfois unique — couvrant la totalité ou une grande portion de l'ensemble $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. D'un point de vue pratique, on peut considérer qu'une entité $\mathcal{N}^{\text{global}}$ réunit tous les indices dans un **super-nœud**. On analyse alors la **cohérence** de ce super-nœud :

414. Définition d'un cluster macro initial.

On note $\mathcal{C}^{(\text{macro})}$ ce grand bloc. Dans certains contextes, cette configuration est naturelle : un système robotique opérant sous un même contrôle, une base de données qu'on n'a pas encore partitionnée, etc.

415. Analyse de la synergie interne.

On introduit un **critère** $\mathcal{H}(\mathcal{C}^{(\text{macro})})$ évaluant l'homogénéité ou la cohésion interne. Il peut s'agir d'un indicateur calculé à partir des poids $\omega_{i,j}$ (moyenne, variance, distribution des liaisons, etc.), ou d'une mesure plus large (taux d'accord sémantique, par exemple).

$$\mathcal{H}(\mathcal{C}) = f(\{\omega_{i,j}\}_{i,j \in \mathcal{C}}).$$

Si \mathcal{H} est trop élevé (sous l'hypothèse que "trop élevé" signifie "trop hétérogène"), on entreprend une scission.

B. Si la Synergie Interne n'est pas Assez Homogène, On Scinde

Lorsqu'un cluster macro $\mathcal{C}^{(\text{macro})}$ s'avère **insuffisamment** cohérent, on le **divide**. Cela se traduit souvent par :

416. Décision de scission.

On identifie, au sein de $\mathcal{C}^{(\text{macro})}$, des sous-groupes plus denses ou des "failles" dans la matrice ω . On partitionne $\mathcal{C}^{(\text{macro})}$ en deux (ou plus) blocs, $\mathcal{C}_\alpha^{(\text{macro}-1)}, \mathcal{C}_\beta^{(\text{macro}-1)}, \dots$, qui héritent chacun d'une **cohésion** mieux définie.

Par exemple, on peut imposer un gain minimum $\delta > 0$ en termes de réduction de l'hétérogénéité :

$$\mathcal{H}(\mathcal{C}_\alpha)\mathcal{H}(\mathcal{C}_\beta) \leq \mathcal{H}(\mathcal{C}^{(\text{macro})}) - \delta.$$

417. Descente itérative.

Après cette première scission, on applique la **même** opération à chaque sous-cluster obtenu. Si l'on constate qu'un bloc reste trop hétérogène, on le scinde de nouveau, et ainsi de suite. D'un point de vue fractal, la **même** règle de division se reproduit à chaque niveau, tant que la **cohérence** n'est pas jugée satisfaisante.

418. Exemple de formulation.

On peut définir $\mathcal{H}(\mathcal{C})$ comme la **variance** des poids $\{\omega_{i,j}\}$ à l'intérieur de \mathcal{C} , ou comme un **écart** normalisé :

$$\mathcal{H}(\mathcal{C}) = \frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} |\omega_{i,j} - \bar{\omega}_\mathcal{C}|.$$

Si $\mathcal{H}(\mathcal{C}) > \theta$, on scinde. Les critères peuvent varier selon la nature des données et la signification de la synergie.

C. Avantages de ce Zoom Descendant

La logique "macro → micro" recèle plusieurs **points forts** qui la distinguent de l'agrégation ascendante :

On commence par un ensemble unique, ce qui procure une **vue d'ensemble** sur l'intégralité du réseau. Cela évite de créer d'embrée de multiples clusters "éparpillés" qu'il faudrait ensuite fusionner.

Si le SCN possède une forme d'auto-similarité (section 6.3), cette descente révèle à chaque niveau des **motifs** comparables aux niveaux supérieurs. Chaque scission reproduit la même règle, “zoomant” dans le bloc pour en extraire des sous-blocs plus cohérents. Il est parfois plus naturel, pour des raisons d'ingénierie ou de supervision, de “partir grand” et de n'effectuer des partitions qu'en cas de nécessité. Cela correspond à un schéma où le macro-niveau règne et ne crée de subdivisions qu'en identifiant des segments mal intégrés.

D. Inconvénients à Garder à l'Esprit

Comme toute méthode descendante, ce schéma n'est pas exempt de **limitations** :

Risque de sur-segmentation. Si le critère d'homogénéité est trop exigeant, on risque de fragmenter exagérément le cluster macro, produisant nombre de petits blocs. Cela reflète une **division excessive** qui peut saper la compréhension globale ou perdre de la connectivité.

Paramétrage délicat. Il faut fixer un **seuil** θ ou un **niveau** de variance acceptable. Un θ trop bas conduit à ne jamais scinder, un θ trop haut peut déclencher des coupes incessantes. On se retrouve face à un problème similaire à la détermination du nombre k de partitions dans les algorithmes classiques de clustering.

Algorithmes non optimaux. Découper $\mathcal{C}^{(\text{macro})}$ de manière “parfaite” (minimisant une fonction de coût globale) est souvent **NP-difficile** si \mathcal{C} est volumineux. On recourt donc à des **heuristiques** (ex. on identifie un couplage faible entre deux groupes et on coupe le long de cette “faille”). Mais rien ne garantit l'optimalité de la répartition obtenue.

Moins de “naissance” organique. Contrairement à la démarche bottom-up, où les micro-clusters émergent localement sans action extérieure, ici la partition est décidée par un **niveau macro**, imposant la coupe depuis le haut. Cela peut être moins “naturel” dans un système où la dynamique DSL locale est la clé principale d'auto-organisation.

Conclusion

Dans un **système “macro → micro”**, on initialise la structure par un **grand** cluster englobant toutes les entités, et l'on **scinde** en sous-ensembles dès qu'un critère d'homogénéité (ou de cohésion interne) n'est plus satisfait. Cette logique **top-down** (ou fractale descendante) consiste à répéter la **même** règle de division à chaque niveau, en repérant des failles internes au bloc actuel. Elle apporte plusieurs **avantages**, notamment une **vue globale** immédiate et la possibilité d'ajuster *a priori* la segmentation. En revanche, elle requiert un **paramétrage** (définition d'un seuil, choix d'un algorithme de coupe) et peut **sur-fragmenter** le réseau si la synergie moyenne est faible ou si l'on exige une homogénéité trop stricte.

Comme souvent dans les architectures DSL multi-niveau, on obtient de bons résultats en **combinant** cette approche top-down avec la démarche bottom-up (chap. 6.5.3). L'une agit comme un filet de sécurité pour éviter une fusion exagérée, l'autre compense le risque de divisions superflues, aboutissant in fine à une hiérarchie plus stable et plus fidèle à la véritable configuration du **Synergistic Connection Network**.

6.5.2.3. Risques d'une Division Excessive ou de la Sur-Segmentацию

Introduction.

Dans la démarche **top-down** (voir 6.5.2.1 et 6.5.2.2), on part d'un cluster global englobant toutes les entités, puis on le scinde dès que l'on juge sa cohésion interne insuffisante. Bien que cela permette de “zoomer” sur des sous-groupes plus homogènes, on court également le risque d'une **sur-segmentation** lorsque les critères de division sont trop stricts ou appliqués trop précocement. Il s'ensuit une multiplication de petits blocs, ce qui peut compromettre la logique de construction hiérarchique à moyenne ou grande échelle. Cette section examine les principaux **mécanismes** menant à la sur-segmentation, puis discute des **conséquences** et des **pistes** pour l'atténuer.

A. Mécanismes menant à la sur-segmentation

Dans un schéma top-down, on définit un **critère** $\mathcal{H}(\mathcal{C})$ évaluant l'hétérogénéité interne d'un cluster \mathcal{C} , associé à un **seuil** θ . Si θ est placé trop bas (ou la cohésion exigée est trop élevée), alors même des blocs relativement cohérents

apparaissent comme “insuffisamment homogènes”. Cela déclenche des **divisions** successives qui morcellent le réseau en de nombreux **petits** sous-groupes.

Sur le plan formel, la règle

$$\mathcal{H}(\mathcal{C}) > \theta \Rightarrow \text{scission de } \mathcal{C},$$

conduit à un découpage sans fin si θ ne tolère qu’un degré de cohésion interne excessivement élevé.

Dans certains systèmes top-down, on évalue la cohésion d’un cluster à chaque **itération**, sans laisser la dynamique locale (ex. mise à jour DSL) prendre le temps de renforcer les poids $\omega_{i,j}$ à l’intérieur du bloc. En ne laissant pas à la **synergie** le temps de se structurer, on perçoit hâtivement des failles internes. On segmente alors avant même que des liaisons fortes aient pu émerger.

Il s’ensuit une fragmentation rapide et parfois injustifiée : un bloc qui aurait pu devenir cohérent se voit coupé en deux, simplement parce qu’on n’a pas “attendu” la stabilisation des $\omega_{i,j}$.

Lorsque la division top-down ne s’accompagne d’aucune **possibilité de fusion** (retour arrière), chaque scission devient pratiquement **irréversible**. Ainsi, même si, à un moment ultérieur, des liens plus forts se forment entre deux blocs séparés, on reste avec un découpage trop fin.

Ce phénomène de **fragmentation irréversible** aboutit à une multitude de petits clusters : au fur et à mesure que la hiérarchie descend, elle ne propose jamais de *regroupement*, même si la dynamique change au sein du réseau.

B. Conséquences sur la structure multi-niveau

Un des apports majeurs d’une hiérarchie multi-niveau est de révéler des **clusters** de taille intermédiaire (mésos) ou des macro-groupes rassemblant plusieurs entités fortement reliées. La **sur-segmentation** fait disparaître ces blocs plus vastes, puisque tout se retrouve scindé en unités trop petites.

Cela réduit la **clarté** de la structure : on obtient des “feuilles” de très faible cardinalité au lieu de “nœuds” significatifs. Au lieu de manipuler quelques dizaines de super-nœuds, on se retrouve avec une kyrielle de mini-blocs. Pour les niveaux supérieurs de la hiérarchie, le gain de simplification initialement espéré (moins de ω à gérer) se voit annihilé : on doit traiter un trop grand nombre de partitions, ce qui complique la gestion du réseau.

Dans des applications (ex. IA cognitive, robotique), on souhaite parfois que des sous-groupes partagent des ressources ou des tâches. Une sur-segmentation aboutit à des groupes hyper-spécialisés qui ne communiquent pas ou peu entre eux, et on perd la **coopération** inter-bloc. On sacrifie alors l’effet de mutualisation que vise une structure multi-niveau.

C. Pistes pour y remédier

La première mesure consiste à **redéfinir** le critère de division pour être moins sévère. Autrement dit, laisser plus de marge à la cohésion interne. On peut aussi imposer qu’un bloc ne soit évalué qu’après un certain délai, permettant aux $\omega_{i,j}$ de se renforcer si la dynamique DSL locale le justifie.

Avant de décider d’une scission, on peut laisser le bloc “mûrir” sous la règle de plasticité (renforcement/décroissance) un certain nombre d’itérations. Ce laps de temps permet à la synergie de s’établir, révélant qu’un groupe est en fait cohérent, et évitant une coupe précipitée.

Comme en bottom-up, un mécanisme de **fusion** au même niveau (cf. 6.4.3.2 sur la coordination latérale) ou une boucle de *feedback* descendant (cf. 6.4) peut éviter les scissions définitives. Même si un bloc a été scindé, si l’on constate ultérieurement une interaction forte entre deux mini-blocs, on peut les **ré-agréger**.

Dans les grandes architectures DSL, on combine souvent **top-down** et **bottom-up** (cf. 6.5.3) : si le top-down divise trop, le *bottom-up* peut en parallèle fusionner certains blocs trop fins, assurant un équilibre entre scission et agrégation.

Conclusion

Lorsque la **logique descendante** (macro → micro) est appliquée de manière trop réactive ou sur la base d’un **seuil** trop exigeant, on assiste à un phénomène de **sur-segmentation** : les blocs se scindent en unités de plus en plus petites,

perdant l'essence d'une structuration hiérarchique utile. Les **causes** principales en sont des **critères de cohésion** excessivement stricts, l'absence de **délai** permettant à la synergie locale de s'installer et le manque de **mécanismes de réagrégation**.

Sur le plan **pratique**, il est donc crucial de **régler** avec soin la fonction \mathcal{H} (ou l'équivalent) et son seuil θ , d'introduire une **phase** d'auto-organisation avant toute scission, et/ou de permettre la **fusion** de blocs trop finement découpés. Cette approche prévient la fragmentation irréversible et préserve l'avantage d'une hiérarchie multi-niveau lisible, qu'on peut ensuite articuler avec d'autres mécanismes DSL (comme l'agrégation bottom-up ou la coordination latérale) pour obtenir une **structure** à la fois cohérente et flexible.

6.5.3. Hybridation (Approche Mixte)

Dans les sections précédentes (6.5.1 et 6.5.2), nous avons présenté deux grandes **orientations** pour structurer un **SCN** (Synergistic Connection Network) de manière multi-niveau : l'**agrégation progressive** (bottom-up) et la **division** (top-down). Cependant, dans la pratique, de nombreux **systèmes** tirent parti d'une **approche mixte** : on permet à la dynamique locale de **fusionner** des entités ou micro-clusters (vision bottom-up), *mais* on conserve également la possibilité de **scinder** (top-down) quand un cluster devient trop hétérogène. Cette **hybridation** procure davantage de **flexibilité** et évite les limites inhérentes à la pure agrégation (6.5.1.3) ou la pure division (6.5.2.3).

6.5.3.1. Combiner Agrégation Partielle et Division Sélective pour plus de Flexibilité

Les approches **bottom-up** (voir 6.5.1) et **top-down** (voir 6.5.2) présentent chacune des limites. L'agrégation ascendante peut rapidement verrouiller le réseau dans une configuration sous-optimale à cause de fusions irréversibles, tandis que la scission descendante risque d'aboutir à des partitions excessivement fines. Une **stratégie hybride** reposant sur l'alternance de **phases** d'agrégation et de **phases** de division permet de maintenir la construction hiérarchique plus fluide et mieux adaptée aux fluctuations de la **synergie** ou de l'hétérogénéité. Les développements qui suivent décrivent ce principe mixte et ses avantages, en cohérence avec la logique DSL multi-niveau.

A. Motivation

Les démarches purement ascendantes induisent parfois des verrous irréversibles : en se fondant sur de multiples agrégations (voir section 6.5.1), on peut constituer de gros blocs cohésifs trop tôt, rendant difficile toute redistribution des entités si ces macro-groupes se révèlent incohérents. À l'inverse, une division uniquement descendante (voir section 6.5.2) peut provoquer une segmentation trop poussée et fragmenter la hiérarchie en innombrables petits sous-ensembles. Il devient alors pertinent de recourir à une **hybridation** qui exploite à la fois la fusion lorsqu'une forte synergie est constatée et la division sélective pour corriger les cas de hétérogénéité interne. Cette oscillation entre fusion et scission empêche les blocages inhérents aux approches unilatérales.

B. Principe de l'Approche Mixte

Le noyau méthodologique réside dans l'emploi successif de l'agrégation locale et de la détection d'incohérences macro, menant à des divisions ciblées. Dans un premier temps, on laisse la dynamique **bottom-up** rassembler des entités en super-nœuds à l'aide de la règle

$$\Delta \omega_{i,j}(t) = \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

ce qui consolide les liens forts. Ensuite, si l'examen macro met en évidence une hétérogénéité dans un super-nœud $N_\alpha^{(k)}$, on introduit un mécanisme de division locale qui affaiblit certains liens internes et aboutit à un découpage partiel. Ainsi, le réseau peut se réorganiser en continu et affiner la hiérarchie en fonction des variations de la synergie.

C. Rôle du Feedback Descendant

Pour réaliser concrètement la scission dans un super-nœud hétérogène, on fait appel à un **flux descendant** (voir 6.4.1.2) fondé sur un signal inhibiteur visant des liens internes spécifiques. Sur le plan mathématique, on peut l'écrire comme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \Delta_{\text{split}},$$

où $\Delta_{\text{split}} > 0$ réduit progressivement la pondération $\omega_{i,j}$ dans la zone désignée. En affaiblissant graduellement ces liens, on amorce la division ; si la situation évolue, le flux descendant peut être stoppé, ce qui évite les ruptures irréversibles. Ce **feedback** autorise une scission sélective et réversible, guidée par le niveau macro, qui surveille et corrige les configurations trop vastes ou incohérentes.

D. Équilibre Continu et Flexibilité

Avec la faculté de réaliser simultanément des **fusions** (renforcement) et des **divisions** (inhibition sélective), le SCN converge vers un **état stable** garantissant qu'aucun super-nœud n'est à la fois trop vaste et trop hétérogène, ni que la synergie entre blocs distincts soit négligée. Les risques liés à l'agrégation irréversible (voir 6.5.1.3) et à la sur-segmentation (voir 6.5.2.3) se trouvent atténus, car le réseau peut revenir sur ses choix ou affiner la hiérarchie au fur et à mesure des signaux multiblocs et des évaluations macro. On obtient ainsi une forme d'**équilibre continu**, dans la mesure où la topologie hiérarchique peut évoluer librement en réponse à la dynamique des pondérations $\omega_{i,j}$.

E. Exemples de Mise en Œuvre

Dans les **réseaux sociaux**, l'agrégation locale repose sur l'union d'individus partageant un même intérêt ou un renforcement de leurs liens, tandis que la division sélective prend effet si un “super-groupe” devient conflictuel : le niveau supérieur (modérateur) enclenche alors une scission partielle pour séparer des factions. Le résultat est un réseau social plus flexible, à l'abri d'une unique communauté géante qui manquerait de cohésion.

Dans les **systèmes robotiques**, les robots qui coopèrent forment des équipes par agrégation. Un contrôle global peut néanmoins forcer la scission d'une sous-équipe si elle poursuit un objectif différent ou si la communication interne se dégrade, ce qui produit un SCN adaptatif où les équipes se forment et se défont en fonction des missions.

Dans les **systèmes sensoriels ou cognitifs**, on peut fusionner des capteurs (ou neurones) fortement corrélés, tandis que le niveau macro “coupe” un bloc bruyant ou conflictuel, isolant les signaux perturbateurs et réassignant éventuellement ces entités ailleurs. Cette logique reflète une **auto-organisation** à deux vitesses, ascendante pour la cohérence locale et descendante pour la répartition sélective.

F. Avantages de la Démarche Mixte

Cette démarche offre des **avantages** considérables en termes de **flexibilité et d'adaptation** : les erreurs typiques d'une approche purement bottom-up (accumulation d'erreurs, agrégations irréversibles) ou d'une approche purement top-down (fragmentation excessive) sont corrigées par la liberté de fission et de fusion contrôlées. Cette souplesse se traduit aussi par une **robustesse vis-à-vis du changement** : lorsque les conditions du réseau se modifient (insertion de nouvelles entités, évolutions de synergie), l'option de “fusion/division” se maintient, ce qui est crucial dans un contexte **temps réel** ou des flux de données constants. De plus, le **contrôle multi-niveau** assure que la logique DSL reste locale (pour le renforcement des liens) alors que l'analyse macro (pour la division éventuelle) s'opère en **feedback** descendant, permettant une régulation hiérarchique efficace.

G. Points à Surveiller

Un des **inconvénients** majeurs tient au risque d'oscillations, lorsque le réseau oscille entre un bloc fusionné puis scindé, puis refusionné. Pour éviter ce “va-et-vient” perpétuel, on introduit souvent des **seuils d'hystéresis** ou des **périodes de stabilisation**, empêchant la redécoupe immédiate d'un bloc à peine fusionné. La **complexité algorithmique** est également plus élevée, car on doit gérer simultanément les règles ascendantes et descendantes, définir des priorités entre fusion et scission, et paramétrier leur temporalité. Si le réseau n'exige pas un tel degré de flexibilité, la difficulté accrue peut ne pas se justifier. Cependant, la comparaison aux approches pures montre que cette méthode **mixte** gagne en souplesse et n'est que légèrement plus complexe à mettre en œuvre.

Conclusion (6.5.3.1)

L'hybridation de l'**agrégation partielle** (bottom-up) et de la **division sélective** (top-down) dans un **SCN** multi-niveau constitue un levier crucial pour éviter l'accumulation d'erreurs par des fusions irréversibles ou la sur-fragmentation de la structure. Le réseau peut auto-organiser des clusters stables là où la synergie est élevée, tout en restant capable de

scinder un groupe hétérogène, conférant plus de **flexibilité** et de **résilience**. Cette alternance de phases, renforcée par un **feedback** macro sélectif, apparaît donc comme une solution permettant au SCN de s'adapter en continu, de progresser vers un équilibre hiérarchique où chaque super-nœud demeure cohérent mais jamais définitivement verrouillé, et de satisfaire aux besoins d'un environnement changeant ou d'objectifs évolutifs.

6.5.3.2. Applications : Systèmes Complexes à Grande Échelle (Réseaux Sociaux, Sensoriels)

Les **systèmes complexes** de grande envergure, tels que les **réseaux sociaux** massifs ou les **réseaux de capteurs** (IoT), sont des terrains privilégiés pour mettre en œuvre l'**approche mixte** décrite précédemment (voir section 6.5.3.1). Dans ces environnements, la quantité d'entités peut s'élever à des milliers ou millions, et la structure de leurs liens $\omega_{i,j}$ ou de leur synergie évolue rapidement. Il s'avère essentiel d'y autoriser non seulement l'**agrégation ascendante** (mécanismes *bottom-up*) mais aussi la possibilité d'une **division sélective** (mécanismes *top-down*) afin d'ajuster en continu la hiérarchie et d'éviter la stagnation dans une configuration inadaptée. Cette **flexibilité** est d'autant plus cruciale que les liens se forment, se transforment ou disparaissent à un rythme soutenu, selon les interactions sociales ou les corrélations de mesure.

A. Réseaux Sociaux de Grande Échelle

Dans un **réseau social**, les utilisateurs peuvent être représentés par des nœuds $i \in \{1, \dots, n\}$, et les pondérations $\omega_{i,j}(t)$ reflètent la force de leur interaction à l'instant t . On peut définir une **similitude** ou **corrélation** $\text{corr}(X_i, X_j)$, où X_i et X_j sont des vecteurs ou historiques d'activité (échanges, centres d'intérêt, contenus partagés). La **dynamique DSL** appliquée localement en mode *bottom-up* prend souvent la forme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\text{corr}(X_i, X_j) - \tau \omega_{i,j}(t)],$$

ce qui consolide les liens entre utilisateurs fortement corrélés. Les agrégations successives (voir 6.5.1) aboutissent à des micro-communautés ou super-groupes, mais elles peuvent enfermer à long terme des factions distinctes dans un même bloc. La **division sélective** (voir 6.5.2) intervient alors pour scinder un macro-groupe jugé trop grand ou hétérogène, par exemple en introduisant un **feedback** descendant qui réduit la pondération $\omega_{i,j}$ entre sous-groupes “rivaux”. Concrètement, on ajoute un terme négatif $\gamma \Delta_{\text{desc}}$ si l'on repère un conflit interne, comme une divergence idéologique ou un clivage d'intérêt. Le réseau social peut ainsi “respirer” : il fusionne localement dès que la synergie est forte, puis scinde un ensemble si une scission partielle devient nécessaire. Les **exemples** incluent des communautés spontanées (rassemblées par un événement viral), qui se subdivisent ultérieurement quand différentes factions divergent. L'**avantage** tient dans la **réactivité** et la **scalabilité** : le système agrège ou segmente, préservant une hiérarchie modulaire, plutôt que de se figer en un seul bloc ou de se dissoudre en myriades de petits groupes.

B. Systèmes Sensoriels (IoT, Réseaux de Capteurs)

Dans un **réseau de capteurs** dispersés (IoT), le nombre d'appareils peut aussi atteindre l'échelle du million. Chaque capteur i fournit une mesure $X_i(t)$, et l'on peut définir la pondération $\omega_{i,j}$ à partir d'une **distance inversée** ou d'une **corrélation** sur les signaux $\{X_i(t)\}$. Une règle DSL *ascendante* :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\text{similarité}(X_i, X_j) - \tau \omega_{i,j}(t)]$$

permet la formation progressive de clusters ou zones cohérentes, rassemblant les capteurs dont les mesures sont proches. Cependant, un unique “super-cluster” peut s'avérer trop vaste et contenir des capteurs géographiquement ou fonctionnellement très différents. Le mécanisme de **division** top-down autorise alors la **scission** si un algorithme de contrôle macro détecte un “mélange” inattendu. Cette division, mise en œuvre par un signal descendant δ_{split} qui abaisse certaines $\omega_{i,j}$, préserve la possibilité d'une réorganisation spatiale : certains capteurs se regrouperont différemment, donnant lieu à une architecture multi-niveau plus stable et mieux segmentée.

La **flexibilité** de l'agrégation–division se révèle cruciale dans un système IoT où la topologie évolue (nouveaux capteurs, pannes, variations de signaux), et où la qualité de service dépend d'une segmentation performante. Plutôt que de se figer dans des macro-zones fixes, on s'adapte aux variations locales et globales.

C. Conclusion et Généralisation

De nombreux **systèmes complexes** — qu'il s'agisse de **réseaux sociaux** ou de **réseaux de capteurs** — profitent considérablement de la logique **hybride** (voir 6.5.3.1) qui allie l'agrégation ascendante et la division descendante. Les entités ou capteurs se regroupent localement lorsqu'une forte synergie ou similarité se manifeste, garantissant un gain d'organisation. Lorsqu'un macro-groupe se révèle trop grand ou hétéroclite, une **division sélective** assure un rééquilibrage. Cette cohabitation de fusion et scission, rendue possible par la **dynamique DSL** et le **feedback**, confère une **adaptation** permanente. Les **exemples** incluent des communautés sociales formées puis reconfigurées, ou des clusters de capteurs agrégés puis redécoupés. Les **formules** simples mentionnées pour la mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\text{corr}(X_i, X_j) - \tau \omega_{i,j}(t)] \quad (-\gamma \Delta_{\text{desc}} \text{ si scission}).$$

illustrent comment l'agrégation correspond à un **renforcement** de $\omega_{i,j}$ pour $\text{corr}(X_i, X_j)$ élevée, alors que la division se modélise par un terme négatif $-\gamma \Delta_{\text{desc}}$. Dans l'un et l'autre cas, l'**auto-organisation** hiérarchique demeure en mouvement : l'approche mixte se révèle alors la plus à même de gérer la variabilité et l'évolution à grande échelle de tels réseaux.

6.5.3.3. Observation des Patterns Fractals si le Cycle Agrégation–Division Suit une Logique de Récurrence

Cadre et Motivation. Les sections précédentes ont montré qu'une **approche hybride** (section 6.5.3.1) combine les mécanismes **bottom-up** (agrégation) et **top-down** (division) dans la construction hiérarchique d'un **SCN** (Synergistic Connection Network). Il apparaît, dans certaines configurations, qu'un **cycle** récurrent d'agrégation et de division puisse émerger : des groupes se forment lorsque leur synergie est élevée, puis se scindent à nouveau lorsqu'ils deviennent trop hétérogènes. Lorsqu'une telle dynamique se reproduit à différents paliers (ou échelles) avec la même *loi* d'auto-organisation, on constate parfois l'apparition de **patterns fractals**, traduisant une **auto-similarité** ou **invariance d'échelle**. Les développements qui suivent analysent comment ce cycle agrégation–division engendre des régularités fractales, et quelles hypothèses sont nécessaires pour les observer.

A. Logique de Répétition et d'Échelle

La notion de **fractale** (chap. 6.3) se définit classiquement par une propriété d'**auto-similarité** : le même schéma de construction (ou d'évolution) se reproduit à plusieurs niveaux de **granularité**, sans introduire d'échelle caractéristique fixe. Dans le contexte d'un **cycle** combinant agrégation et division, ce comportement se concrétise si, à chaque itération ou à chaque palier, la *même* règle DSL localement et la *même* procédure de scission globalement se déploient.

On suppose qu'à un moment donné, des entités (ou clusters) se **fusionnent** partiellement si leur pondération $\omega_{i,j}$ est assez élevée, conduisant à la création de super-nœuds. Dès qu'un super-nœud devient lui-même trop grand ou trop hétérogène (critère \mathcal{H} trop élevé), une **division** le scinde en sous-blocs. Les deux (ou plusieurs) blocs nés de la scission peuvent à leur tour vivre la même alternance de fusion avec d'autres blocs et de division interne. Cette **référence** d'un schéma de “fusion si cohérent, division si incohérent” constitue une base favorable à l'**invariance d'échelle**.

Pour que cette récurrence se traduise en **patterns fractals**, il faut que les **paramètres** η , τ , ou le critère \mathcal{H} de division ne fixent pas un unique “seuil absolu” de taille, mais soient paramétrés **relativement** à la structure courante. Autrement dit, si, à chaque échelle, la règle DSL et la procédure de scission conservent la *même forme*, on ne rompt pas la *loi de croissance* ni la *loi de découpage*. Dès lors, le système peut afficher des **lois de puissance** en termes de distribution de tailles de clusters, signatures typiques des structures fractales.

On peut imaginer un bloc \mathcal{C} de taille N . À chaque cycle, un sous-ensemble de αN entités fusionnent avec βN entités voisines si leur synergie dépasse un **seuil** relatif, puis on scinde γN entités si leur cohésion chute en deçà d'un certain ratio. Une telle règle, conservée à chaque palier, entretient un “même flux” à chaque échelle, propice à la fractalité.

B. Mécanisme de Stabilisation Fractale

Le **DSL** (Deep Synergy Learning) met à jour les pondérations $\omega_{i,j}$ sous une forme générale :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \Delta_{\text{feedback}}(i,j),$$

où $\eta > 0$ et $\tau > 0$ paramètrent le renforcement/décroissance, tandis que Δ_{feedback} reflète les influences descendantes (scission sélective) ou latérales (coordination). Si Δ_{feedback} applique la même forme d'inhibition (ou de récompense) à chaque "palier" avec un rescaling adapté à la taille courante, la dynamique DSL aboutit parfois à un **cycle récurrent** ; des clusters se forment par renforcement, s'agrandissent, puis reçoivent un retour descendant provoquant une scission partielle lorsque leur hétérogénéité interne augmente.

Un **renforcement local** se produit lorsque les entités ou *clusters* présentant une synergie élevée $S(i,j)$ consolident leurs liens et se rassemblent en blocs plus vastes, tandis qu'une **inhibition macro** s'exerce si un bloc C_α présente une trop grande dispersion, dans ce cas, une procédure descendante réduit spécifiquement les pondérations $\omega_{i,j}$ responsables de la discorde, forçant ainsi la scission de C_α en sous-blocs plus homogènes. Cette dynamique se **répète** à chaque niveau : dans un esprit fractal, on réapplique la même logique indépendamment du fait que C_α regroupe 50, 500 ou 50 000 entités, si bien que l'on observe des **motifs auto-similaires**, avec agrégation localement et division globalement, à chaque échelle.

C. Observations Pratiques et Limites

Les **patterns fractals** déployés dans un réseau hiérarchique reposent sur l'idée que la **même** combinaison de renforcement et d'**inhibition** agit à toutes les échelles de la structure. Cette exigence d'**auto-similarité** peut cependant être mise en péril si l'on introduit une rupture de symétrie d'échelle. Il se produit alors une "cassure" dans la cohérence fractale si, par exemple, les seuils de scission θ_k diffèrent trop fortement d'un palier k à l'autre, ou si le taux η est multiplié par un facteur grandissant à mesure qu'on progresse dans la hiérarchie. Il en résulte qu'une échelle caractéristique apparaît et détruit le principe d'invariance.

Un second écueil consiste en un **risque d'oscillation** voire de quasi-chaos si la logique macro descendante intervient de manière trop fréquente. Il se peut qu'un cluster émerge localement, puis se retrouve scindé au niveau supérieur, puis se reforme, engendrant une dynamique d'allers et retours incessants. Pour qu'une **structure fractale** stable (ou un régime quasi-périodique) puisse s'établir, il est souvent nécessaire d'introduire un mécanisme d'**hystéresis** ou un délai temporel dans le déclenchement des fusions et divisions, afin de prévenir les réorganisations à trop haute fréquence.

Un troisième point d'intérêt réside dans les **exemples de lois de puissance** qui peuvent s'ensuivre. On peut observer, dans certains scénarios, un spectre des tailles de clusters satisfaisant une **distribution** $P(N) \propto N^{-\alpha}$. Dans ce cas, on se situe face à un **comportement fractal** (ou un état critique) qui se manifeste lorsqu'il n'existe pas d'échelle caractéristique imposée au réseau. Pour vérifier cette structure "scale-free", on recourt fréquemment à des analyses en représentation log-log ou à des mesures de **box-counting** permettant d'évaluer la dimension fractale du système, que ce soit dans la topologie spatiale ou dans la distribution des connexions. Il convient de souligner que cette propriété n'est pas automatique : un suivi rigoureux du réseau est nécessaire pour confirmer que l'**auto-organisation** aboutit bien à une loi de puissance significative, et le moindre paramètre susceptible de varier d'un palier à l'autre peut rompre cette loi et faire disparaître la fractalité attendue.

Conclusion (6.5.3.3)

Lorsque le **cycle agrégation-division** (voir 6.5.3.1) s'opère de manière **récurrente**, sans échelle de taille imposée, il peut conférer au SCN une forme d'**auto-similarité** dans son organisation. Des *patterns fractals* apparaissent alors, témoignant d'une **invariance d'échelle** : la même dynamique DSL (renforcement local, scission globale) se reproduit à différentes granularités, si bien que l'on peut observer des lois de puissance ou des signatures fractales dans la distribution des tailles de blocs ou dans la structure des liens $\omega_{i,j}$.

La **logique** fractale se caractérise par une **loi** de répétition identique à chaque palier : la fusion opère sur les paires (ou micro-clusters) synergiques, la division intervient sur tout bloc hétérogène, et le tout se répercute sur l'ensemble du réseau *sans* briser la cohérence des paramètres. Dans un tel cadre, le SCN se transforme en un système *autosimilaire*, où la hiérarchie ne se fige jamais définitivement, mais évolue par un enchaînement continu de fusions partielles et de scissions sélectives, générant un motif fractal *persistant* ou *périodique*.

6.5.4. Algorithmes de Mise en Œuvre

Après avoir décrit les principes (agrégation vs. division) et l'idée d'une approche **mixte** (6.5.3), il est temps d'aborder plus **concrètement** la dimension algorithme. La section 6.5.4.1 propose un **pseudo-code** illustratif qui montre comment, dans un SCN (Synergistic Connection Network), on peut :

419. **Déetecter les micro-clusters**,
420. **Fusionner** ces micro-clusters en “super-nœuds”,
421. Gérer un **contrôle macro** (top-down) pour scinder ou valider ces super-nœuds.

6.5.4.1. Pseudo-Codes : (1) Détection Micro-Cluster, (2) Fusion en Super-Nœud, (3) Contrôle des Macro-Niveaux

Cadre Global. L'objectif est de décrire, sous forme de pseudo-codes, l'articulation pratique de la **dynamique locale** (déttection de micro-clusters via DSL), de la **fusion** en super-nœuds (agrégation), et du **contrôle** macro (avec possibilité de division). On suppose qu'au **niveau micro** (ou $\ell = 0$), on gère un ensemble d'entités $\{\mathcal{E}_i\}_{i=1\dots n}$ reliées par des pondérations $\omega_{i,j}^{(0)}$. L'itération de cette procédure peut être prolongée vers des **niveaux 1, 2, ... jusqu'à un palier macro ou intermédiaire**.

La structure principale se décompose en trois volets :

422. Une **dynamique locale** qui met à jour $\omega_{i,j}^{(0)}$ selon la règle DSL et détecte les micro-clusters.
423. Une **fusion** (agrégation) pour construire des super-nœuds à partir des clusters détectés.
424. Un **contrôle** macro, qui valide ou scinde (division) les super-nœuds jugés trop hétérogènes.

La présentation qui suit reste schématique. Les détails algorithmiques (complexité, choix des seuils, etc.) sont modulables.

A. Détection Micro-Cluster : la Dynamique Locale

Ce premier pseudo-code réalise la **mise à jour** DSL au **niveau micro** (ou palier $\ell = 0$), puis détecte des **clusters** localement, qualifiés de *micro-clusters*.

```

1: Initialize ω(i,j)^{(0)} for all (i,j) with small or random values
2: for t = 1 to T_local do
3:   for each pair (i,j):
4:     ω(i,j)^{(0)} ← ω(i,j)^{(0)} + η₀ * [ S₀(i,j) - τ₀ * ω(i,j)^{(0)} ]
5:   # Optionally clamp or saturate ω(i,j)^{(0)} to [0,1] or a chosen range
6: end for

7: clusters_level0 ← find_clusters( ω(·,·)^{(0)}, threshold_local )
  # e.g., BFS or community detection where ω^{(0)}(i,j) > threshold_local

```

Interprétation **Mathématique.**
Le taux $\eta_0 > 0$ et le terme $\tau_0 > 0$ régulent la vitesse d'adaptation, et $S_0(i,j)$ indique la **synergie** ou **similarité** entre entités \mathcal{E}_i et \mathcal{E}_j . La fonction *find_clusters* détecte des composantes connexes si $\omega_{i,j}^{(0)}$ dépasse un certain θ_{loc} , ou exploite un algorithme de partition (ex. Louvain, *community detection* légère) adapté à la densité.

La ligne 5 illustre la possibilité de forcer $\omega_{i,j}^{(0)}$ à rester dans $[0,1]$, une pratique courante pour éviter la divergence ou les valeurs négatives.

B. Fusion en Super-Nœuds (Agrégation)

Une fois identifiés, les micro-clusters de **niveau 0** sont transformés en **super-nœuds** au **niveau 1**. On reconstruit alors les pondérations $\omega^{(1)}$ entre ces super-nœuds via une fonction Ψ .

```

9: super_nodes_level1 = {}
10: for each cluster C $\alpha$  in clusters_level0:
11:   create super-node N $\alpha^{(1)}$ 
12:   N $\alpha^{(1)}$ .members = C $\alpha$ 
13:   super_nodes_level1.add(N $\alpha^{(1)}$ )

14: for each pair (N $\alpha^{(1)}$ , N $\beta^{(1)}$ ):
15:    $\omega_{\alpha,\beta}^{(1)} = \Psi(\{\omega_{i,j}^{(0)} | i \in C_\alpha, j \in C_\beta\})$ 
```

Description.

Les lignes 10–13 établissent une correspondance directe entre un cluster $C_\alpha^{(0)}$ détecté et un super-nœud $N_\alpha^{(1)}$. La fonction Ψ agrège les valeurs $\omega_{i,j}^{(0)}$ entre les entités internes à C_α et C_β . Typiquement, Ψ peut être :

$$\omega_{\alpha,\beta}^{(1)} = \Psi(\{\omega_{i,j}^{(0)} : i \in C_\alpha, j \in C_\beta\}),$$

où Ψ est une somme, une moyenne, un maximum, etc. Ce calcul produit une **nouvelle** matrice $\omega^{(1)}$ représentant les liens entre super-nœuds $\{N_\alpha^{(1)}\}$.

C. Contrôle des Macro-Niveaux : Division ou Validation

Au **niveau 1**, chaque super-nœud $N_\alpha^{(1)}$ peut être jugé **hétérogène** ou **cohésif**. En cas d'hétérogénéité, une **scission** (top-down) est possible. Le pseudo-code ci-après montre une version simplifiée de cette opération :

```

18: for each super-node N $\alpha^{(1)}$  in super_nodes_level1:
19:   H $\alpha$  = heterogeneity(N $\alpha^{(1)}$ ,  $\omega^{(1)}$ )
20:   if H $\alpha$  > threshold_macro:
21:     sub_clusts = subdivide(N $\alpha^{(1)}$ .members,  $\omega^{(0)}$ )
      # Possibly do BFS or community detection again at level 0
22:     remove N $\alpha^{(1)}$  from super_nodes_level1
23:     for each subc in sub_clusts:
24:       Nx = create new super-node
25:       Nx.members = subc
26:       super_nodes_level1.add(Nx)
27: re_compute  $\omega^{(1)}$  using function  $\Psi(\cdot)$  # or partial update
```

Explication.

La fonction *heterogeneity* (ligne 19) calcule un indice \mathcal{H} de cohésion interne. Si $\mathcal{H}(N_\alpha^{(1)}) > \theta_{macro}$, la procédure *subdivide* revisite les entités $E_i \in N_\alpha^{(1)}$. *members* et cherche une partition interne plus cohérente (lignes 21–25). Les sous-blocs substituent le super-nœud $N_\alpha^{(1)}$ d'origine dans *super_nodes_level1*. On met ensuite à jour la matrice $\omega^{(1)}$, voire on itère une dynamique DSL au niveau 1.

D. Intégration en Boucle Itérative

Pour former une **architecture hiérarchique** (et pas seulement un niveau 0 puis un niveau 1), on répète le principe suivant :

425. **Mise à jour DSL** au palier $\ell = 0$ et **détection** micro-clusters,

426. **Fusion** de ces micro-clusters en super-nœuds $N_\alpha^{(1)}$,

427. **Contrôle** macro (ligne 18+) : on valide ou on subdivise si besoin,

428. Si la taille du réseau l'exige, on réapplique un **processus** DSL au niveau 1 avec η_1, τ_1 , on détecte des clusters de niveau 1, on crée des super-nœuds $\mathcal{N}_\alpha^{(2)}$, etc.

La hiérarchie s'échelonne en paliers $k = 0, 1, 2, \dots$ jusqu'à atteindre un niveau **macro** où il ne reste plus qu'un petit nombre de super-nœuds ou un unique bloc. On peut inclure un **flux descendant** correctif pour forcer une scission sélective, notamment si un super-nœud est repéré comme trop hétérogène.

E. Commentaires Mathématiques et Convergence

Lien avec les Algorithmes de Community Detection. Le schéma ci-dessus s'apparente à une **détection de communautés** multi-niveau. Toutefois, la spécificité **DSL** réside dans la **mise à jour** $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S - \tau\omega_{i,j}(t)]$ qui fait émerger les clusters plutôt que de simplement les "découper".

Complexité et Approches Heuristiques. Chaque niveau requiert un **find_clusters** (ligne 8) et un **subdivide** (ligne 21) potentiellement complexes. On recourt souvent à des heuristiques (BFS sur les liens $> \theta$, modularité, etc.) pour demeurer en $O(n\log n)$ ou $O(n)$ si le graphe est épars. Les algorithmes exacts de partition restent NP-difficiles pour n

grandConvergence ou Évolution Continue.

Lorsque l'environnement est **statique**, on peut parvenir à un état stable : plus aucun super-nœud n'est scindé, plus aucun micro-cluster ne se modifie. Dans des **contextes dynamiques** (flux de données, changements de liens), on exécute ces boucles en continu, adaptant la hiérarchie en **temps réel** (chap. 9).

Conclusion (6.5.4.1)

Le **pseudo-code** présenté illustre une **architecture** typique :

429. **Dynamique locale** :

on applique le **DSL** au **niveau** micro ($\ell = 0$), détecte des **clusters** via un algorithme de sur-seuil ou de community detection simplifiée.

430. **Fusion** :

on convertit ces **clusters** en **super-nœuds** $\mathcal{N}_\alpha^{(1)}$, reconstruisant une matrice $\omega^{(1)}$ par une fonction Ψ .

431. **Contrôle macro** :

on valide ou subdivise un super-nœud $\mathcal{N}_\alpha^{(1)}$ jugé trop hétérogène, en créant de **nouveaux** super-nœuds plus petits.

Répéter cette logique par itérations successives et paliers multiples ($k = 0, 1, 2, \dots$) bâtit une **hiérarchie** complète, exploitant l'**auto-organisation** locale (bottom-up) et le **contrôle** ou la **sélection** globale (top-down). Les sections suivantes (6.5.4.2 et 6.5.4.3) approfondiront le réglage des **paramètres** (seuils, etc.) et l'extension à des variantes (inhibition compétitive, recuit simulé) pour assurer une convergence stable et éviter la stagnation dans des partitions sous-optimales.

6.5.4.2. Paramètres Clés (Seuil de Synergie, Ratio d'Homogénéité, etc.)

Contexte et Problématique. Les algorithmes multi-niveau décrits précédemment (section 6.5.4.1) reposent sur une alternance entre l'**agrégation** (bottom-up) et la **division** (top-down). Cette dynamique implique divers **paramètres** cruciaux qui déterminent les conditions de fusion et de scission, et donc la forme finale de la hiérarchie. Parmi ces paramètres, on trouve notamment le **seuil de synergie** dictant la fusion locale et le **ratio d'homogénéité** influençant la division macro. D'autres, comme les taux η et les facteurs τ de la mise à jour DSL, ou la fonction Ψ d'agrégation, ont également un impact décisif. Les développements ci-après décrivent comment ces paramètres s'imbriquent et comment ils orientent l'équilibre entre la formation de super-nœuds et leur éventuelle scission.

A. Seuil de Synergie θ_{synergy}

Dans la démarche **bottom-up**, un **seuil** de synergie θ_{synergy} sert à déterminer quand deux entités (ou deux clusters) méritent d'être **fusionnés**. Mathématiquement, on peut considérer qu'on ne fusionne un couple $\{i, j\}$ que si $\omega_{i,j} \geq \theta_{\text{synergy}}$. De même, dans l'algorithme "find_clusters", on ne retient que les arêtes du graphe dont la pondération dépasse ce seuil. Cela aboutit à des composantes connexes que l'on assimile à des **micro-clusters**.

Un θ_{synergy} trop élevé force une exigence stricte de cohérence, ce qui restreint la fusion à des liens très forts. Cela tend à créer un nombre important de **petits** clusters ou à empêcher la formation de sous-groupes plus volumineux. Au contraire, un θ_{synergy} trop bas mène à des blocs de grande taille peu homogènes, qu'il faudra ensuite probablement **scinder** (top-down) en raison d'une hétérogénéité interne excessive.

Il existe une manière de rendre ce seuil **adaptatif** : au lieu de fixer θ_{synergy} a priori, on peut le définir comme un quantile de la distribution des $\omega_{i,j}$. Par exemple, retenir les 20% de liens les plus forts assure un pourcentage prévisible de liaisons considérées "actives" pour la fusion, et donc un nombre de clusters local plus stable.

B. Ratio d'Homogénéité / Hétérogénéité θ_{macro}

L'**hétérogénéité** $\mathcal{H}(\mathcal{C})$ est un indicateur crucial dans la phase **top-down**, au moment de décider la scission d'un super-nœud \mathcal{C} . Cette hétérogénéité peut se définir sous différentes formes : variance des pondérations $\omega_{i,j}$, écart absolu moyen, ou encore ratio entre liens forts et liens faibles. Une formulation usuelle s'appuie sur la somme des écarts entre chaque poids $\omega_{i,j}$ et la moyenne des poids internes. Plus précisément, on introduit

$$\mathcal{H}(\mathcal{C}) = \frac{1}{|\mathcal{C}|^2} \sum_{i,j \in \mathcal{C}} |\omega_{i,j} - \bar{\omega}_{\mathcal{C}}|$$

où $\bar{\omega}_{\mathcal{C}}$ désigne la moyenne des $\omega_{i,j}$ pour les entités internes à \mathcal{C} . Si la somme des écarts $|\omega_{i,j} - \bar{\omega}_{\mathcal{C}}|$ se révèle élevée, alors \mathcal{C} présente un fort degré d'hétérogénéité.

Le **critère de scission** consiste à fixer un **seuil** θ_{macro} . Lorsque $\mathcal{H}(\mathcal{C})$ dépasse θ_{macro} , on considère que la cohésion interne du super-nœud est insuffisante et qu'il convient d'opérer une division. Conformément au **mécanisme top-down**, un **feedback** descendant vient alors réduire spécifiquement certaines liaisons $\omega_{i,j}$ au sein de \mathcal{C} , ce qui "casse" la communauté en sous-groupes plus homogènes. Un θ_{macro} trop élevé entraîne une scission uniquement pour les blocs très hétéroclites, au risque de conserver parfois des clusters imposants. Inversement, un θ_{macro} trop bas suscite une **sur-segmentation** (cf. 6.5.2.3), dans laquelle même de modestes divergences déclenchent la division.

Le **paramétrage adaptatif** offre la possibilité de modifier θ_{macro} au fil de l'évolution du réseau ou selon la distribution de $\mathcal{H}(\cdot)$. Par exemple, on peut sélectionner un quantile donné pour définir la valeur du seuil, ou ajuster θ_{macro} en fonction de la phase d'itération. Cette démarche assure une maîtrise plus fine du rythme de scission au niveau macro, conciliant la nécessité d'homogénéité dans un super-nœud et la flexibilité requise pour l'**auto-organisation** hiérarchique.

C. Paramètres de Vitesse : η et τ dans la Dynamique DSL

La mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ dépend de manière linéaire du **taux d'apprentissage** η . Lorsque η est trop élevé, les variations de ω sont rapides, voire abruptes, pouvant générer des **oscillations** ou déstabiliser le réseau. À l'inverse, un η très petit éteint la convergence sur un nombre important d'itérations, ralentissant la capacité du système à repérer et consolider les clusters. Dans un **SCN** hiérarchique, on adopte fréquemment un η_0 relativement grand au **niveau micro** pour identifier les clusters plus vite, tandis que, pour les niveaux macro ($\ell = 1, 2, \dots$), on utilise un η_k plus faible afin d'éviter que la structure globale ne se reconfigure trop brutalement au moindre changement.

Le **facteur** $\tau > 0$ représente la décroissance dans la même dynamique, par l'expression $\tau \omega_{i,j}$ à retrancher de la mise à jour. Cet élément évite la croissance sans limite des pondérations et limite la persistance de liaisons peu utiles. Un τ plus grand accentue l'**oubli**, c'est-à-dire que tout lien non soutenu par une synergie $S(i,j)$ élevée tendra rapidement

vers zéro. À l'inverse, un τ trop petit maintient un réseau plus **dense** : on y conserve de nombreux liens faibles, ce qui accroît la complexité de la structure et en diminue la sélectivité dans la formation des clusters.

Du fait de l'existence de **niveaux multiples** (ℓ), il est envisageable d'assigner à chaque palier ℓ ses propres valeurs η_ℓ et τ_ℓ . De cette façon, la **dynamique DSL** se module en fonction de la granularité. Une approche classique consiste à rendre η_0 (et parfois τ_0) relativement importants, de sorte que le niveau micro réagisse vivement pour créer ou dissoudre rapidement des liaisons, et à choisir η_k, τ_k plus tempérés lorsqu'on s'élève dans la hiérarchie, de manière à ne pas ébranler trop fréquemment la configuration macro. En procédant ainsi, on préserve la **réactivité** souhaitée dans les couches inférieures, tout en garantissant une plus grande **stabilité** du réseau aux niveaux supérieurs.

D. Paramètres de Filtrage et de Sélection dans l'Agrégation

Lors de la création de **super-nœuds** au niveau $\ell + 1$ à partir de **clusters** de niveau ℓ , on introduit une **fonction** Ψ pour définir la pondération $\omega_{\alpha,\beta}^{(\ell+1)}$. De manière générale, on écrit

$$\omega_{\alpha,\beta}^{(\ell+1)} = \Psi(\{\omega_{i,j}^{(\ell)} \mid i \in \mathcal{C}_\alpha^{(\ell)}, j \in \mathcal{C}_\beta^{(\ell)}\}),$$

ce qui signifie que l'on agrège l'ensemble des liens $\omega_{i,j}^{(\ell)}$ reliant les entités du bloc $\mathcal{C}_\alpha^{(\ell)}$ à celles du bloc $\mathcal{C}_\beta^{(\ell)}$. Le **choix** de Ψ peut prendre diverses formes : somme, moyenne, maximum, ou même un **minimum** ou un quantile, selon la logique du réseau. Un **maximum** favorise la propagation des liens les plus forts et peut surestimer la synergie globale, alors qu'une **moyenne** rend l'agrégation plus modérée. Dans certains contextes, un **minimum** ou un **quantile** s'avère judicieux pour refléter la robustesse d'un lien dominant ou, au contraire, l'homogénéité générale d'un ensemble. Une fois $\omega_{\alpha,\beta}^{(\ell+1)}$ calculé, on recourt fréquemment à un **filtrage** ou à un **seuil de transmission** $\theta_{(\ell+1)}$: on ne conserve que les super-arêtes dont la pondération dépasse $\theta_{(\ell+1)}$, ce qui limite la densité du graphe au palier macro ou méso. Cette opération influence directement la forme hiérarchique résultante, car elle régit la connectivité entre super-nœuds : si le filtre est trop strict, on risque d'aboutir à une structure dispersée de blocs faiblement reliés ; s'il est trop permissif, le réseau demeure très dense, et les fusions tendent à se multiplier, parfois sans réelle cohésion. L'équilibre entre Ψ et le filtrage $\theta_{(\ell+1)}$ s'avère donc déterminant pour réguler la **logique DSL** à chaque palier et assurer une **auto-organisation** stable et adaptée au niveau d'échelle visé.

E. Combinaisons et Calibration Globale

L'**hystérosis** et l'introduction de **fenêtres de stabilisation** constituent l'une des clés pour limiter les oscillations brutales dans un système multi-niveau. Dans un **algorithme DSL** complexifié, on empêche qu'une fusion ou une scission déjà réalisée ne soit immédiatement inversée par la phase suivante. On se dote par exemple de deux seuils de fusion, $\theta_{\text{fusion}}^{\text{high}}$ et $\theta_{\text{fusion}}^{\text{low}}$, afin de fusionner deux blocs si $\omega_{i,j}$ dépasse $\theta_{\text{fusion}}^{\text{high}}$, tout en s'assurant de ne pas "dé-fusionner" tant que le poids reste au-dessus de $\theta_{\text{fusion}}^{\text{low}}$. Un principe analogue s'applique aux divisions : un bloc fraîchement scindé ne sera pas ré-agrégré si sa pondération ou sa cohésion interne n'a pas franchi un seuil supérieur. Cette stratégie atténue le "va-et-vient" permanent, permettant une **stabilisation** avant que le mécanisme inverse ne prenne le relais.

La **gestion multi-échelle** impose souvent que chaque niveau ℓ du réseau hiérarchique puisse définir ses propres paramètres $\{\eta_\ell, \tau_\ell, \theta_{\text{synergy}}^{(\ell)}, \theta_{\text{macro}}^{(\ell)}\}$. Cette variabilité rend possible la **réactivité** à petite échelle (où η_0 et τ_0 sont assez importants pour permettre des ajustements rapides) et la **stabilité** à grande échelle (où η_k et τ_k décroissent ou se fixent à des valeurs plus modérées). Bien que cette souplesse des réglages soit précieuse, elle complique la configuration globale : dans la pratique, on recourt à des essais ou à une adaptation automatique pour ajuster ces seuils et vitesses de mise à jour.

Les **boucles d'itération** mettent en jeu trois moments essentiels : la mise à jour **DSL locale** (avec η_0, τ_0) pour consolider ou affaiblir les liens $\omega_{i,j}$ selon la synergie locale ; la **détection/fusion** (à l'aide d'un seuil θ_{synergy}) pour agréger de manière ascendante les entités ou sous-blocs très cohérents ; et la **validation ou division macro** (fondée sur la mesure d'hétérogénéité $\mathcal{H}(\mathcal{C})$ et un seuil θ_{macro}) pour scinder de manière descendante les blocs jugés trop dispersés. Il convient de calibrer l'ordre et la périodicité de ces actions pour obtenir une convergence harmonieuse : un déclenchement trop rapide du feedback macro risquerait de démanteler des clusters encore en formation, tandis qu'un retard excessif dans la division maintiendrait un super-bloc incohérent. Cette **cohérence globale** dans l'emploi

successif de la dynamique locale, de l'agrégation et de la division constitue l'un des points cruciaux de la **conception** d'un SCN multi-niveau.

Conclusion

La **réussite** d'un algorithme multi-niveau DSL, associant agrégation et division, dépend de **plusieurs** paramètres qui interagissent :

- 432. **Seuil de synergie** θ_{synergy} , fixant la condition de fusion locale,
- 433. **Ratio d'homogénéité** θ_{macro} , fixant la condition de division macro,
- 434. **Paramètres DSL** η_ℓ et τ_ℓ régulant la vitesse et la dissipation au palier ℓ ,
- 435. **Fonction d'agrégation** Ψ , modulant la formation des pondérations au niveau supérieur,
- 436. **Filtrage et hystéresis**, contrant les oscillations et assurant un comportement stable.

Ces paramètres façonnent la **structure** hiérarchique et sont souvent **calibrés** empiriquement ou via des heuristiques adaptatives. Un mauvais réglage peut entraîner un trop grand regroupement (clustering géant peu cohérent) ou, à l'inverse, une trop forte segmentation (sur-fragmentation). Une bonne **combinaison** fournit un **SCN** multi-niveau **robuste, flexible, et capable** de s'auto-organiser en réponses aux signaux locaux (bottom-up) comme aux contraintes globales (top-down).

6.5.4.3. Intégration avec l'Inhibition ou le Recuit (Chap. 7) pour Éviter la Stagnation

Introduction et Contexte. Les algorithmes multi-niveau (section 6.5.4.1) reposant sur l'**agrégation** (bottom-up) et la **division** (top-down) nécessitent un certain "dynamisme" pour évoluer vers des structures hiérarchiques suffisamment pertinentes. Or il arrive qu'un **blocage** ou une **stagnation** se produise : le réseau se fige dans une configuration localement stable sans l'être globalement, ou bien certaines liaisons $\omega_{i,j}$ conservent des valeurs moyennes, empêchant la différenciation de sous-groupes plus cohérents. Pour s'extraire de ces situations, on recourt à des mécanismes complémentaires comme l'**inhibition** ou le **recuit simulé**, décrits en détail au **Chap. 7**. Les développements ci-après résument l'idée et montrent comment ces procédés s'intègrent dans le cadre DSL multi-niveau.

A. Principe de l'Inhibition

Dans la dynamique DSL classique, la mise à jour des pondérations suit la formule $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$. Il est cependant possible d'ajouter un **terme d'inhibition** pour contraindre la somme des liens sortants $\sum_k \omega_{i,k}(t)$ et inciter chaque nœud \mathcal{E}_i à limiter la densité de ses connexions. Une forme représentative s'écrit

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

avec $\gamma > 0$ modulant la force inhibitrice. Ce mécanisme, qualifié d'**inhibition latérale ou globale**, a pour effet de "pousser" chaque entité i à sélectionner quelques liaisons prioritaires tout en relâchant les autres, ce qui favorise un **contraste** plus marqué entre liens forts et liens faibles et empêche la stagnation dans un état moyennement connecté.

Lorsque l'on souhaite éviter qu'un cluster se stabilise dans une configuration sous-optimale où de nombreux liens médians subsistent, l'inhibition pousse en effet $\omega_{i,j}$ vers zéro sauf pour les paires (i,j) réellement synergiques.

Sur un plan **multi-niveau**, le palier supérieur peut également déclencher une inhibition partielle pour des super-nœuds surconnectés, limitant ainsi leur extension artificielle et rejoignant la logique d'une division top-down : si un bloc est trop vaste ou trop hétéroclite, un **feedback** descendant peut diminuer certains liens internes et ouvrir la voie à une scission ultérieure, garante d'une structure globale mieux équilibrée.

B. Principe du Recuit Simulé (Injection de Bruit)

Le **recuit simulé** introduit un terme aléatoire dans la mise à jour des pondérations afin d'échapper aux états de blocage ou aux minima locaux. La **dynamique** se généralise alors en

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ correspond à un bruit (gaussien, uniforme) et $\sigma(t)$ définit une “température” positive qui décroît au fil des itérations ($\sigma(t) \rightarrow 0$ quand $t \rightarrow \infty$). Lorsque $\sigma(t)$ est élevée, les fluctuations dans $\omega_{i,j}$ demeurent importantes, ce qui encourage une **exploration** de nombreuses configurations. À mesure que la température diminue, le réseau “gèle” progressivement sa structure dans un état supposé plus stable ou plus optimal. L’absence de bruit dans un **DSL** peut conduire à une convergence précoce vers un **minimum local**, alors que l’injection contrôlée de $\sigma(t)$ autorise certaines liaisons $\omega_{i,j}$ à monter ou à descendre aléatoirement, même si la synergie $S(i,j)$ n'est pas très marquée, révélant ainsi des sous-groupes latents ou détruisant des connexions faiblement fondées. L’**agrégation** se trouve de ce fait renforcée lorsqu'un couple (i,j) bénéficie d'un surcroît de bruit positif, et la **division** top-down peut s’opérer plus facilement si le bruit négatif fragilise des liens internes déjà instables. Au final, le recuit simulé, par la variation $\sigma(t)$, offre une flexibilité complémentaire à la dynamique DSL, favorisant l'**exploration** quand la température est haute et la **consolidation** quand la température se rapproche de zéro.

C. Combinaison dans un Cycle Multi-Niveau

Les **phases** d'un **SCN** hiérarchique suivent un cycle dans lequel on exécute successivement la **mise à jour** DSL locale, l'agrégation ascendante (voir 6.5.3.2) et la potentielle division descendante (selon la logique top-down). Le **recuit** peut être inséré durant la phase de mise à jour : on ajoute alors un terme $\sigma(t) \xi_{i,j}(t)$ si l'on souhaite relancer la dynamique en cas de stagnation. De même, l'on peut recourir à un **terme d'inhibition** si on estime qu'un nœud ou qu'un bloc agrège trop de liaisons, en réduisant spécifiquement la somme de ses pondérations $\sum_j \omega_{i,j}$. Dans les faits, il arrive qu'on conditionne l'activation du bruit ou de l'inhibition à la détection d'un faible gradient $\|\Delta \omega(t)\|$ signalant une convergence prématuée. On peut alors, pour débloquer la situation, élever temporairement $\sigma(t)$ (pour le recuit) ou γ (pour l'inhibition). La structure hiérarchique profite ainsi d'un “shake-up” contrôlé qui modifie l’agencement local ou macro, libérant le réseau d'un éventuel état sous-optimal. L'**équilibre** final suppose de faire décroître la température $\sigma(t)$ et de stabiliser la force inhibitrice, de sorte que le SCN se cristallise dans une configuration plus stable. Cette combinatoire d'agrégation, de scission, de recuit et d'inhibition, orchestrée dans un **cycle multi-niveau**, constitue l'une des stratégies les plus robustes pour organiser un SCN complexe, lui permettant de traverser les minima locaux, de réajuster ses blocs hiérarchiques et de maintenir une flexibilité face aux évolutions de la synergie ou aux aléas du réseau.

Exemple d'un Pseudo-Code.

Un modèle :

```
for iteration in 1..T:
    for each pair (i,j):
         $\omega(i,j) \leftarrow \omega(i,j) + \eta * [S(i,j) - \tau * \omega(i,j)]$ 
        +  $\sigma(t) * \text{random\_noise}(i,j)$  # recuit
        -  $\gamma_{\text{inhibit}}(i) * \text{sum}(\omega(i,*))$  # inhibition latérale
    if norm( $\Delta \omega$ ) < epsilon:
        raise  $\sigma(t)$  or  $\gamma_{\text{inhibit}}$  # or apply "pulse"
    # then do detection of clusters, do macro merges or splits, etc.
```

Cette structure assure qu'on ne s'enferme pas trop tôt dans des configurations rigides, et qu'on a la **possibilité** de se ré-agencer.

D. Avantages

Le principal avantage est de sortir d'une *configuration stable* localement en introduisant un aléa modéré (recuit) ou en contraignant la somme des liens (inhibition). On évite ainsi les “demi-mesures” : trop de liaisons intermédiaires qui ne mènent pas à des clusters nets, ou un “super-nœud” mal fondé dont on n’ose pas scinder.

Sur le plan **mathématique**, l'approche recuit s'inspire de techniques globales de minimisation d'énergie. En embrassant un peu plus d'exploration, on peut accéder à un **minimum** plus global, dépassant le minimum local où le DSL se serait bloqué. L'inhibition, quant à elle, impose une **sélection** plus franche des connexions, clarifiant la hiérarchie.

Les mécanismes d'inhibition et de recuit peuvent être utilisés ensemble. On peut, par exemple, lancer le recuit au début (température élevée) pour explorer, puis introduire l'inhibition latérale en cours de route afin de forcer la **sparsité**. Cette séquence s'accorde avec la logique **agrégation/division** : la fusion se fait plus aisément quand on a “secoué” le réseau, tandis que la division s'opère plus distinctement lorsque les liens sont “assainis” par l'inhibition.

Conclusion (6.5.4.3)

Les **mécanismes** d'inhibition et de recuit (détailés au Chap. 7) représentent des **leviers** précieux pour **éviter la stagnation** dans un algorithme multi-niveau DSL. Sans eux, le réseau peut facilement se figer dans une structure sous-optimale ou conserver un excès de liens moyens, rendant la hiérarchie peu claire. Avec :

437. **Inhibition**, on introduit une **concurrence** entre les liaisons, forçant un choix plus marqué : certains liens se renforcent, d'autres disparaissent, rehaussant la qualité des clusters.

438. **Recuit simulé**, on injecte un **bruit** qui “explore” de nouvelles configurations, augmentant la probabilité de franchir des barrières locales et d'atteindre une organisation plus globale.

Cette **intégration** complète harmonieusement le cycle agrégation–division, autorisant des remises en cause partielles de la structure lorsque nécessaire et procurant la **souplesse** requise pour s'adapter à un environnement évolutif ou à des données complexes.

6.6. Études de Cas et Illustrations

Les principes **multi-niveau** du **DSL** (Deep Synergy Learning) et les mécanismes d'**auto-organisation** qu'il propose (agrégation, division, synchronisation, etc.) trouvent des champs d'application variés et concrets. Dans cette section 6.6, nous passons en revue quelques **études de cas** et **domaines** illustrant l'impact potentiel de la synergie multi-échelle. Qu'il s'agisse de **vision multi-modal**, d'**analyse contextuelle** de la parole, de **robotique sensorimotrice**, d'**agents conversationnels** ou de **simulation d'événements**, le fil conducteur demeure la capacité du réseau à gérer les **données** (ou entités) à différents **paliers**, créant un équilibre entre détails locaux et structures globales.

6.6.1. Systèmes de Vision Multi-Modal

Les systèmes de **vision** actuels doivent non seulement reconnaître des images, mais souvent intégrer plusieurs **modalités** (ex. image + annotation textuelle, ou image + métadonnées audio). Dans un contexte **multi-modal**, le **DSL** propose de structurer l'information en **entités** et **liens synergiques**, afin de repérer des clusters pertinents à plusieurs échelles.

6.6.1.1. Micro-Clusters = Images Localement Semblables, Macro = Classes Sémantiques

Contexte et Logique Générale. Dans un système de **vision** à grande échelle, il est fréquent de manipuler un très grand nombre d'images, chacune éventuellement décomposée en fragments ou "patchs" (sous-parties de l'image). L'**objectif** est de parvenir à une structure hiérarchique : à une échelle locale (micro), on groupe les patches très semblables (textures, couleurs, formes), tandis qu'à une échelle plus globale (macro), on obtient des **catégories sémantiques** (types d'objets, classes d'usages). La **dynamique** DSL (Deep Synergy Learning), avec ses mécanismes de mise à jour des pondérations ω et son agrégation multi-niveau, offre un cadre souple pour cette organisation. Les développements ci-après illustrent la façon dont se forment des micro-clusters de "mini-images" puis, par agrégation, des classes plus abstraites.

A. Micro-Clustering Local : Groupement de Patches Visuels

Au niveau le plus **bas** (noté $\ell = 0$), on considère un ensemble massif de **mini-images** ou "patchs" extraits des grandes images. Chaque patch \mathcal{E}_i est relié à d'autres patches \mathcal{E}_j par des pondérations $\omega_{i,j}$ mesurant leur **similarité** visuelle locale. Cette similarité (score $S(i, j)$) peut reposer sur des comparaisons de **couleurs**, de **descripteurs** SIFT ou HOG, ou encore de **textures**.

La mise à jour DSL local applique la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta_0 [S_0(i, j) - \tau_0 \omega_{i,j}(t)].$$

Lorsque $S_0(i, j)$ est élevé (patchs très semblables), $\omega_{i,j}$ tend à croître, **favorisant** la formation de groupes de patches proches. Après plusieurs itérations, on détecte ainsi, via un critère de sur-seuil ou un algorithme de **connected components**, des **micro-clusters** de patches. Ces micro-clusters peuvent correspondre à des **motifs** récurrents (ciel bleu, feuillage vert, zones de textures identiques, etc.) ou à des éléments communs (fragments de visages, de lettres, etc.).

Le **rôle** de cette étape micro est d'extraire une **granularité** fine, en laissant la dynamique DSL créer des petits regroupements localement cohérents. Il n'est pas nécessaire d'avoir de label humain ou de connaissance externe : l'**auto-organisation** s'appuie uniquement sur le renforcement des liens $\omega_{i,j}$ fortement **synergiques** (patchs quasi identiques ou très proches).

B. Passage Macro : Classes Sémantiques

Une fois les micro-clusters formés, on se place au **niveau** $\ell = 1$, où chaque cluster local de patches se voit représenté par un **super-nœud** $\mathcal{N}_\alpha^{(1)}$. Les pondérations $\omega_{\alpha,\beta}^{(1)}$ entre ces super-nœuds se calculent via une fonction Ψ (moyenne, somme, maximum, etc.) appliquée aux liens $\{\omega_{i,j}^{(0)}\}$ entre les patches membres de α et β .

Sur ce graphe de **super-nœuds**, on applique de nouveau la démarche DSL ou un critère d'**homogénéité** pour regrouper en classes plus vastes : on peut former un macro-nœud $\mathcal{N}_p^{(2)}$ qui englobe plusieurs super-nœuds $\{\mathcal{N}_\alpha^{(1)}, \dots\}$. Ces macro-nœuds, ou **catégories** plus globales, peuvent correspondre à de véritables **concepts** sémantiques : animaux, véhicules, paysages, etc.

La **logique** d'agrégation hiérarchique permet donc, d'une part, de détecter localement les similitudes visuelles (couches micro) et, d'autre part, d'**unifier** ces similarités en classes d'objets pertinents (couche macro). Le DSL répercute la force des liens $\omega_{i,j}$ depuis le niveau patch (textures) jusqu'à la reconnaissance de **familles** d'images plus abstraites.

C. Retombées Pratiques et Architecture Finalisée

À l'**échelle micro**, on dispose d'informations extrêmement **locales** : par exemple, un patch de taille 8×8 correspond à une petite zone homogène. La **dynamique** DSL fait émerger des micro-clusters homogènes, utiles pour un pré-classement ou une indexation fine. Au **niveau macro**, l'agrégation délivre des **classes** globales qui reflètent des **catégories** ou des **objets**. Ainsi, dans une base de plusieurs millions d'images, on aboutit à une structure hiérarchique où chaque image se décompose en divers patches, lesquels appartiennent à des micro-clusters, et ces micro-clusters se rattachent à des catégories plus vastes.

Cette **organisation** hiérarchique facilite la **navigation** multi-résolution : on peut opérer une recherche par similitude locale (trouver tous les patches ressemblant à un certain motif) ou par **grande classe** (retrouver toutes les images contenant un chat, une voiture, un paysage de neige, etc.). Le **DSL** assure la liaison entre ces deux échelles, en propageant la **cohérence** visuelle locale vers des regroupements plus sémantiques.

Conclusion

L'exemple du **traitement d'images** illustre clairement l'apport d'une **démarche multi-niveau** en DSL :

- **Micro** : groupement de patches $\{\mathcal{E}_i\}$ fortement similaires (couleurs, textures, motifs locaux).
- **Macro** : agrégation de ces micro-clusters en **catégories** plus larges (objets, thèmes sémantiques).

Aucun label humain n'est nécessaire pour guider ce processus : la **synergie** visuelle joue le rôle de signal local (similarité), et la **dynamique** DSL (renforcement/décroissance) construit pas à pas des **clusters** stables, puis des **super-clusters**, etc. Ce mécanisme génère ainsi un **réseau** à plusieurs paliers (patchs → micro-clusters → super-catégories) reflétant la réalité à la fois **locale** (textures) et **globale** (concepts). Ce paradigme se généralise à d'autres domaines (audio, texte) où l'on veut unir des attributs "microscopiques" (ex. phonèmes, tokens) en entités plus macro (mots, phrases, documents), toujours via la logique d'**auto-organisation** du DSL.

6.6.1.2. Approche Fractale : Répétition de Motifs "Objets" à Différentes Granularités (ex. Sous-Catégories)

Introduction et Contexte Global. De nombreux systèmes de **vision multi-modal** illustrent la présence d'**objets** ou de **sous-objets** dont la **forme** ou la **structure** se retrouve à diverses **échelles**. Un objet complexe (tel qu'un bâtiment, un végétal) peut receler des éléments répétés (fenêtres, feuilles) présentant des caractéristiques quasi identiques, simplement redimensionnées ou agencées différemment. Cette **répétition** et **auto-similarité** évoque le **concept fractal** (cf. chap. 6.3 sur la fractalité). Du point de vue **Deep Synergy Learning (DSL)**, la possibilité de décrire un motif à l'échelle micro et de le retrouver dans une **catégorie** plus large (macro) s'appuie sur le principe d'**auto-organisation** hiérarchique. Les développements ci-après explorent en quoi cette logique fractale se matérialise et quels bénéfices elle apporte à la reconnaissance d'images (ou de sous-catégories d'objets) via le **DSL**.

A. Répétition de Motifs et Multi-Échelle dans les Images

La notion de **fractal** en vision se fonde sur une **invariance** (ou quasi-invariance) lorsqu'on change d'échelle. Les mêmes *motifs* réapparaissent, que l'on examine une zone microscopique ou la totalité de l'image. Dans un système DSL :

Motifs locaux répétés. Les images naturelles (bâtiments, véhicules, plantes) révèlent souvent des *sous-formes* identiques ou analogues, distribuées au sein d'un *grand* objet. Par exemple, on retrouve des fenêtres identiques au sein d'une façade, ou plusieurs roues identiques dans un convoi de wagons.

Au **niveau micro**, le DSL repère des similitudes entre patchs présentant la même texture ou la même géométrie. De proche en proche, il agrège ces micro-similarités. Le résultat est un **cluster** local reflétant un motif récurrent (la "forme" fenêtre, par exemple).

Sous-catégories issues d'un même patron. Lorsqu'on monte en échelle (du niveau patch micro à un niveau de classes plus larges), on peut voir apparaître des *sous-catégories* d'un même type. Dans la classification d'espèces végétales, des arbres partageant un *même* motif foliaire s'inscrivent dans une sous-catégorie spécialisée, mais dont la forme générale se répète (feuilles, nervures).

Le **DSL** hiérarchique consolide ainsi ces groupes de patchs (micro) en *sub-classes* (macro), propageant la cohérence depuis la structure locale jusqu'à une catégorie englobante. Cet enchaînement reproduit l'**auto-similarité** propre aux fractals : la *même organisation* (mêmes types de motifs) se manifeste à plusieurs niveaux de détail, simplement transposée ou mise à l'échelle différemment.

B. Notion de Fractalité dans la Vision

L'**auto-similarité** fractale s'observe dans de nombreuses images naturelles (motifs de nuages, feuillages, etc.). Dans un **DSL** qui autorise à la fois une mise à jour locale (renforcement de $\omega_{i,j}$) et une agrégation + division multi-niveau, on peut détecter :

Box-counting et lois de puissance. Lorsqu'on recense le *nombre* de groupes à une taille donnée ou l'intensité moyenne de similarité en fonction de l'échelle, on peut observer une loi de puissance. Si le graphe des $\omega_{i,j}$ fait apparaître une dimension fractale lors du *box-counting*, cela traduit une récurrence structurelle : la hiérarchie "ressemble" à elle-même à différentes échelles.

Exemple : sous-catégories. On peut considérer un exemple : à l'échelle macro, on identifie la catégorie "bâtiments". En descendant un niveau, on voit se dégager "bâtiments à fenêtres rectangulaires" vs. "bâtiments à arches rondes", etc. En descendant encore, chaque fenêtre ou arche se décompose en éléments plus petits, toujours suivant la même logique d'agencement. Le **DSL** met à jour ω de façon identique : un cluster trop hétéroclite se scinde, un micro-groupe très semblable se fusionne, et l'on retrouve la *même forme d'organisation* d'un niveau à l'autre, caractéristique d'une forme fractale.

C. Avantages d'une Approche Fractale en Vision

Robustesse à l'Échelle. Les systèmes de vision doivent gérer des variations d'échelle (un objet vu de près ou de loin) et de perspective. Dans une **structure fractale**, la répétition de motifs à différentes tailles autorise l'identification d'un *même type d'entité* qu'il soit grand ou petit dans l'image. Le **DSL** hiérarchique renforce cette capacité de reconnaître des patches similaires même s'ils sont "scalés" ou partiels, aboutissant à la **convergence** d'indices vers une même catégorie.

Multi-Résolution. Une scène peut être vue sous différentes résolutions : un objet lointain n'offre pas de détail (macro-niveau), puis un zoom fait apparaître des micro-structures. Le **DSL** fractalement organisé permet d'exploiter autant la vue "générale" (regroupement large) que la vue "fine" (micro-clusters), tout en assurant la continuité *auto-similaire* entre ces niveaux.

Découverte de Structures. Sans supervision externe, un réseau DSL auto-similaire découvre des **patterns récurrents** (fenêtres, roues, feuilles) au niveau micro, les agrège en "sous-ensembles d'objets" (ex. sous-catégories de voitures selon le type de roues), puis les regroupe dans des super-catégories (voitures, camions, chars) au niveau macro. Cette **imbrication** reproduit la "mise en abyme" fractale : chaque entité complexe se compose d'entités similaires plus petites, etc.

Conclusion

L'**approche fractale** en vision multi-modal fait ressortir la **répétition** de motifs à différentes **granularités**. Un **motif local** (patch d'image) peut se répliquer ou s'assembler en un *objet complet* (macro), qui lui-même se révèle être un

sous-ensemble d'une *catégorie plus large*. Le **DSL**, par sa logique hiérarchique et **auto-organisée**, exploite exactement cette **auto-similarité** pour construire, de bas en haut, des **sous-catégories** ou des regroupements d'objets récurrents. Ce fonctionnement épouse la **structure fractale** fréquemment constatée dans les images naturelles ou synthétiques, offrant :

- **Robustesse** aux changements d'échelle,
- **Multi-résolution** cohérente,
- **Découverte** non supervisée de sous-catégories issues d'un même patron.

Ce paradigme multi-niveau, enrichi par la répétition de *mêmes* motifs à diverses échelles, permet au système **d'apprendre** et de **segmenter** la réalité visuelle selon un prisme fractal, offrant ainsi une reconnaissance hiérarchique plus flexible et plus riche.

6.6.1.3. Gains en Robustesse : Le Système s'Adapte si de Nouvelles Images Arrivent (Chap. 9)

Introduction et Contexte. Dans un **système de vision** reposant sur l'**architecture multi-niveau** d'un **Deep Synergy Learning (DSL)**, il est fréquent d'ajouter de nouvelles images (ou de nouveaux flux multimédias) au fil du temps. La capacité d'intégrer ces données additionnelles sans redémarrer l'entraînement à zéro, ni déstabiliser la structure hiérarchique, constitue un atout majeur en matière de **robustesse**. Les développements qui suivent clarifient la manière dont, grâce à la logique DSL, l'arrivée d'images inédite se traduit par une **mise à jour** locale des pondérations, qui se propage aux niveaux supérieurs (macro) de manière continue et adaptative.

A. Réactivité des Liens Synergiques face à l'Arrivée de Nouvelles Images

Le **DSL** procède par mise à jour itérative des pondérations $\omega_{i,j}(t)$ à l'aide d'une formule du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ désigne un taux d'apprentissage, $\tau > 0$ un facteur de décroissance, et $S(\mathcal{E}_i, \mathcal{E}_j)$ la **synergie** (ou similarité) entre deux entités $\mathcal{E}_i, \mathcal{E}_j$. Lorsqu'une image inédite \mathcal{E}_{new} survient, on l'introduit comme un nouveau nœud dans le réseau, initialisant $\omega_{\text{new},j}$ à de petites valeurs (ou zéro) et évaluant $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ pour les entités $\{\mathcal{E}_j\}$ préexistantes. Les mises à jour successives renforcent ou réduisent $\omega_{\text{new},j}$ en fonction de la cohérence (ou non) entre \mathcal{E}_{new} et \mathcal{E}_j . Si la synergie est élevée, la nouvelle image consolide un lien fort et se rattache à un cluster existant ; si la synergie reste faible, elle demeure isolée ou forme un micro-cluster indépendant, jusqu'à ce que d'autres images similaires surviennent.

Après un laps de temps, les **informations** associées à \mathcal{E}_{new} (ses liens forts, ses liens quasi-nuls) remontent dans la structure multi-niveau : au palier suivant, le super-nœud correspondant à un cluster local inclut la nouvelle entité, et les pondérations $\omega_{\alpha,\beta}^{(\ell+1)}$ sont recalculées via la fonction Ψ . Ainsi, on met à jour les liaisons au **niveau** macro sans nécessiter une refonte complète.

B. Robustesse et Adaptation

Ce mécanisme permet au réseau de **continuer** son évolution en absorbant de la **nouvelle** information sans ré-apprendre globalement.

Dans des algorithmes classiques (par ex. entraînement supervisé avec un réseau de neurones convolutionnel), l'arrivée d'un nouveau lot d'images out-of-distribution impose souvent de **réentraîner** (ou au moins de fine-tuner) le modèle sur l'ensemble des données. Avec le **DSL**, on ne reconstruit pas l'architecture, on **insère** \mathcal{E}_{new} au sein de la matrice des pondérations, et on laisse les règles de synergie influer sur la consolidation ou la suppression de liens.

Si la nouvelle image se rapproche d'un cluster (ex. un groupe d'images de chats), elle fortifiera le micro-cluster correspondant. Dans certains cas, si l'image introduit une hétérogénéité inattendue (ex. un micro-cluster devient trop

vaste et disparate), on recourt à la **division** au niveau macro (section 6.5.2) : un bloc d'images se scinde pour refléter le nouvel équilibre. Le système se montre **flexible**, capable de reconfigurer partiellement la hiérarchie.

Lorsque la nouvelle image est anomalie ou bruit, ses liaisons demeurent faibles (faible $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$), donc $\omega_{\text{new},j}$ ne grandit pas. Elle peut être isolée ou former un cluster marginal avec d'autres images bruitées. Cette structure peu connectée ne perturbe pas la classification globale, ce qui renforce la **robustesse**.

C. Dimensions Mathématiques de l'Adaptation

Dans un réseau déjà stabilisé, l'ajout d'une nouvelle image peut rester sans effet si aucune pondération $\omega_{\text{new},j}$ ne réussit à s'élever au-delà d'un seuil. Pour **assurer** qu'on réexamine les liens, on peut injecter un **bruit** contrôlé (recuit simulé, cf. chap. 7) ou un **terme** d'inhibition forçant un rééquilibrage local. Mathématiquement :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t) - \gamma \sum_k \omega_{i,k}(t).$$

Cette version plus complète mixe bruit ($\sigma(t) \xi$) et inhibition latérale ($\gamma \sum_k \omega_{i,k}$), qui aident à **sortir** d'un équilibre trop figé.

La logique DSL demeure : localement, $\omega_{i,j}$ s'ajuste, puis le **cluster** correspondant se met à jour, puis cette mise à jour touche les paliers supérieurs via la fonction Ψ . Lorsque suffisamment d'itérations se sont écoulées, la nouvelle image est **intégrée**, et si son arrivée a créé une incohérence dans un bloc, celui-ci s'est subdivisé ou a re-fusionné partiellement pour retomber dans un état stable.

D. Liens avec Chap. 9 (Évolutions Temps Réel)

Le **chap. 9** approfondit cette idée d'**apprentissage continu** ou **streaming** : on ne dispose pas d'un jeu de données fixe, mais d'un flux ininterrompu. Le **DSL**, en assurant un algorithme **auto-organisé** et **distribué**, traite cette arrivée progressive d'images, guidé par la notion de synergie $\{\omega_{i,j}\}$. La **robustesse** ainsi conférée se traduit par :

- Un **coût** de mise à jour local (pas de recalcul global).
- Une possibilité de **réorganisation** si nécessaire (division top-down, recuit pour sortir d'un plateau).
- Une **navigation hiérarchique** qui reste fonctionnelle au fur et à mesure que de nouveaux objets ou de nouvelles catégories émergent.

Conclusion

Lorsque de **nouvelles images** parviennent dans un **système de vision multi-modal** reposant sur la **logique DSL** (et agrégation/division multi-niveau), la **robustesse** observée s'explique par la **plasticité** inhérente au réseau. L'arrivée d'entités inédites s'intègre localement : on évalue la synergie $\{\omega_{\text{new},j}\}$, on met à jour, et on propage cette information vers les paliers supérieurs. Il n'est pas requis de "mettre à l'arrêt" ou de **réentraîner** l'ensemble : la structure s'ajuste progressivement, un atout essentiel en **temps réel** ou en flux **continu**. Les mécanismes d'**inhibition** et de **recuit** (chap. 7) renforcent encore la capacité du DSL à échapper aux configurations figées et à englober naturellement de nouvelles données, assurant une **évolution** organique de la hiérarchie, sans rupture ni réinitialisation globale.

6.6.2. Analyse Contextuelle de la Parole et du Langage

Les principes du **DSL** (Deep Synergy Learning), initialement illustrés en vision multi-modal (6.6.1), s'appliquent tout aussi bien à l'**analyse** de la parole ou du texte. Le langage humain se compose en effet de **niveaux multiples** — des sons élémentaires (phonèmes) jusqu'aux concepts abstraits — offrant ainsi un cadre naturel pour la synergie multi-échelle.

6.6.2.1. Micro : phonèmes ou chunks de phrase, macro : concepts sémantiques

Au plus bas niveau (micro), la parole ou le texte se décompose en **unités élémentaires** : les **phonèmes** (pour l'oral) ou les **morphèmes** / "chunks" (pour l'écrit). Par exemple, un flux sonore est découpé en courtes séquences (ex. 20–50 millisecondes) dont on extrait des **features** (MFCC, spectrogrammes locaux...), alors qu'un texte peut se scinder en tokens ou petits segments (une partie d'un mot, un groupement de mots, etc.).

Le **DSL**, à ce niveau, gère un ensemble d'entités $\{\mathcal{E}_i\}$ représentant ces briques linguistiques. Les pondérations $\omega_{i,j}$ quantifient la **synergie** ou la similarité entre fragments. Un $\omega_{i,j}$ élevé signifie que deux phonèmes (ou deux tokens) **coïncident** souvent ou s'articulent fréquemment dans un même contexte local.

Lorsque ces fragments s'**agrègent** (voir les logiques bottom-up de 6.5), ils forment des unités plus larges : **mot**s, **groupes de mots**, puis **concept**s sémantiques (familles de termes). Le **DSL** peut réévaluer la synergie $\omega_{\alpha,\beta}$ à ce palier, regroupant des **notion**s proches (ex. "ordinateur" / "programme" / "machine").

Mathématiquement, on définit une fonction d'agrégation Ψ qui combine les pondérations $\omega_{i,j}^{(0)}$ en pondérations $\omega_{\alpha,\beta}^{(1)}$ au niveau conceptuel. On obtient alors un graphe sémantique plus abstrait, où les noeuds sont des **concept**s détectés (macro-niveau).

Dans un **DSL**, la distinction micro-macro s'opère **dynamiquement**, sans imposer un modèle figé comme un "parse tree" unique. On peut former plusieurs **clusters** de tokens/phonèmes selon le contexte. L'architecture multi-niveau (chap. 6.2) laisse place à une organisation hiérarchique plus flexible que le schéma strict "mot → syntagme → phrase".

Cette **souplesse** permet au système de repérer de nouveaux concepts, d'en fusionner certains, ou d'en séparer d'autres, en fonction de la synergie calculée lors de l'analyse sémantique ou phonétique.

A. Exemples de Mise en Pratique

Reconnaissance de la Parole (ASR)

Au niveau **micro**, des segments audio (frames) sont reliés en fonction de leurs similarités spectrales, formant des **clusters** phonétiques. Puis, au niveau plus haut, on agrège ces phonèmes en **mot**s. Enfin, on peut détecter des **concept**s / intentions (macro).

Le **DSL** facilite la prise en compte simultanée de différents signaux acoustiques (voix), et de la synergie entre phonèmes courants pour **valider** ou **corriger** un mot ambigu.

Analyse Sémantique de Texte

Au niveau **micro**, un texte se segmente en tokens (mots ou sous-mots). Les liens $\omega_{i,j}$ se renforcent si deux tokens apparaissent souvent ensemble ou dans des contextes semblables. Des **clusters** de mots synonymes ou fortement associés émergent localement.

Au niveau **macro**, on obtient des **concept**s (thèmes) qui regroupent tout un champ lexical : par exemple, un cluster "sport" vs. un cluster "finance". Cet ensemble de concepts forme alors un graphe sémantique plus global.

B. Intérêt Mathématique et Pratique

Mathématiquement, la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ peut se définir comme une **similarité** sémantique (ex. co-occurrence, vecteurs embedding) ou une **compatibilité** phonétique. Le DSL actualise les pondérations $\omega_{i,j}$ par la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta[S(i,j)\tau \omega_{i,j}(t)].$$

Plus les entités (tokens, phonèmes) apparaissent conjointement, plus le lien se renforce. Au contraire, si elles se dissocient, $\omega_{i,j}$ tend vers 0 (ou un niveau faible).

Au fur et à mesure qu'on monte en échelle, on combine $\{\omega_{i,j}^{(k)}\}$ via une fonction Ψ . Les super-nœuds (macro) reflètent des concepts plus abstraits.

Cette hiérarchie dynamique est **robuste** : si un nouveau token apparaît souvent, il trouvera sa place dans un cluster ou en fondera un (6.5.4.2 sur les paramètres). Il ne requiert pas de **réapprentissage** global.

Applications

Compréhension du langage : on peut imaginer un DSL gérant la **cohérence contextuelle** (ex. “ce segment phonétique se connecte fortement à tel concept s'il apparaît dans un champ lexical précis”).

Correction automatique : si un token se retrouve isolé (faible synergie), on peut le **réassigner** à un cluster sémantique proche, jouant un rôle analogue aux algorithmes de correction contextuelle.

Conclusion

Dans l'**analyse contextuelle** de la parole et du langage, on distingue :

- Un **niveau micro** (phonèmes, chunks de phrase, tokens) où la **synergie** se bâtit localement entre entités proches,
- Un **niveau macro** (concepts, champs sémantiques, intentions) englobant un ensemble cohérent de mots/expressions.

Le **DSL** offre la possibilité de **relier** ces deux plans via des **liens** $\omega_{i,j}$ adaptatifs, permettant à la structure linguistique de se **former** (agrégation) ou de se **subdiviser** (division) selon la cohésion sémantique. Résultat : un **réseau** linguistique hiérarchisé, prêt à interpréter des **segments** (micro) tout en reconnaissant des **concept**s (macro), et assez flexible pour accueillir de nouveaux tokens ou variations contextuelles sans reconstruction globale.

6.6.2.2. Multi-Échelle : Phrase → Paragraphe → Document, Fractalité Possible si la Structure se Répète (Thèmes, Sous-Thèmes)

Introduction et Contexte Général. Le **langage** (oral ou écrit) peut être décrit selon plusieurs **paliers** : le palier **micro** (mots, tokens) s'assemble en **phrases**, lesquelles se regroupent en **paragraphes** (un sous-thème), puis en **document** complet (un thème majeur). Cette organisation hiérarchique se retrouve dans de nombreux textes (articles, essais, romans). Dans un **Deep Synergy Learning (DSL)** multi-niveau, la logique de **synergie** (pondérations $\omega_{i,j}$) et de **dynamique** (agrégation, division) permet de construire et d'adapter cette structure, tout en révélant des **patterns** potentiellement fractals lorsque le *même* thème (ou sous-thème) apparaît à différentes échelles.

A. Paliers du Texte : De la Phrase au Document

Niveau Micro : la Phrase.

Le premier palier (micro) regroupe les **mots** (tokens, segments). On définit des pondérations $\omega_{i,j}$ évaluant la similarité ou la synergie entre deux mots (par exemple, affinité sémantique, co-occurrence, relations syntaxiques). Après application de la mise à jour DSL, on peut détecter des *clusters* de mots qui forment un **énoncé** cohérent. Sur le plan mathématique, cette phrase $\mathcal{C}_\alpha^{(0)}$ se caractérise par une somme de pondérations $\sum_{i,j \in \mathcal{C}_\alpha} \omega_{i,j}$ relativement importante, traduisant la focalisation contextuelle.

Niveau Mésoscopique : le Paragraphe.

Le paragraphe regroupe plusieurs **phrases** reliées par un sous-thème commun. Dans un **DSL** multi-niveau, chaque phrase \mathcal{C}_α devient un super-nœud $\mathcal{N}_\alpha^{(1)}$. Les liens $\omega_{\alpha,\beta}^{(1)}$ entre phrases se calculent en agrégeant les poids $\{\omega_{i,j}^{(0)}\}$. Un paragraphe cohésif présente donc un bloc de phrases fortement interconnectées au niveau 1, selon la fonction Ψ (moyenne, somme, etc.). Un paragraphe plus hétérogène peut inciter une **division** (top-down) si la cohésion est jugée insuffisante.

Niveau Macro : le Document.

Le document entier (ou un chapitre d'un livre) s'obtient en **fusionnant** les paragraphes lorsqu'ils partagent un thème majeur. La **dynamique** DSL continue de s'appliquer : si un grand bloc (document) se révèle hétéroclite, on peut le scinder en chapitres ; s'il est déjà homogène, on le considère comme un macro-nœud stable. En outre, on peut aller plus loin en reliant différents documents entre eux, si l'on souhaite construire un **réseau** inter-articles (un corpus).

B. Fractalité et Répétition de Thèmes

Dans un *texte* ou *discours* structuré, un **même** concept (thème principal) se décline souvent en sous-thèmes, puis en sous-sous-thèmes, à différents paliers (chapitre, section, paragraphe, phrase). Cet agencement **auto-similaire** peut être vu comme fractal ; on retrouve le *même* motif d'organisation à des granularités distinctes, par exemple la reprise d'un motif argumentatif ou la récurrence d'un champ lexical sous des variations minimales.

Les textes (et leurs distributions lexicales) montrent parfois des **lois** de puissance. La **distribution** en fréquences des mots (loi de Zipf), la répartition des occurrences de thème, etc. Un **DSL** qui met en place un cycle d'agrégation–division (voir 6.5.3) peut exploiter cette **récurrence** en scindant un bloc dès que l'hétérogénéité interne dépasse un seuil, et en fusionnant localement des entités très similaires. Ainsi, la structure qui émerge est **autosimilaire** : on retrouve la *même* loi de regroupement (ou de scission) d'un paragraphe à l'autre, d'un chapitre à l'autre.

Si l'on trace les super-nœuds (chapitres, sections, paragraphes, phrases) dans un graphe DSL multi-niveau, on voit se dessiner un **arbre** (ou un "graphe fractal") révélant l'invariance d'échelle. Un concept macroscopique (ex. "analyse fractale du texte") se reflète en multiples sous-sections, qui elles-mêmes se divisent en paragraphes, etc., chacun reprenant le même *patron* argumentatif.

C. Synergie Multi-Échelle et Valeur Pratique

Grâce à cette **structure** hiérarchique, on navigue dans le texte depuis un **niveau** macro (thème principal) jusqu'à un **niveau** micro (détails dans une phrase). Sur le plan mathématique, on utilise $\omega_{\alpha,\beta}^{(k)}$ pour passer d'un paragraphe α à un autre paragraphe β si leur synergie est forte, ou pour descendre d'un paragraphe α vers ses **phrases** internes. Cette multi-résolution rend l'exploration plus souple.

Dans un contexte évolutif (un document collaboratif, un flux textuel), l'ajout d'un **nouveau paragraphe** ou d'une phrase modifiée se répercute localement (mise à jour des liaisons ω), pouvant mener à la **fusion** avec un sous-thème existant ou la scission d'un paragraphe devenu hétéroclite. Ce fonctionnement limite la nécessité de reconstruire la totalité de la structure, assurant une **adaptation** progressive.

Chaque *phrase* $\mathcal{C}_\alpha^{(0)}$ est un cluster de mots, agrégé en un super-nœud $\mathcal{N}_\alpha^{(1)}$ pour le *paragraphe*. Les paragraphes $\mathcal{N}_\beta^{(1)}$ se regroupent en $\mathcal{N}_\beta^{(2)}$ (chapitre), etc. À chaque niveau, la *même* mise à jour DSL ou la *même* condition d'homogénéité s'exerce, de sorte que l'on obtient, si les thèmes se répètent, une forme de *pattern fractal* (la distribution des liens ω suit une structure similaire d'un niveau à l'autre).

Conclusion

Dans l'**analyse** du **langage**, on retrouve l'idée d'**organisation multi-niveau** : un **texte** se découpe en phrases, paragraphes, sections, documents. Le **DSL** multi-niveau prend en charge ces paliers en :

- **Agrégeant** localement (phrases→paragraphes) lorsque la synergie interne est élevée,
- **Scindant** un bloc (paragraphe, chapitre) si l'hétérogénéité \mathcal{H} devient trop forte,
- **Conservant** la continuité d'une *même* logique (renforcement, division) d'une échelle à l'autre.

Cette approche peut faire émerger une **structure fractale** si le *même thème* (ou la même forme argumentative) se répète à plusieurs échelles. On y trouve alors :

- **Auto-similarité** dans la répartition sémantique (lois de puissance, récurrences lexicales),

- **Réorganisation** adaptative à l'arrivée de nouveaux segments textuels,
- **Navigation** aisée du niveau macro (concept général) vers le niveau micro (phrases, mots).

Ainsi, le **DSL** s'avère un cadre fécond pour modéliser la **fractalité** potentielle des textes (thèmes, sous-thèmes répétitifs), tout en fournissant une base algorithmique permettant au réseau de **réagir** à des modifications ou enrichissements du document, sans bloquer ni devoir réapprendre dans son intégralité.

6.6.2.3. Rétroaction Top-Down si un Sujet Global est Identifié, Influant l'Interprétation Locale

Contexte et Objectif. Dans un **DSL** (Deep Synergy Learning) multi-niveau appliquée à l'**analyse du langage**, l'organisation des entités (mots, fragments de phrases, etc.) s'effectue d'abord au **niveau micro** (groupements locaux ou clusters de tokens), avant que ces ensembles ne soient agrégés pour former des **sous-thèmes** ou des **paragraphes** (niveaux intermédiaires) et, enfin, parvenir à un **sujet global** (niveau macro). L'une des forces du modèle DSL est de permettre une **rétroaction descendante** (top-down) : une fois qu'un **sujet majeur** s'est dégagé, le palier macro peut envoyer un *feedback* qui réoriente ou désambiguise certains liens locaux. Cette section explique comment, dès que le macro-nœud identifie un thème principal, il peut influencer l'**interprétation** de mots ambigus, de sous-phrases, ou de segments micro, assurant ainsi une **cohérence** multi-niveau.

A. Identification d'un Sujet Global et Flux Descendant

Au fur et à mesure que le **DSL** agrège les liens $\omega_{i,j}$ entre entités (mots, expressions, etc.), on peut parvenir, par un processus d'**agrégation** (bottom-up), à un macro-nœud représentant un **sujet majeur**, tel qu'« économie », « biologie », « intelligence artificielle », ou tout autre concept principal. Cette identification s'effectue souvent via un **cluster** de grande taille, rendu cohérent par la somme ou la moyenne de ses liens internes ($\omega_{\alpha,\beta}$).

Une fois ce thème principal \mathcal{T} établi, un **flux** top-down est possible. Sur le plan **formel**, on introduit un terme $\Delta_{\text{down}}^{(\text{macro})}(i,j)$ dans la mise à jour des pondérations :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] + \Delta_{\text{down}}^{(\text{macro})}(i,j).$$

Ce terme $\Delta_{\text{down}}^{(\text{macro})}(i,j)$ peut être **positif** ou **négatif** selon que le macro-nœud \mathcal{T} estime utile de renforcer la liaison (i,j) (car il est jugé pertinent pour le sujet) ou de l'inhiber (car il est hors sujet).

B. Influence sur l'Interprétation Locale (Mots Ambigus, Chunks Polyvalents)

En **linguistique**, de nombreux mots (ex. « bank », « seal », « network ») présentent plusieurs sens possibles. Le choix d'une **interprétation** dépend fortement du **contexte**. Si le macro-nœud a identifié le sujet global $\mathcal{T} = \text{« finance »}$, alors un mot comme « bank » est lu dans son acception financière plutôt que géographique (rive d'un fleuve). Mathématiquement, le flux $\Delta_{\text{down}}^{(\text{macro})}$ agit sur les liens $\omega_{\text{bank},\text{loan}}$ (renforcement) vs. $\omega_{\text{bank},\text{river}}$ (diminution).

De la même manière, si le macro-niveau indique un **champ lexical** (ex. « biologie »), des tokens ambigus au niveau micro seront naturellement attirés vers des clusters locaux alignés sur ce thème, et leurs liens avec d'autres sphères seront réduits. Cela **stabilise** la structure micro autour d'une cohérence partagée : le mot « cellule » se connectera plus fortement à « ADN », « protéine » qu'à « prison » ou « téléphone », si le macro-nœud « biologie » est reconnu.

Ce flux descendant n'annule pas la logique micro : si un mot ou une expression n'a réellement **aucun** rapport avec le sujet macro, le score de synergie S restera faible, et la rétroaction top-down ne suffira pas à forcer un lien artificiel. Un **équilibre** se crée par la confrontation du flux ascendant (représentations locales factuelles) et du flux descendant (contexte global), garantissant une **interprétation plus riche** et moins ambiguë.

C. Ampleur du Feedback et Dynamique d'Équilibrage

Mécanisme de contrôle.

Pour éviter des oscillations extrêmes, on peut régler l'amplitude de la rétroaction $\Delta_{\text{down}}^{(\text{macro})}$ par un paramètre $\gamma_{\text{topdown}} > 0$. On définit, par exemple, une fonction $\Phi(\mathcal{T}, i, j)$ qui indique la pertinence du lien (i, j) au vu du sujet \mathcal{T} , puis

$$\Delta_{\text{down}}^{(\text{macro})}(i, j) = \gamma_{\text{topdown}} \Phi(\mathcal{T}, i, j).$$

Si la valeur de Φ est positive, le lien (i, j) est encouragé ; si elle est négative, il est affaibli. Par cette modération, le DSL ne bascule pas trop rapidement.

Analogie cognitive.

Le fait qu'un locuteur ou un lecteur « comprenne » le thème d'un texte et revoie son interprétation des mots locaux renvoie à la **mise en contexte**. Au niveau micro, on s'appuie sur des associations directes (synergie sémantique brute) ; au niveau macro, on oriente ou on “désambiguise” ces associations pour coller au sujet principal.

Exemple pratique.

Si un article est globalement identifié comme un texte sur l'« IA » (intelligence artificielle), des mots comme « réseau », « neurone » ou « entraînement » seront alignés sur le champ “machine learning” plutôt que sur des sens moins pertinents (réseau social, neurone biologique, entraînement sportif). Les liens $\{\omega_{\text{neurone,entraînement}}\}$ se verront renforcés sous l'impulsion top-down, consolidant la cohérence IA.

Conclusion

La **rétroaction** top-down (macro → micro) joue un **rôle** fondamental dans un **DSL** multi-niveau appliquée au **langage** :

439. **Sujet global identifié.** Le niveau macro détecte un **thème** dominant,

440. **Signal descendant.** On “pousse” un feedback dans la mise à jour $\Delta_{\text{down}}^{(\text{macro})}$, modifiant $\omega_{i,j}$,

441. **Interprétation locale plus précise.** Les mots, expressions ou tokens ambigus sont réorientés vers le sens le plus compatible avec le thème identifié,

442. **Équilibre** de la dynamique. On ne force pas la structure locale contre la synergie intrinsèque, mais on l'influence pour une **cohérence** multi-niveau.

Ce mécanisme illustre l'un des principes du **DSL** : l'information circule **ascendante** (agrégation de liens locaux) et **descendante** (validation, scission ou repondération en fonction d'un sujet macro). La **compréhension** textuelle en bénéficie, car la **cohésion** globale rétroagit sur les choix de clustering micro, réduisant l'ambiguïté sémantique et consolidant les **relations** lexicales en un réseau linguistique stable et mieux aligné sur le **contexte**.

6.6.3. Robotique Synergique : Intégration Sensorimotrice

Dans le domaine de la **robotique**, les principes d'**auto-organisation** et de **synergie** (au cœur du **DSL**) s'appliquent tout particulièrement à la gestion **multi-niveau** des informations sensorielles et des actions motrices. Le robot, équipé de multiples **capteurs** (caméras, gyroscopes, LIDAR, capteurs de force, etc.), doit non seulement **fusionner** ces données pour percevoir son environnement, mais également **coordonner** diverses actions (rouler, saisir, se déplacer en essaim avec d'autres robots...) pour mener à bien un **comportement global** cohérent.

6.6.3.1. Micro-Niveau : Capteurs Basiques (Caméras, Gyros...), Macro : Comportement Global (Navigation, Prise d'Objets...)

Principes Généraux et Contexte. Les systèmes robotiques reposent souvent sur une **pluralité** de capteurs : caméras (vision), gyroscopes (orientation), LIDAR, capteurs de force, etc. Pour accomplir des **tâches** plus globales (navigation, interaction, manipulation d'objets), le robot doit intégrer ces multiples flux sensoriels. Dans une **approche de Deep**

Synergy Learning (DSL) multi-niveau, l'articulation est la suivante : au niveau **micro**, on s'occupe de la synergie entre capteurs eux-mêmes, tandis qu'au niveau **macro**, on obtient des comportements globaux. Les développements ci-après montrent comment, grâce à la mise à jour adaptative des pondérations $\omega_{i,j}$, le DSL construit des **clusters** sensoriels cohérents et, par agrégation, aboutit à des modules comportementaux plus larges.

A. Niveau Micro : Le Monde des Capteurs

Au niveau le plus bas (ou $\ell = 0$), on considère chaque capteur C_i comme un **nœud** dans un **SCN**. Il produit un flux de mesures (images, angles, distances) associé à une **entité** \mathcal{E}_i . On relie les paires $\{\mathcal{E}_i, \mathcal{E}_j\}$ par des pondérations $\omega_{i,j}$, qui traduisent la **synergie** ou la **corrélation** entre deux flux sensoriels. Ce mécanisme s'inspire de la mise à jour DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Si deux capteurs (par exemple, une caméra frontale et un LIDAR) “voient” la même forme d’obstacle, leur **similarité** $S(\mathcal{E}_i, \mathcal{E}_j)$ sera forte, entraînant une hausse de $\omega_{i,j}$. Inversement, des capteurs faiblement corrélés (vision frontale et capteur de force sur la pince) n’auront pas de liaison importante. La **dynamique** DSL, en renforçant ou en diminuant les pondérations, fait émerger naturellement des **clusters** de capteurs coopératifs (ex. cluster “détection d’obstacle” regroupant les capteurs directionnels, cluster “calibration d’équilibre” regroupant gyroscopes + accéléromètres).

On peut écrire la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ de multiples façons : indicateur de corrélation statistique si deux capteurs produisent des mesures parallèles, indice de concordance si la vision confirme les distances du LIDAR, etc. Le **DSL** consolide ces indices pour obtenir, au niveau micro, des sous-ensembles de capteurs spécialisés.

B. Niveau Macro : Le Comportement Global

Au palier supérieur, on **agrège** (via une fonction Ψ) les liaisons $\omega_{i,j}$ internes à chaque cluster sensoriel afin d’obtenir des super-nœuds décrivant des **modules** plus larges. Le robot peut disposer d’un super-nœud “évitement d’obstacle” englobant les capteurs frontaux cohérents, d’un super-nœud “navigation globale” englobant GPS + boussole + gyroscopes, etc. Chacun de ces super-nœuds fournit une **fonctionnalité** ou un **sous-comportement**.

La **tâche** globale (macro) du robot, telle qu’aller d’un point A à un point B en évitant les obstacles, se décompose donc en une **composition** de super-nœuds sensoriels et d’algorithmes de contrôle : le DSL, en reliant ces super-nœuds, détermine un **module** “navigation” intégrant la synergie sensorielle requise (vision + LIDAR + odométrie). Pour la **manipulation** (prise d’objet), un autre macro-module intègre caméras de détection d’objets, capteurs de force sur la pince, etc. Sur le plan **formel**, ces super-nœuds peuvent être traités comme des **nœuds** $\{\mathcal{N}_\alpha^{(k)}\}$ au palier k , tandis qu’un **macro-nœud** unique, fusionnant ces sous-comportements, émerge au niveau ultime.

C. Exemples Concrets

Robot mobile.

Un robot à roue peut comporter :

- un cluster sensoriel “orientation” (gyroscope + accéléromètre) pour stabiliser la trajectoire,
- un cluster “vision frontale + LIDAR” pour évaluer les obstacles,
- un cluster “GPS + boussole” pour la localisation générale.

Ces ensembles, identifiés localement par la logique DSL, se combinent au niveau macro pour orchestrer la **navigation** (choisir la route, freiner si un obstacle est trop proche, etc.).

Bras manipulateur.

Dans une tâche de **saisie d’objets**, on identifie un cluster “capteur force + caméra pince” pour le contrôle de la préhension, éventuellement un cluster “vision externe + localisation d’objet” pour pointer l’objet, et un cluster “contrôle bras + base” assurant la posture globale. La hiérarchie de super-nœuds (micro→macro) donne une vue

modulaire de la robotique, où chaque sous-module sensoriel s'intègre dans un comportement de plus haut niveau (saisir, déplacer, poser).

D. Avantages Mathématiques et Opérationnels

Plutôt que de traiter chaque capteur individuellement dans un *unique* bloc de décision central, le DSL multi-niveau crée des **clusters** sensoriels naturellement cohérents. D'un point de vue mathématique, on aboutit à un **réseau** $\{\omega_{i,j}\}$ plus épars, où seuls les liens justifiés par la **synergie** subsistent. Cela allège la complexité, en substituant un trop-plein de connexions par une structure plus sélective.

Si un capteur se dégrade (ex. caméra obstruée), son lien ω avec les autres capteurs chute (faible S), incitant le système à s'appuyer sur d'autres sources. Au niveau macro, le **comportement global** (ex. "évitement d'obstacles") s'ajuste en s'alimentant davantage d'un LIDAR intact ou d'une caméra latérale encore fiable.

Les clusters sensoriels se mettent à jour **localement** : chaque capteur compare ses mesures à celles d'un petit sous-ensemble de capteurs apparentés. L'agrégation en super-nœuds de plus haut niveau s'effectue de façon asynchrone et incrémentale, favorisant la **réactivité**. On n'a pas besoin de réapprendre *ex nihilo* ou de tout centraliser, ce qui est crucial pour un robot en mouvement.

Conclusion

Dans la **robotique synergique**, le **niveau micro** consiste à gérer la **myriade** de capteurs (vision, inertIELS, force, distance...), tandis que le **niveau macro** porte sur le **comportement global** du robot (navigation, prise d'objets, etc.). Le **DSL** relie ces deux paliers en définissant des pondérations $\omega_{i,j}$ qui traduisent la **coopération** sensorielle : à l'échelle micro, des capteurs forment des clusters sensoriels cohérents ; à l'échelle macro, ces clusters se combinent pour accomplir la tâche globale, depuis la simple détection d'obstacles jusqu'à la manipulation.

Cette **organisation** hiérarchique est **adaptative** : si les flux sensoriels évoluent (capteurs défectueux, changement d'environnement), la mise à jour locale des pondérations assure une **auto-organisation** continue, dont les répercussions se font sentir au niveau macro. On obtient ainsi un **réseau** où la **sélection** (agrégation ou division) de sous-systèmes se fait progressivement, sur la base de la **synergie** sensorielle, et où le **comportement** (macro-nœud) émerge de l'intégration structurée de clusters sensoriels (micro). C'est cette **plasticité** qui donne au robot la **robustesse** nécessaire pour évoluer dans un environnement complexe et en perpétuelle évolution.

6.6.3.2. Logiciel Multi-Threads où le DSL Gère la Synergie Localement, puis un "Super-Nœud" Pilote l'Action Globale

Contexte Général et Motivations. Dans la robotique à grande échelle, l'**architecture logicielle** fait souvent appel à un **multi-threading** (ou multi-processus léger) pour traiter en parallèle divers **capteurs** (caméras, gyroscopes, LIDAR, etc.). Chacun de ces fils d'exécution s'occupe d'un sous-ensemble de données ou d'une tâche partielle (comme le traitement d'images ou l'acquisition lidar), tandis qu'un **module** central, parfois appelé "super-nœud" ou "pilot", orchestre le **comportement global** du robot (navigation, saisie d'objet, collaboration, etc.). Au sein du **Deep Synergy Learning (DSL)**, l'idée est d'exploiter cette répartition logicielle : chaque thread met en œuvre localement la **règle** de mise à jour $\omega_{i,j}$, tandis que le super-nœud agrège l'information et oriente l'action finale. On obtient ainsi une **auto-organisation** distribuée sur les capteurs, couplée à une **coordination** hiérarchique pour la prise de décision.

A. Répartition Multi-Threads pour la Gestion Locale de la Synergie

Au **niveau micro**, on considère qu'un ensemble de capteurs $\{\mathcal{E}_i\}$ sont distribués dans différents fils d'exécution (threads). Chaque thread reçoit les flux de quelques capteurs (caméra frontale, caméra latérale, LIDAR, etc.) et traite localement la **synergie** $\omega_{i,j}$. Par exemple, une **mise à jour** DSL de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

s'exécute au sein d'un thread si \mathcal{E}_i et \mathcal{E}_j relèvent d'un même groupe de capteurs ou d'un domaine d'opération connexe. L'information $S(\mathcal{E}_i, \mathcal{E}_j)$ peut être un indice de **similarité** (corrélation d'images, cohérence de mesures lidar) ou de **complémentarité** (caméra + infrarouge).

Dans un système multi-threads, chaque fil d'exécution tourne à sa propre fréquence, mettant à jour certaines parties de la matrice ω . Pour assurer la **cohérence**, on recourt à des verrous (mutex) ou des algorithmes lock-free afin que $\omega_{i,j}$ ne soit pas altéré de façon incohérente. Chaque thread reçoit périodiquement les états des capteurs, calcule $\Delta\omega$ pour les liens pertinents, puis valide ces changements. Cela **distribue** la tâche de mise à jour DSL et rend l'auto-organisation plus **échelonnée** : des clusters sensoriels locaux émergent par simple renforcement/décroissance en temps réel.

Mathématiquement, la matrice ω est fragmentée en plusieurs *blocs* gérés par autant de threads : chaque bloc correspond aux liens (i, j) entre capteurs dont la synergie doit être évaluée conjointement. Par exemple, on peut coupler la caméra frontale et le LIDAR frontal dans un **bloc** "déttection d'obstacles", un autre bloc rassemblant les capteurs inertIELS (gyroscopes, accéléromètres) pour la stabilisation. Les **clusters** de capteurs se manifestent localement si $\omega_{i,j}$ se consolide, renforçant la collaboration de ces entités.

B. Super-Nœud pour le Pilotage Global

Une fois les synergies locales (micro) établies, la **tâche** du robot (macro) consiste à agir : se déplacer, saisir un objet, éviter un obstacle, etc. Dans un DSL multi-niveau, on conçoit un **super-nœud** (un thread maître ou un module) qui regroupe les **super-nœuds** issus de l'agrégation sensorielle. Autrement dit, le super-nœud perçoit la configuration globale des clusters (ex. cluster "vision-frontale-lidar", cluster "accéléromètre-gyroscope") et combine leurs informations pour **décider** de l'action. S'il s'avère que la synergie "éviter obstacle" est forte (danger imminent), le super-nœud enclenche la commande motrice de freinage ; si c'est le cluster "saisie d'objet" qui s'active, il dirige le bras vers l'objectif.

Périodiquement, chaque thread local transmet au super-nœud un *résumé* : $\omega_{\alpha,\beta}$ pour différents segments sensoriels, ou un **score** de fiabilité. Le super-nœud peut alors "fusionner" (via Ψ) les données et décider, par exemple, de scinder ou de réunir certains clusters si l'hétérogénéité interne le justifie (voir 6.5.2). Cela se formalise par :

$$\omega_{\alpha,\beta}^{(\text{macro})} = \Psi(\{\omega_{i,j}^{(\text{local})} : i \in \alpha, j \in \beta\}),$$

puis le super-nœud évalue l'opportunité de moduler (inhiber ou renforcer) des liens via un flux descendant (chap. 6.4).

Dans la pratique, un **robot mobile** gère la synergie locale d'"obstacle detection" par un thread "lidar+caméra frontale" ; un autre thread combine inertIELS + GPS pour la navigation large. Le super-nœud global reçoit leurs états, voit s'il doit prioriser l'évitement d'obstacles ou continuer la route, et envoie des **commandes** au système moteur. En cas de conflit (capteur en panne), il peut forcer une reconfiguration en poussant un *terme descendant* défavorable aux liaisons du capteur défectueux.

C. Avantages en Robotique Synergique

Une approche **multi-threads** répartit la gestion de la synergie $\omega_{i,j}$; chaque fil d'exécution prend en charge un sous-ensemble de capteurs, allégeant la complexité globale. Le super-nœud ne fait qu'**agréger** et **décider**, évitant de gérer $O(n^2)$ liaisons en un seul point central.

Un événement local (caméra détectant un obstacle) modifie $\omega_{\text{cam},\text{lidar}}$ presque instantanément. Le super-nœud, en lisant ces informations, enclenche l'action appropriée. Cette asynchronie évite d'attendre un cycle complet d'un pipeline monolithique.

La défaillance d'un thread (par ex. une caméra latérale obstruée) n'immobilise pas tout le système. Les autres threads continuent leur mise à jour DSL, et le super-nœud réduit la confiance (ou la synergie) associée au module défaillant. Au besoin, on peut **ajouter** un capteur (thread nouveau) sans reprogrammer la totalité : ses liens ω émergent progressivement dans la logique DSL, et le super-nœud l'intègre si un cluster se forme autour de lui.

Conclusion

Dans une **architecture logicielle** robotique **multi-threads**, le **DSL** opère localement (micro) pour gérer la **synergie** entre capteurs distribués, chaque fil d'exécution maintenant ses pondérations ω selon la formule DSL. Ensuite, un **super-nœud** (thread principal) recueille la **configuration** globale et pilote l'**action** macro (navigation, saisie, interaction). Cette combinaison de :

- **Auto-organisation** distribuée (chaque thread capteur gère ω localement),
- **Agrégation hiérarchique** (super-nœud qui fusionne et oriente la décision),

permet une **répartition** de la charge (scalabilité), une **réactivité** élevée (changement local immédiatement pris en compte), et une **robustesse** (grâce à l'adaptabilité du DSL aux pannes ou ajouts de capteurs). L'ensemble s'inscrit dans la **philosophie** de la robotique synergique : intégrer de multiples flux sensoriels et modules de traitement en un **réseau** cohérent, réagissant en temps réel au monde extérieur.

6.6.3.3. Notion Fractale si on Observe la Même Logique de Commande à Divers Sous-Systèmes Moteurs

L'architecture **DSL** (Deep Synergy Learning) appliquée à la robotique ne se borne pas au traitement des capteurs ; elle peut également s'étendre aux **ensembles d'actionneurs** et de commandes motrices. À l'échelle la plus locale, des groupements de joints ou de moteurs se coordonnent pour des mouvements élémentaires (par exemple, les doigts d'une pince ou les roues antérieures d'un véhicule). À un niveau intermédiaire, plusieurs de ces sous-ensembles coopèrent dans un module plus vaste (un bras complet, un chariot de locomotion), tandis qu'au niveau le plus global (macro-nœud), le robot exécute un **comportement** ou une **tâche** englobante. Lorsque la même logique DSL (mise à jour des pondérations ω , agrégation, division, rétroaction descendante, etc.) est reproduite à chaque échelle d'organisation, il se crée une forme de **structure fractale**, car la même règle d'auto-organisation se réplique du plus petit sous-système moteur à l'unité robotique intégrale.

Il est d'abord essentiel de comprendre le rôle des **sous-systèmes moteurs** à l'échelle locale. Chaque actionneur ou groupe de joints est traité comme une entité \mathcal{E}_i . Les pondérations $\omega_{i,j}$ traduisent la synergie ou la synchronisation entre deux actionneurs i et j . La règle DSL

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

peut conduire à l'émergence de **clusters** moteurs dès lors que plusieurs joints se trouvent souvent co-activés ou dont les positions se coordonnent. Les doigts d'une pince se coordonnent dans la "manipulation fine", les roues droites d'un châssis forment un mini-groupe fonctionnel pour déplacer le côté droit, etc. La mise à jour adaptative des pondérations ω autorise ces regroupements à évoluer si le robot entreprend des tâches différentes (de la locomotion à la saisie d'objets), ce qui modifie les fréquences et les patrons de co-activation.

Une fois ces **clusters** moteurs identifiés au niveau micro, un **palier intermédiaire** agrège leurs synergies, créant des super-nœuds pour des modules plus vastes : un "bras gauche" ou une "base roulante" où se concentrent les synergies internes. Le mécanisme est le même qu'avec les capteurs, simplement transposé au monde des actionneurs. On applique une fonction Ψ agrégeant $\{\omega_{i,j}\}$ sur un sous-système donné, et l'on obtient $\omega_{\alpha,\beta}$ entre super-nœuds α et β . À ce stade, la *même* équation DSL ou la même division top-down peut se produire : si un sous-système devient trop "hétérogène" (par exemple, un grand regroupement d'actionneurs qui n'ont pas tant besoin de coopérer), on scinde, ou si deux sous-systèmes collaborent souvent, leurs liens ω se voient renforcés pour fusionner.

Le **palier macro** couronne le processus, lorsque le robot exécute un **comportement intégral**. Le super-nœud global, englobant l'ensemble des sous-nœuds moteurs, établit quelles combinaisons d'ensembles d'actionneurs déclencher pour chaque tâche. Lorsque le robot alterne entre locomotion et manipulation, il active ou module les liens $\omega_{\alpha,\beta}$ entre la base roulante et le bras, reflétant la co-activation ou la "pertinence conjointe" de ces modules. La même dynamique DSL assure qu'un comportement très souvent sollicité se traduise par des pondérations ω plus élevées entre les sous-systèmes pertinents, tandis qu'un comportement moins usuel demeure moins intégré ou subit la décroissance paramétrée par τ .

La notion **fractale** jaillit du fait que l'on retrouve, à chaque échelle, la *même* logique : le niveau micro (quelques joints) applique la règle DSL pour gérer les liens $\omega_{i,j}$, le niveau intermédiaire (un sous-système comme un bras complet) applique la même équation pour agréger ou diviser les sous-modules, et le niveau macro (tout le robot) applique encore cette formule pour orchestrer les super-nœuds. En pratique, on peut voir des distributions de pondérations ω qui présentent une **auto-similarité** lorsqu'on change d'échelle, ou constater que la fonction de clusterisation exhibe une loi de puissance quant à la taille des modules moteurs. Cela relève d'une **structure fractale**, car la *même forme* d'auto-organisation DSL se reproduit du plus petit sous-système à la commande intégrale de l'agent.

Un bénéfice évident se situe dans la **flexibilité**. À l'échelle micro, on laisse la main à un cluster local pour, par exemple, gérer la coordination fine de quelques doigts. Lorsque le robot doit synchroniser deux bras pour soulever un objet, un palier intermédiaire perçoit la synergie (co-activation récurrente) et forme un super-nœud “opération bimanipulative”, qui, au niveau macro, se connecte éventuellement à la base roulante si le robot doit transporter l'objet. Cette réplication de la *même règle* DSL sur plusieurs plans laisse la structure se reconfigurer s'il apparaît qu'un module d'actionneur n'a plus besoin de coopérer. D'un point de vue mathématique, cela s'apparente à un graphe ω multi-niveau qui, selon les mêmes principes de renforcement ou d'inhibition, fait émerger un ensemble de super-nœuds cohérents.

La conclusion est que lorsqu'on **observe** à chaque palier cette *même* logique de commande auto-organisée, on obtient des **paysages fractals** dans la commande motrice. Il y a un continuum entre de petites sous-entités (couple de joints) et l'entité globale (tout le robot), en passant par des modules (un bras, une base de locomotion), tous gérés par la même mise à jour ω . Cela produit une **cohérence** hiérarchique, une plasticité face au changement de tâche, et des dynamiques macroscopiques qui émergent naturellement d'une multiplication de petits ajustements microscopiques. L'**approche fractale** n'est donc pas qu'un concept abstrait : elle garantit qu'à chaque échelle, la commande obéit à la même *grammaire* DSL, donnant au robot la capacité de gérer un large spectre de configurations motrices sans requérir un contrôle monolithique.

6.6.4. Agents Conversationnels Riches et Contextuels

Le domaine des **agents conversationnels** (chatbots, systèmes de dialogue) bénéficie lui aussi d'une organisation **multi-échelle**. En effet, une conversation ne se limite pas à une suite linéaire de répliques : elle se structure en **segments, sous-thèmes, intentions** globales, etc. Le **DSL** (Deep Synergy Learning), avec son mécanisme d'**auto-organisation** (pondérations adaptatives ω , agrégation/division multi-niveau), offre un cadre où :

- 443. Les **segments** de conversation (micro) peuvent se regrouper selon leur cohérence ou leur synergie sémantique,
- 444. Les **intentions** ou **topics** plus larges (macro) émergent de l'agrégation de ces segments,
- 445. Le réseau conversationnel peut se réorganiser au fil du dialogue, gérant le contexte de manière dynamique.

Dans ce qui suit (6.6.4.1), nous détaillons comment, au *niveau micro*, on manipule des **segments** de conversation, tandis qu'au *niveau macro*, on repère des **intentions** ou **sujets** d'échange.

6.6.4.1. Micro : Segments de Conversation, Macro : Intentions ou Topics

Dans l'analyse conversationnelle, il est naturel de considérer la discussion sous différents paliers : au niveau **micro**, on manipule des segments (ou “turns”) correspondant à une question, une réponse, ou un bloc de discours. Au niveau **macro**, ces segments s'agrègent pour former des intentions ou “topics” plus larges, tels que l'enchaînement des idées autour d'une question “météo” ou d'une requête “horaires de train”. Le **Deep Synergy Learning (DSL)** offre une manière de faire émerger automatiquement cette organisation multi-niveau : le niveau micro regroupe les segments proches, et le niveau macro identifie les intentions sous-jacentes.

Il est d'abord utile de décrire comment se traite le **niveau micro** des segments de conversation. Chaque segment \mathcal{E}_i représente un bloc de discours (par exemple, une phrase d'un utilisateur, une réponse du chatbot) que l'on encode sous la forme d'un vecteur sémantique \mathbf{v}_i (embedding). On introduit une **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ mesurant la similarité sémantique ou contextuelle entre deux segments. Une fonction usuelle est la similarité cosinus :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}.$$

Dans le DSL, on maintient pour chaque couple (i, j) une pondération $\omega_{i,j}$, mise à jour à l'aide de l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est un taux d'apprentissage et $\tau > 0$ un facteur de décroissance. Lorsque deux segments de conversation \mathcal{E}_i et \mathcal{E}_j abordent un même sujet, leurs embeddings sont proches, ce qui donne une synergie $S(i, j)$ élevée et fait croître $\omega_{i,j}$. Un ensemble de segments fortement interconnectés peut dès lors constituer un **cluster** local, reflétant la poursuite d'un sujet précis (questions et réponses cohérentes, réactions enchaînées, etc.).

Une fois qu'au palier micro on observe des groupes de segments fortement reliés, on peut passer au **niveau macro**. L'agrégation de ces segments permet de définir des super-nœuds, c'est-à-dire des **intentions** ou **topics** plus vastes. On applique généralement une fonction Ψ pour agréger les pondérations micro en $\omega_{\alpha,\beta}^{(\text{macro})}$, selon

$$\omega_{\alpha,\beta}^{(\text{macro})} = \Psi(\{\omega_{i,j} \mid i \in \alpha, j \in \beta\}),$$

où α, β sont des ensembles de segments (clusters) déjà identifiés. On peut choisir Ψ comme une moyenne, une somme, ou un maximum. On obtient alors un graphe macro où chaque super-nœud représente un **topic** (par exemple “discussion sur la météo”, “demandes d'information sur les horaires de train”), et où les arcs $\omega_{\alpha,\beta}^{(\text{macro})}$ traduisent la proximité ou la cohérence entre ces topics (si la conversation a fusionné deux thématiques).

Il est intéressant de rapprocher ce fonctionnement d'un système de chatbot “classique”. Généralement, l'identification d'intentions s'effectue via un classifieur supervisé qui assigne chaque phrase à un label d'intention (ex. salutations, question sur la météo, etc.). Dans le DSL, l'intention n'est pas imposée à l'avance, mais **émerge** de la synergie entre segments. On n'a pas besoin de spécifier une liste fermée d'intentions : si un nouveau sujet apparaît, le DSL l'isole en créant un nouveau cluster de segments corrélés. Cette plasticité s'avère particulièrement utile lorsqu'on veut gérer **plusieurs** sujets en parallèle, sans entremêler leurs segments de manière chaotique.

On peut illustrer ce mécanisme par un petit exemple numérique. Considérons des segments $\{E_1, E_2, E_3, \dots\}$. Les similarités $S(E_1, E_2) = 0.8$, $S(E_2, E_3) = 0.7$, etc., alimentent la mise à jour DSL. Les pondérations $\omega_{1,2}$ et $\omega_{2,3}$ croissent vers $0.8/\tau$ et $0.7/\tau$. En conséquence, on regroupe $\{E_1, E_2, E_3\}$ en un cluster (la même discussion autour d'un thème local). De leur côté, $\{E_4, E_5\}$ peuvent constituer un autre cluster s'ils abordent un sujet distinct (par exemple, un bavardage sur la cuisine). On obtient alors deux **topics**.

Sur le plan pratique, ce schéma confère des **gains** en analyse conversationnelle. Les segments qu'un agent conversationnel reçoit sont organisés en clusters, ce qui permet à l'agent de “savoir” quel sous-ensemble discute d'un sujet donné. Lorsqu'un nouveau segment \mathcal{E}_{new} arrive, il calcule la synergie $S(\mathcal{E}_{\text{new}}, E_i)$ pour les segments existants $\{E_i\}$. Si cette synergie est forte avec ceux du topic “météo”, alors $\omega_{\text{new},i}$ se renforce, et \mathcal{E}_{new} rejoint ce topic, assurant une **contextualisation** immédiate. S'il n'existe pas de topic correspondant (faible synergie avec tout l'existant), le DSL alloue un **nouveau** cluster pour cette thématique émergente.

Le DSL multi-niveau permet également qu'il y ait **coexistence** de plusieurs topics en parallèle : un fil sur la météo, un fil sur la cuisine, éventuellement un fil sur l'économie, etc. Le système ne force pas un “mode monothématisé”, comme le ferait un classifieur d'intentions qui assignerait chaque phrase à une intention unique. Ici, les topics constituent des **super-nœuds** agrégés depuis la matrice $\omega_{i,j}$ des segments, et peuvent même avoir des liaisons modestes ou élevées entre eux si la conversation fait des transitions thématiques.

En conclusion, au **niveau micro**, on considère que chaque segment de conversation est une entité \mathcal{E}_i . Le DSL calcule $\{\omega_{i,j}\}$ selon une similarité sémantique, révèle des micro-clusters de segments cohérents, puis agrège ces micro-clusters en **intentions** ou **topics** (niveau macro). Le caractère adaptatif du DSL fait que l'arrivée de nouveaux segments réactualise localement ω , permettant de fusionner ou de scinder des topics sans nécessiter un apprentissage global ou une taxonomie préalable. L'agent conversationnel exploite alors ces topics pour gérer le **contexte** et le **flow** entre multiples sujets, assurant une **robustesse** et une **souplesse** rares dans les architectures de chatbot plus classiques.

6.6.4.2. Multi-Échelle : La Conversation se Hiérarchise en Sous-Thèmes, Thèmes plus Généraux

Lors d'un échange conversationnel, qu'il s'agisse d'un dialogue humain-humain ou d'un système de chatbot interagissant avec plusieurs utilisateurs, on observe fréquemment plusieurs **sous-thèmes** parallèles qui s'insèrent dans un **thème** plus global ou un **contexte** englobant. Le **Deep Synergy Learning (DSL)**, par sa faculté de créer des clusters localement (niveau micro) puis de les **agréger** ou de les **diviser** au fil du temps (niveau macro), permet une **hiérarchisation** organique de la conversation. Les paragraphes qui suivent décrivent comment, au sein d'un DSL, des segments localement liés par la synergie forment des sous-thèmes, et comment ces sous-thèmes à leur tour se rejoignent (ou se distinguent) pour générer des thèmes d'échelle supérieure, parfois avec un caractère **fractal** lorsque la même structure se répète à divers niveaux.

A. Du Sous-Thème Local au Thème Général : Construction Progressive

Tout d'abord, on considère la conversation découpée en **segments** $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Chaque segment (phrase, réplique ou bloc de texte) est relié aux autres par des pondérations $\omega_{i,j}$. Dans la **logique** DSL, la mise à jour itérative

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

se fonde sur une **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (similarité sémantique, continuité thématique, etc.). Au fil des itérations, on voit émerger des **clusters** regroupant des segments qui traitent le même micro-sujet, comme “recherche de billets de train” ou “discussion anecdotique sur la météo”. Chacun de ces clusters peut être assimilé à un **sous-thème** local.

Une fois ces sous-thèmes définis, le système agrège les pondérations internes pour former des **super-nœuds** $\{\mathcal{N}_\alpha\}$. Si plusieurs sous-thèmes se trouvent eux-mêmes synergiques (par exemple, tous traitent de voyages, l'un portant sur le train, l'autre sur l'avion, un troisième sur les hôtels), ils se **fusionnent** en un **thème** plus large. Pour ce faire, on utilise une fonction d'agrégation Ψ :

$$\omega_{\alpha,\beta}^{(\text{macro})} = \Psi(\{\omega_{i,j}\}_{i \in \alpha, j \in \beta}),$$

puis on applique la même règle DSL ou un critère \mathcal{H} (hétérogénéité) pour décider de la **fusion** ou de la **division** de thèmes. Ainsi, un grand thème “voyage” peut s’auto-organiser en sous-thèmes “train”, “avion” et “réservation d’hôtel”.

B. Logique Fractale dans la Hiérarchisation de la Conversation

Il est fréquent qu'un même **concept** (ex. “déplacements”) apparaisse à différents niveaux : d'abord, on l'évoque localement dans quelques segments (détails de billets), puis on le retrouve sous forme d'un sous-thème regroupant ces segments, et, enfin, on le voit ressurgir à un niveau plus global où la conversation discute d'un projet de voyage plus vaste. Cette forme de récurrence multi-niveau rappelle la notion de **fractal** (chap. 6.3) : la *même structure* (discussion autour du déplacement) se répète à plusieurs échelles de granularité.

Dans un DSL, la *même* règle de mise à jour $\omega(t+1) = \omega(t) + \dots$ se rejoue entre :

- les entités segments (niveau micro),
- (ii) les super-nœuds sous-thèmes (niveau intermédiaire),
- (iii) les super-nœuds plus englobants (niveau macro).

Le schéma d'agrégation–division (chap. 6.5) se reproduit donc à plusieurs paliers, générant une structure de clusters (ou super-nœuds) “auto-similaire” d'un niveau à l'autre. On peut donc, en certain sens, qualifier de “fractale” cette hiérarchie conversationnelle, puisqu'elle reproduit un *motif* identique d'**auto-organisation**.

C. Avantages Concrets pour l'Agent Conversationnel

Quand un **agent** conversationnel identifie un sous-thème α (par ex. “le train de demain matin”), il possède la liste des segments $\{\mathcal{E}_i\}$ rattachés à α . S'il constate que α se joint à un autre sous-thème β (“réservation d'hôtel”) sous un thème

plus large “voyage”, il unifie son contexte : l’agent peut thus “comprendre” que tous ces segments s’inscrivent dans un plan de voyage global. Cela favorise la **cohérence** dans les réponses et la possibilité de naviguer entre micro-détails (“prix du billet de train”) et macro-sujet (“voyage en Europe la semaine prochaine”).

Dans une conversation chaotique où plusieurs sujets se déploient, le DSL autorise la **coexistence** de clusters (sous-thèmes distincts), lesquels s’agrègent ou se divisent au fil des nouveaux segments. On n’a pas besoin d’une structure préétablie. Chaque **nouveau** segment \mathcal{E}_{new} calcule sa similarité $S(\mathcal{E}_{\text{new}}, \mathcal{E}_i)$ avec les segments existants, renforce ou diminue $\omega_{\text{new},i}$, et finit par rallier un sous-thème ou en créer un nouveau. L’agent peut traiter simultanément plusieurs fils de discussion.

Si un grand sous-thème devient trop large, la procédure DSL (chap. 6.5.2) peut déclencher une scission top-down : on sépare un “macro-sous-thème” en deux plus petits. Inversement, deux sous-thèmes trop proches fusionnent. Le tout s’actualise localement dans la matrice ω , sans exiger un recalcul complet. Cette **adaptation** soutient les dialogues de longue durée ou hautement changeants.

Conclusion

Au sein d’une **conversation** complexe, la **hiérarchie multi-niveau** permet de distinguer :

446. **Sous-thèmes** (niveau micro) : petites grappes de segments, localement synergiques,

447. **Thèmes** globaux (niveau macro) : unification de sous-thèmes fortement apparentés.

Dans un **DSL**, ces *clusters* se forment ou se dissolvent par la mise à jour adaptative de ω . Dès lors que la même *logique* (agrégation, division, feedback descendant) s’applique à plusieurs paliers, on peut parler de **récurrence fractale**. La **répétition** d’un même patron sémantique (micro→macro) reflète la manière dont un concept local (ex. “détails de tickets de train”) s’intègre dans un **contexte** plus vaste (“voyage” ou “organisation d’un séjour”). L’**agent** conversationnel bénéficie ainsi d’une structure interne naturellement évolutive, capable de naviguer entre plusieurs fils de discussion et d’agréger ces fils si une cohérence plus générale se dégage, ou de scinder un sujet si la conversation se subdivise, le tout sans nécessiter un cadre de topics pré-codé ou un entraînement supervisé rigide.

6.6.4.3. Avantage : Le DSL Repère des “Patrons Fractals” de Dialogue ou d’Échanges

Contexte et Enjeu. Dans un **échange** ou un **dialogue**, il n’est pas rare d’observer la récurrence d’un même *modèle d’interaction* à différents niveaux de granularité. Par exemple, on peut avoir un motif local (question–réponse) reproduit à l’échelle plus large (ensemble de sous-thèmes, chacun contenant des séquences de questions–réponses) et enfin à l’échelle globale (plusieurs blocs de discussion qui suivent la même logique). Lorsqu’un **Deep Synergy Learning (DSL)** multi-niveau est appliqué à l’analyse et la structuration du dialogue, cette **auto-similarité** récurrente s’assimile à une **fractalité**. Les développements qui suivent décrivent comment le DSL, par sa démarche d’agrégation et de division hiérarchique, parvient à **repérer** des “patrons fractals” dans l’agencement des segments, et en quoi cela **profite** à l’agent conversationnel.

A. Patrons Fractals : Définitions et Parallèles

Une structure fractale se caractérise par l'**auto-similarité** : la même forme ou le même schéma se reproduit à différentes échelles. Dans le contexte d’un dialogue, on peut voir un petit motif local (ex. “question–réponse–clarification”) réapparaître à une échelle plus large (plusieurs tours de parole structurant la même logique), et éventuellement au niveau d’un macro-sujet (une suite de sous-thèmes organisés en question–réponse). Le DSL, en utilisant la même **règle** (mise à jour des liaisons ω , agrégation, etc.) à chaque palier, capte naturellement cette **auto-similarité**.

Exemple de motif fractal.

Un motif “A pose une question, B répond, A clarifie” peut se répéter dans une conversation courte (quelques segments) ou plus longue (série de sous-thèmes). Chaque *patern* (A, B, C) se traduit par des pondérations $\omega_{i,j}$ élevées lorsque le segment “B” se lie fortement à “A” (réponse cohérente) et “C” se lie à “B” (clarification). Au niveau macro, on retrouve un agrégat de telles triades, mettant en évidence la même structure conversationnelle répétée.

B. Mécanisme DSL de Détection Fractale

Hiérarchie ascendante (agrégation).

Dans le DSL, on commence par **analyser** les segments individuels $\{\mathcal{E}_i\}$. Si deux segments $\mathcal{E}_i, \mathcal{E}_j$ se montrent récurrents dans la même logique (ex. question-réponse), leur pondération $\omega_{i,j}$ grimpe. Si, en outre, un troisième segment \mathcal{E}_k s'active avec $\{i, j\}$ (par ex. clarification), on peut aboutir à un cluster $\{\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k\}$ correspondant à un schéma local. Au palier supérieur, si plusieurs de ces triades suivent une *même* structure, le DSL agrège les clusters en un **super-nœud** “pattern conversationnel”.

Pour formaliser cela, on définit une *énergie* ou un *score* inter-segments. Si un motif $\mathcal{M} = \{\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c\}$ reproduit la séquence question-réponse-clarif, alors la somme

$$\Omega(\mathcal{M}) = \sum_{(x,y) \in \mathcal{M} \times \mathcal{M}} \omega_{x,y}$$

peut devenir élevée (les segments $\mathcal{E}_a, \mathcal{E}_b, \mathcal{E}_c$ se relient fortement). Lorsque cette somme $\Omega(\mathcal{M})$ dépasse un certain θ_{motif} , on identifie \mathcal{M} comme un “pattern” stable. Plusieurs motifs analogues $\mathcal{M}', \mathcal{M}''$ peuvent avoir la même structure, révélant la *recurrence* fractale.

Au niveau macro, la même **logique** DSL agit sur ces motifs (clusters) entre eux. Si l'on retrouve \mathcal{M} ou un motif similaire \mathcal{M}' dans un autre sous-thème, on observe une **auto-similarité** dans la distribution des liens. On peut donc parler de fractalité, car l'organisation question-réponse-clarif se répercute à une échelle plus large, comme un “shéma” s'appliquant à plusieurs groupes de segments.

C. Avantages pour l'Agent Conversationnel

Un utilisateur peut aborder plusieurs sous-thèmes de manière identique (à chaque fois : question, réponse, clarification). Le DSL, en reconnaissant ce *pattern*, permet à l'agent de mieux **anticiper** la structure attendue. Il peut par exemple plus rapidement fournir des réponses plus ciblées si la séquence se répète.

Le motif fractal n'est pas rigide. On repère la forme d'échange (ex. question-réponse-clarif) peu importe le *contenant* lexical. Le DSL, via la similarité $\omega_{i,j}$, autorise une grande variété de formulations sans casser le *pattern*. Ainsi, l'agent conversationnel peut exploiter ce schéma structurel pour organiser ses réactions, même si le contenu des segments varie beaucoup.

Dans des dialogues massifs (plusieurs participants, de multiples bifurcations), repérer des **motifs fractals** offre un mode de compression : plutôt que de gérer un graphe monolithique de segments, on sait qu'un certain motif local se décline en de multiples occurrences. On peut regrouper ces occurrences et naviguer plus efficacement dans la structure du dialogue.

Conclusion

En **obtenant** des “patrons fractals” dans la structure du dialogue, le **DSL** révèle l'**auto-similarité** de certains schémas conversationnels, à différents paliers : un *pattern* local (question-réponse-clarif) peut se reproduire dans des sous-thèmes, puis se retrouver encore en macro-sujets. La clé de voûte de cette détection fractale est la *répétition* de la même **règle** (mise à jour ω , agrégation, division, feedback) à chaque niveau. Cette configuration procure à l'**agent** conversationnel :

448.Une **reconnaissance** de schémas conversationnels récurrents,

449.Une **facilité** à gérer simultanément plusieurs occurrences du même motif (ex. multiples questions-réponses structurées),

450.Une **architecture** hiérarchique flexible, où les segments se groupent localement, puis s'unissent au macro-niveau, sans imposer d'a priori strict sur la forme ou le contenu.

Le résultat est un système **robuste** et **évolutif**, capable de cartographier un large dialogue en identifiant les *mêmes formes* d'échange qui se déclinent à divers degrés de granularité, formant ainsi un **patron fractal** de la conversation.

6.6.5. Applications dans la Simulation et la Prédiction d'Événements

Les principes **multi-échelle** du DSL (Deep Synergy Learning) trouvent un écho tout particulier dans la **simulation** et la **prédiction** d'événements. En effet, de nombreux phénomènes (économiques, climatiques, épidémiologiques, boursiers, etc.) s'expriment à la fois sous forme d'**événements ponctuels** (micro) et de **tendances** (macro). Le **DSL** permet alors d'**organiser** ces événements de manière adaptative, dévoilant des **mégapatterns** ou tendances globales tout en gardant la granularité des points individuels.

6.6.5.1. Micro : Événements Ponctuels, Macro : Tendances ou Mégapatterns (ex. Bourse, Climat)

Introduction et Cadre d'Analyse. Dans de nombreux flux temporels (données boursières, variables climatiques, événements sporadiques), chaque “point” observé peut être vu comme un **événement** local. Pour en extraire des **tendances** plus globales, on recourt souvent à une analyse multi-niveau : un **niveau micro** prend en charge l'identité et la collaboration ou similarité des événements ponctuels, tandis qu'un **niveau macro** agrège ces événements pour dégager des **tendances** (ou “mégapatterns”) plus vastes, comme un cycle boursier haussier ou un changement climatique global. Le **Deep Synergy Learning (DSL)** fournit une méthodologie pour construire et adapter ces niveaux, tout en traitant l'apparition de nouveaux événements au fil du temps.

A. Niveau Micro : Événements Ponctuels

Il est courant, dans un flux temporel (bourse, climat, logs...), de définir chaque **événement** \mathcal{E}_i comme une entité $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Un événement peut consister, par exemple, en une transaction (bourse) ou une perturbation locale (climat).

Pour deux événements \mathcal{E}_i et \mathcal{E}_j , on introduit une **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ reflétant leur **corrélation** ou leur **similarité** dans l'espace de variables considérées. On peut écrire :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \rho(\mathbf{x}_i, \mathbf{x}_j),$$

où ρ est un coefficient de corrélation (Pearson, Spearman, etc.) ou tout autre indicateur de concomitance. Cette approche s'adapte aisément aux données boursières (corrélation entre deux transactions) ou aux mesures météorologiques (similitude spatio-temporelle de deux perturbations).

Dans la logique DSL, on entretient pour toute paire (i, j) une pondération $\omega_{i,j}$ modifiée selon :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Lorsque deux événements s'avèrent fortement corrélés ou se produisent de manière répétée à des moments similaires, leur synergie S est élevée et $\omega_{i,j}$ augmente. Cela aboutit à la **clusterisation** d'événements ponctuels fortement liés.

B. Niveau Macro : Mégapatterns et Tendances

Une fois que, au niveau micro, les événements $\{\mathcal{E}_i\}$ s'organisent en **clusters** ou sous-ensembles homogènes (transactions identiques, perturbations climatiques similaires), on peut agréger ces ensembles pour former des **super-nœuds** $\{\mathcal{N}_\alpha\}$. Dans le cadre du DSL, on emploie une fonction Ψ :

$$\omega_{\alpha,\beta}^{(\text{macro})} = \Psi(\{\omega_{i,j}^{(\text{micro})}\}: i \in \mathcal{C}_\alpha, j \in \mathcal{C}_\beta).$$

On obtient alors un **graph** macro où chaque super-nœud \mathcal{N}_α correspond à un **ensemble** d'événements corrélés (e.g. transactions à un moment précis ou perturbations climatiques simultanées). Ce graphe macro, lui aussi soumis à la dynamique DSL, permet de repérer quand plusieurs super-nœuds $\mathcal{N}_\alpha, \mathcal{N}_\beta$ partagent une synergie élevée, suggérant qu'ils forment un **mégapattern** (ex. phase de marché haussière, vague de chaleur régionale).

Dans ce macro-niveau, un mégapattern correspond à un **regroupement** de super-nœuds $\mathcal{N}_\alpha, \mathcal{N}_\beta, \dots$ qui coopèrent régulièrement, signant une tendance persistante. En bourse, on reconnaîtra un **Bull Market** si un large bloc de transactions entretient des liens ω élevés (forte concordance de prix en hausse). En climat, on rassemblera plusieurs perturbations locales (orages, vents forts) en un **bloc** cohérent, trahissant une dépression qui s'étend sur une vaste zone ou un effet de “vague de chaleur”.

C. Exemples Concrets

Bourse et événements.

Dans un flux boursier, chaque **transaction** forme un événement ponctuel. On définit la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ par la ressemblance de leur prix, heure, ou “carnet d’ordre”. Les transactions extrêmement proches (mêmes heures, mêmes montants, mêmes actifs financiers) agrègent localement $\omega_{i,j}$ en un **cluster** de micro-événements. Au **niveau macro**, on assemble ces clusters en **phases** (Bull, Bear, neutral), aboutissant à un mégapattern (par ex., un marché systématiquement acheteur pendant deux semaines).

Climat et perturbations.

On conçoit chaque **orage**, chaque **tempête locale**, ou chaque **front froid** comme un événement \mathcal{E}_i ; la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ tient compte de la distance géographique, de l’intervalle temporel, de la similarité d’intensité. Les orages rapprochés s’agrègent localement. Au niveau macro, des ensembles d’orages répétitifs sur une zone conduisent à un mégapattern “vague de tempêtes” ou “saison cyclonique”.

D. Bénéfices Mathématiques et Opérationnels

L’approche DSL se prête particulièrement bien à un flux en **temps réel** ou incrémental. Dès qu’un nouvel événement \mathcal{E}_{new} survient (transaction, perturbation), on l’associe à la matrice ω , on calcule $\omega_{\text{new},j}$ pour les $\{\mathcal{E}_j\}$ déjà existantes, et l’on actualise la **synergie** correspondante. S’il y a suffisamment de liens forts, \mathcal{E}_{new} rejoint un cluster ou, sinon, crée un nouvel ensemble.

Le DSL crée en parallèle un **clustering** de niveau micro (évolution locale) et un de niveau macro (tendances). Cette hiérarchie autorise un compromis entre la **finesse** (événements ponctuels) et la **vision** d’ensemble (cycles, vagues de température). En $O(n)$ ou $O(n^2)$ selon l’algorithme, on peut rafraîchir la structure après chaque événement, voire lot d’événements.

Une fois un mégapattern stabilisé (une synergie $\omega_{\alpha,\beta}^{(\text{macro})}$ élevée entre plusieurs super-nœuds $\{\alpha, \beta\}$), on peut en **déduire** qu’il y a de fortes chances pour que les prochains événements prolongeant cette cohérence se rattachent au même pattern. Sur le plan boursier, on anticipe la poursuite d’une tendance haussière ; sur le plan climatique, on anticipe la prolongation d’une vague de chaleur. Le **DSL** fournit ainsi un mécanisme de repérage de phénomènes étendus sans recourir à une modélisation paramétrique rigide.

Conclusion

À l’échelle **micro**, le DSL considère chaque **événement** (transaction, perturbation) comme une entité \mathcal{E}_i . La **dynamique** de pondération $\omega_{i,j}$ révèle des micro-clusters **cohérents**. Au **niveau macro**, on agrège ces micro-clusters et on détecte des **mégapatterns** ou **tendances** plus globales. Les points saillants sont :

- 451. **Agrégation** (bottom-up) : de l’événement individuel au méga-ensemble d’événements,
- 452. **Adaptation** continue : chaque nouvel événement met à jour localement ω , reconfigurant si besoin les patterns,
- 453. **Structure** multi-niveau explicite : le robot (ou le système d’analyse) identifie à la fois les détails (chacun des événements) et la “vue d’ensemble” (tendances).

On obtient ainsi un **réseau** où la **complexité** (n événements) est amortie par l’auto-organisation : au fil des itérations, les **pondérations** ω s’ordonnent en clusters micro, puis en **macro-nœuds** de tendance ou mégapattern. Ce fonctionnement est particulièrement adapté aux **séries temporelles** évolutives (bourse, climat...), où la **détection** et la **mise à jour** de grandes structures (phases de marché, périodes climatiques) se fait **progressivement** et **auto-organisé**.

6.6.5.2. Éventuels Invariants d'Échelle (Lois de Puissance) dans les Distributions d'Événements

Contexte et Logique Générale. Il existe de nombreux phénomènes, tant en bourse qu'en climatologie, épidémiologie ou sismologie, où les événements observés (transactions financières, orages, éruptions volcaniques, etc.) semblent se répartir selon des **lois de puissance** plutôt que des lois gaussiennes. Cette caractéristique se décrit comme *scale-free* ou fractale, car on y discerne une **auto-similarité** : à différentes **échelles** d'observation, on retrouve la même forme de distribution. Le **Deep Synergy Learning (DSL)** s'avère particulièrement pertinent pour analyser et exploiter ces invariants d'échelle, grâce à sa capacité d'**auto-organisation** multi-niveau des événements.

A. Lois de Puissance et Distributions "Scale-Free"

Une **loi de puissance** (ou loi Pareto) se manifeste lorsque la probabilité $P(X > x)$ d'observer un événement d'**intensité** ou de **taille** supérieure à x se comporte comme $x^{-\alpha}$ pour $\alpha > 0$. Concrètement, cela signifie qu'il existe beaucoup d'événements de taille modeste, mais que des événements extrêmement grands (krach boursier, cyclone majeur, séisme de haute magnitude) restent non négligeables, beaucoup plus fréquents que ne le prédirait une loi normale.

Dans les **marchés financiers**, la queue de la distribution des variations de prix s'approche souvent d'une loi de puissance, expliquant les "fat tails" (événements extrêmes plus fréquents qu'en gaussien). En **climat**, l'intensité des tempêtes ou des inondations peut s'inscrire dans une dynamique *heavy-tailed*. En épidémiologie, la survenue d'un foyer extrêmement infectieux correspond à un comportement hors de la zone médiane.

On parle de *scale-free* lorsque la distribution ne change pas fondamentalement si l'on "change d'unité" ou si l'on prend un "zoom" plus ou moins large. Dans un cadre fractal, cette invariance d'échelle suggère une **auto-similarité** : ce que l'on voit au niveau local (quelques événements) se reflète au niveau plus global (cluster d'événements, macro-structures). Le DSL, en permettant un agrégation hiérarchique, peut révéler cet invariant d'échelle dans la distribution de la taille des clusters.

B. DSL et Détection des Invariants d'Échelle

Dans un **DSL**, on définit un **graph** de pondérations $\omega_{i,j}$ reliant des **événements** $\{\mathcal{E}_i\}$. La mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

permet de renforcer ou d'affaiblir les liens en fonction de la synergie S , qui peut refléter une proximité temporelle, géographique, ou d'intensité. Lorsque le processus est **multi-niveau**, on construit des super-nœuds (macro) par agrégation Ψ . La distribution des degrés (ou des tailles de cluster) peut, dans un phénomène scale-free, présenter une loi de puissance $P(k) \sim k^{-\alpha}$. Le **DSL** ne postule pas a priori de gaussien ou non, mais organise les événements en clusters, possiblement **dominés** par quelques "hubs" et un grand nombre de petits nœuds, reproduisant la signature d'un réseau scale-free.

Dans cette structure, le niveau **micro** prend en charge chacun des événements \mathcal{E}_i . Lorsque plusieurs \mathcal{E}_i s'avèrent fortement liés, ils forment un **cluster** local. Des **clusters** très volumineux (quelques macro-événements) apparaissent, alors qu'une multitude de petits clusters restent marginaux. Ce **déséquilibre** (quelques gros hubs vs. beaucoup de petits nœuds) signale un possible *power-law*. Au niveau macro, les super-nœuds reproduisent le même schéma : on obtient encore quelques super-nœuds massifs (mégapatterns) et beaucoup de plus petits.

En théorie des réseaux complexes, un graphe **scale-free** est défini par une distribution en loi de puissance de ses degrés $\{k_i\}$. Si on associe chaque événement \mathcal{E}_i à un noeud et on calcule le degré en fonction de $\{\omega_{i,j}\}$ (ou de la somme de ces poids), on peut retrouver un $P(k) \sim k^{-\gamma}$. Le DSL, par son mécanisme de croissance (renforcer les liens les plus utiles, laisser s'éteindre les autres), tend à engendrer ou consolider ce genre de structure, avec quelques nœuds "majoritairement connectés", correspondant aux événements les plus extrêmes ou les plus récurrents.

C. Conséquences et Intérêts Pratiques

Une loi de puissance implique que des événements extrêmes, bien que rares, ne sont pas négligeables : on n'a pas la décroissance exponentielle rapide d'une gaussienne. L'**auto-organisation** DSL, en reconnaissant des **macro-nœuds** de grande taille (ex. synonyme de forte corrélation de plusieurs événements), signale la présence d'un "hub" ou d'un

“mégapattern” significatif. On peut alors anticiper que de nouveaux événements \mathcal{E}_{new} se rattachent à ce hub si la même dynamique persiste (ex. marché boursier restant en mode bull, climat restant dans une séquence orageuse).

De nombreux modèles statistiques se heurtent à la difficulté des distributions à queue lourde. Le DSL n’impose pas de forme paramétrique : il adapte ω à ce qui se manifeste, laissant s’exprimer la queue lourde (quelques très gros clusters) ou l’absence de seuil net. Cela évite les **erreurs** qu’entraînerait un a priori gaussien.

Pour mettre en évidence l'**invariance d’échelle**, on peut, par exemple, étudier la distribution des tailles de clusters (au niveau micro, puis macro). S’il apparaît une pente stable en log–log, on conclut à la fractalité. Le DSL, en soulignant les grands clusters dominants ou en révélant une multiplicité de petits clusters, conforte cette analyse : on vérifie qu’il y a un “hub and spoke” structure dans le graphe ω , typique d’une *scale-free network*.

Conclusion

Dans l’étude de flux d’événements, qu’il s’agisse de **bourse**, de **climat** ou d’autres domaines où la **distribtuion** peut présenter des **lois de puissance** :

454. Le **niveau micro** du DSL regroupe des **événements** $\{\mathcal{E}_i\}$ selon leur **synergie** (corrélation, proximité, etc.),

455. Le **niveau macro** agrège ces groupes en **mégapatterns** ou **tendances** (ex. bull market, vague de chaleur),

456. Le **caractère scale-free** (fat tails, invariance d’échelle) se retrouve dans la formation de quelques **macro-clusters** très importants, tout en laissant une myriade de petits clusters.

Ce **phénomène** fractal ou scale-free n’est pas exceptionnel dans les séries temporelles ou spatiales réelles ; au contraire, on le rencontre dès lors que des événements extrêmes sont plus probables qu’un modèle normal ne le prévoit. Le **DSL**, en s’y adaptant sans supposer de forme a priori, apporte un **outil** robuste de représentation multi-niveau, révélant et exploitant les invariants d’échelle présents dans la dynamique des événements.

6.6.5.3. Gains : Meilleure Anticipation, Clusterisation Plus Lisible

Justification et Cadre Global. Les sections précédentes (6.6.5.1–6.6.5.2) ont abordé l’application du **Deep Synergy Learning (DSL)** à des scénarios de simulation ou de prédiction d’événements (transactions boursières, données climatiques, épidémies, etc.). Dans ce contexte, chaque événement ponctuel forme une entité \mathcal{E}_i au niveau micro, tandis que le **DSL** agrège ces entités en super-nœuds macro pour déceler des “mégapatterns” ou tendances globales. Cette structuration multi-échelle engendre deux avantages concrets : une **meilleure anticipation** d’évolutions (ou crises) et une **clusterisation** beaucoup plus lisible, facilitant l’analyse et la prise de décision.

A. Meilleure Anticipation des Évolutions ou Crises

Renforcement local et indicateurs d’alarme.

Dans la dynamique DSL, on maintient pour chaque paire (i, j) un poids $\omega_{i,j}$. Quand un ensemble d’événements $\{\mathcal{E}_k\}$ se révèle fortement corrélé — par exemple, une série de variations boursières liées ou un ensemble de perturbations climatiques proches — leurs pondérations $\omega_{k,\ell}$ s’élèvent, formant un **cluster** local. Lorsque la somme

$$\Omega(\mathcal{C}_\alpha) = \sum_{k,\ell \in \mathcal{C}_\alpha} \omega_{k,\ell}$$

d’un cluster \mathcal{C}_α dépasse un certain seuil θ , on en déduit qu’un **mégapattern** naît (au palier macro). Il se peut qu’il s’agisse d’un début de bulle spéculative en bourse ou d’une instabilité météorologique se généralisant. Le fait de surveiller $\Omega(\mathcal{C}_\alpha)$ ou la taille de ce cluster peut fournir un **indicateur** d’émergence potentielle de crise ou de phénomène extrême.

Anticipation de phénomènes extrêmes.

La logique du DSL **prévient** l'analyste ou l'automate d'une dérive importante dès que les pondérations ω se synchronisent autour d'un **sous-groupe** d'événements. Contrairement à un modèle statistique statique (par ex. Gaussien), le DSL s'adapte localement, non seulement aux moyennes, mais aussi aux corrélations actives. Dès la mise en place d'une concentration d'événements inhabituels, le **cluster** associé grandit — signe avant-coureur. Cela facilite des **alertes** précoce, car la synergie locale s'affirme avant même que le mégapattern ne devienne flagrant à la simple observation des moyennes.

B. Clusterisation Plus Lisible et Hiérarchie Compréhensible

Dans des flux de données complexes (bourse, climat, logs de grande dimension), la simple liste d'événements peut être incohérente ou surchargée. Le DSL procure une **hiérarchie** : d'abord, la mise à jour ω repère les micro-regroupements, puis l'**agrégation** (via Ψ) construit un graphe macro où chaque super-nœud représente un **mégapattern** important. Cela se traduit par une **réduction** de la complexité : quelques super-nœuds dominants reflètent la structure globale, tandis que les **petits** clusters assurent une granularité fine pour ceux qui souhaitent analyser les détails.

Au niveau **macro**, les clusters identifiés correspondent à des phases ou des régimes (ex. bull market, "saison cyclonique"), qui sont **significatifs** pour la prise de décision. L'analyste, le trader ou le climatologue n'a pas besoin de gérer individuellement des milliers d'événements : ils peuvent se référer à 2 ou 3 grands mégapatterns. Sur le plan **mathématique**, cette clarification résulte de la **dynamique** DSL, qui renforce certains liens et en éteint d'autres, produisant un graphe plus **lisble** (quelques hubs vs. une multitude de petits nœuds).

Une entité (gestionnaire de portefeuille, centre de prévision météo) peut prendre des décisions plus **ciblées** en considérant seulement la configuration macro des mégapatterns. Si un super-nœud "housse" grossit dangereusement, un opérateur en bourse peut se méfier d'un krach à venir, ou inversement profiter d'une flambée haussière. Si un super-nœud "vague de chaleur" s'impose, les pouvoirs publics peuvent anticiper des plans d'urgence (restriction d'eau, alertes canicule). Le DSL évite donc la trop grande granularité d'un clustering purement local et dévoile une **vision** multi-niveau.

C. Exemple Numérique Illustratif

Considérons un flux de 1 000 **événements** $\{\mathcal{E}_1, \dots, \mathcal{E}_{1000}\}$. Chaque nouvel événement \mathcal{E}_{new} met à jour $\omega_{\text{new},j}$ par la formule

$$\omega_{\text{new},j}(t+1) = \omega_{\text{new},j}(t) + \eta [S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) - \tau \omega_{\text{new},j}(t)].$$

Si \mathcal{E}_{new} se révèle très cohérente avec plusieurs $\{\mathcal{E}_j\}$ déjà existants, elle renforce $\omega_{\text{new},j}$ et rejoint un cluster local \mathcal{C}_α . Au niveau **macro**, si $\Omega(\mathcal{C}_\alpha) = \sum_{i \in \mathcal{C}_\alpha} \sum_{j \in \mathcal{C}_\alpha} \omega_{i,j}$ dépasse un certain θ ou si l'on détecte une corrélation transversale avec un autre cluster \mathcal{C}_β , on forme un **super-nœud** $\mathcal{N}_{\alpha,\beta}$ plus vaste. Cela **allège** la représentation : au lieu de manipuler 1 000 événements individuels, l'analyste regarde 5 super-nœuds macro. L'**anticipation** s'améliore, car un macro-nœud en expansion reflète une **dynamique** forte et peut amorcer une crise ou un pic extrême.

Conclusion

En contexte de **simulation** et **prédiction** d'événements, qu'ils soient boursiers, climatiques ou autres, le **DSL** s'impose comme un outil :

457. **Plus apte à anticiper** les évolutions critiques. Le renforcement local ω autour d'un sous-groupe d'événements indique rapidement l'émergence d'un mégapattern (ex. bulle, vague de chaleur).

458. **Plus lisble** dans sa **clusterisation**, grâce au passage multi-niveau (micro→macro). Les analystes ou décideurs n'ont pas à traiter une infinité de micro-événements ; ils lisent un graphe final clair, où quelques super-nœuds incarnent les **tendances** essentielles.

Cette convergence entre **meilleure anticipation** et **clustering plus lisble** illustre la raison d'être du DSL : concilier la finesse de l'information (chaque événement séparé) et la cohésion d'une **vision globale**, tout en assurant une adaptation continue (chap. 9) lorsque de nouveaux événements surviennent.

6.7. Conclusion

6.7.1. Synthèse des Contributions du Chapitre

- Rappeler l'importance de l'**apprentissage multi-échelle** dans le DSL, la possibilité d'observer une **fractalité** ou une auto-similarité à divers paliers, et la manière dont les flux bottom-up / top-down s'organisent.

6.7.2. Rôle de la Fractalité

- Insister sur le fait que la fractalité peut décrire des phénomènes d'**auto-similarité** structurelle quand la répartition des entités et des synergies respectent certaines règles.
- Avantage conceptuel : mieux comprendre la logique d'emboîtement des clusters.

6.7.3. Limites et Perspectives

- Méthodes fractales ne s'appliquent pas à tous les SCN ; c'est un cas où les patterns de synergie reproduisent à chaque échelle les mêmes lois d'organisation.
- Perspectives : explorer plus en détail la dimension fractale, la concevoir comme un indicateur de "cohérence" multi-niveau dans le DSL.

6.7.4. Liens avec les Chapitres Suivants

- Chap. 7 (optimisations) : la multi-échelle et la fractalité peuvent influencer la manière d'optimiser (recuit multi-niveau, etc.).
- Chap. 8 (multimodal) : on peut retrouver des structures fractales dans des flux très hétérogènes.
- Chap. 9 (temps réel) : gérer la fractalité en flux continu, adaptation incrémentale des clusters à différentes granularités.

6.7.5. Vision Globale

- L'apprentissage multi-échelle s'avère un **pilierset** pour sculpter la complexité du DSL : il offre une **structure** et des **concepts** (dont la fractalité) qui améliorent la lisibilité et la robustesse de l'auto-organisation.

6.7. Conclusion

Au fil de ce chapitre 6, nous avons mis en avant l'importance de l'**apprentissage multi-échelle** dans le **DSL** (Deep Synergy Learning) et la **possibilité** d'observer une **fractalité** ou une forme d'auto-similarité structurelle à divers paliers. Nous avons également détaillé la façon dont les **flux ascendants** (bottom-up) et **descendants** (top-down) interagissent, soutenant une **dynamique** qui relie intimement les micro-clusters (niveau local) et les macro-structures (niveau global).

6.7.1. Synthèse des Contributions du Chapitre

Apprentissage Multi-Échelle.

L'ensemble des sections étudiées a mis en lumière la **structure** multi-niveau déployée par le **Deep Synergy Learning (DSL)** : plutôt que de se contenter d'un unique palier (comme dans de nombreux modèles neuronaux ou algorithmes de clustering), le DSL organise les **entités** observées selon plusieurs **niveaux** distincts. Les **éléments** au niveau **micro** (patches visuels, tokens textuels, capteurs unitaires, transactions ponctuelles, etc.) sont d'abord reliés par des pondérations $\{\omega_{i,j}\}$ mises à jour via l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Cette étape conduit à la formation de **clusters** ou regroupements locaux. Ensuite, une **fonction d'agrégation** Ψ (voir la section 6.2) permet de “monter” à un niveau **macro** ou **méso** : on crée des **super-nœuds** \mathcal{N}_α correspondant à des ensembles de grande cohérence, puis l'on met à jour les pondérations $\omega_{\alpha,\beta}$ entre ces super-nœuds au moyen de la même règle DSL. Cette **hiérarchisation** s'exprime sous forme d'un **flux ascendant** (bottom-up), où les clusters micro se combinent pour former des macros plus vastes, et d'un **flux descendant** (top-down), par lequel le niveau macro impose un *feedback* pour clarifier, scinder ou réassigner les liens micro si un sous-ensemble se révèle incohérent.

Fractalité et Auto-Similarité.

Il est apparu que dans plusieurs domaines – vision multimodale, conversation, robotique, événements temporels (bourse, climat) –, une **répétition** de motifs se manifeste à diverses échelles : une même trame structurelle ou un même “pattern” se retrouve aussi bien au niveau local (quelques entités) qu'au niveau global (macro-nœuds). Cette **auto-similarité** est l'essence de la **fractalité**, s'illustrant dans les lois de puissance (queues lourdes, distributions scale-free) et dans la répétition de motifs d'interactions (ex. séquences dialogue, organisations sensorimotrices). Le **DSL**, grâce à sa logique d'**auto-organisation** fondée sur la mise à jour $\omega_{i,j}(t+1) = \dots$, met en évidence ces invariants d'échelle : il amplifie les liens dans les zones denses (clusterisation locale), puis agrège ces clusters en super-nœuds macros, pouvant révéler des “hubs” dominants ou des structures globales à la topologie fractale.

Organisation des Flux.

Un point central est la **gestion** conjointe de deux types de flux. D'un côté, un **flux ascendant** (bottom-up) produit les agrégations successives – chaque palier transforme ses entités en super-nœuds pour le palier supérieur. De l'autre côté, un **flux descendant** (top-down) s'exprime en imposant un **feedback** sur les pondérations micro en fonction du contexte macro. Cette rétroaction, dans la formulation DSL, prend la forme d'un **terme** Δ_{down} ou d'une modulation de $\omega_{i,j}$. Elle agit comme un mécanisme de validation, de scission ou d'inhibition quand le niveau macro perçoit une incohérence. Cette **cohérence** entre flux ascendants (consolidation locale) et flux descendants (guidage contextuel) garantit la **stabilité** de la structure globale, évitant les errances ou les convergences trop partielles.

Ainsi, ce chapitre a mis l'accent sur la **puissance** de la démarche DSL pour gérer la **multi-échelle** : qu'il s'agisse de patches d'image s'assemblant en classes sémantiques, de segments de conversation se regroupant en intentions, de capteurs robotique menant à un comportement complet, ou d'événements ponctuels s'agrégant en mégapatterns, la même **logique** d'auto-organisation fait apparaître à la fois une finesse locale et une vision macro. Les **phénomènes** fractals ou *scale-free* qui émergent (patterns répétitifs, lois de puissance) n'exigent pas une hypothèse statistique rigide : le **DSL** s'adapte localement, identifiant et exploitant toute **répétition** ou **corrélation** à différents paliers, dévoilant ainsi de riches structures hiérarchiques dans des systèmes de grande complexité.

6.7.2. Rôle de la Fractalité

La **fractalité** repose sur l'idée qu'un *même* schéma, observé à une échelle donnée, peut se **répliquer** ou se **réinventer** à d'autres échelles, traduisant ainsi une **auto-similarité** potentielle. Dans le cadre du **Deep Synergy Learning (DSL)**, cette propriété s'exprime par la récurrence des mécanismes d'agrégation et de division à tous les paliers de l'organisation (micro, méso, macro). Les configurations locales, qu'elles correspondent à des clusters de segments de conversation, de capteurs, de patchs visuels ou de transactions, se retrouvent avec une forme analogue au niveau plus global, où l'on regroupe ces mêmes clusters en super-nœuds.

Sur un plan **mathématique**, un système fractal s'identifie fréquemment par l'analyse de la **dimension fractale** d'un graphe ou d'une distribution : si la répartition des degrés des noeuds (ou la répartition des énergies ou intensités) suit une **loi de puissance**, on parle de "scale-freeness" ou "invariance d'échelle". Dans un **DSL**, la construction du graphe $\{\omega_{i,j}\}$ par la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

peut alors produire un réseau dont la topologie répond à cette caractéristique "hub-and-spoke", où de gros clusters apparaissent et où la distribution des tailles (ou des degrés) se signale par un comportement de type Pareto. Cette observation reflète l'**auto-similarité** : si, à l'échelle micro, des petits regroupements se forment selon la même règle d'auto-organisation, alors à l'échelle intermédiaire ces regroupements se combinent, puis à l'échelle macro, on retrouve encore le même profil d'agencement.

D'un point de vue **conceptuel**, la fractalité offre un double intérêt. Elle permet d'abord une **comprendre** plus aisée de la hiérarchie formée : au lieu de considérer la structure comme un arbre imposé, on réalise que chaque niveau réutilise la même dynamique $\{\omega_{i,j}\}$ de renforcement et d'inhibition, d'agrégation et de division. Cette itération cyclique de la même loi rend la vision globale plus unifiée, car on sait qu'on applique partout la même "grammaire" d'auto-organisation. On y gagne ensuite en **modélisation** : si les distributions présentent des lois de puissance, on peut raisonnablement s'attendre à trouver des "fat tails" (quelques énormes clusters au milieu d'une myriade de petits) à chaque échelle, sans qu'il soit nécessaire de réinventer un nouveau cadre statistique. Cette redondance d'une seule et même règle à plusieurs paliers explique le caractère fractal et simplifie la lecture d'un ensemble potentiellement très complexe. La fractalité n'est donc pas un simple *concept* : c'est un *levier* pour déployer le DSL de façon homogène du local au global, et c'est un *fenêtre* pour percevoir la distribution multi-niveau comme un tout, où la répétition de formes révèle les lois d'organisation profondes.

6.7.3. Limites et Perspectives

Conditions d'Apparition de la Fractalité.

Bien que l'approche **multi-échelle** du Deep Synergy Learning (DSL) s'applique dans de nombreux domaines (vision multimodale, robotique sensorimotrice, analyse conversationnelle, etc.), la **fractalité** n'est pas systématiquement garantie. Il s'agit d'un cas particulier dans lequel la structure ou la dynamique du **Synergistic Connection Network (SCN)** reproduit à plusieurs échelles une forme d'**auto-similarité**, souvent associée à des distributions en lois de puissance ou à un "effet hub" dans le graphe $\{\omega_{i,j}\}$. Un SCN qui ne verrait jamais se former de gros clusters dominants, ou dont la répartition des degrés se rapprocherait davantage d'une loi gaussienne, présenterait peu ou pas de **fractalité** mesurable.

D'un point de vue plus formel, la **fractalité** requiert que la *même* loi d'**organisation** (agrégation, renforcement, division) aboutisse, à chaque palier, à une structure similaire à celle observée aux autres paliers. Autrement dit, la répartition des liens $\{\omega_{i,j}\}$ ou la distribution de la taille des clusters suit une loi "autosimilaire" lorsqu'on passe du niveau micro au niveau macro. Dans certaines applications, les **conditions** pour qu'apparaissent des lois de puissance ou des distributions scale-free (longues queues) ne sont pas remplies : on peut être en présence d'un phénomène plus linéaire, plus symétrique, ou dominé par d'autres effets (ex. saturation rapide des ω ou homogénéité rigide). Dans ces situations, la fractalité demeure anecdote ou absente.

Il importe donc de distinguer l'architecture multi-niveau du DSL (toujours présente) de la **fractalité** (cas où la structure s'avère *auto-similaire* à divers paliers). Il est parfaitement concevable qu'un **DSL** multi-niveau ne révèle aucune auto-similarité notable si les données ou la dynamique ne contiennent pas de régularité fractale. La fractalité n'est donc pas une conséquence obligatoire de l'algorithme, mais une propriété émergente dans certaines configurations de $\{\omega_{i,j}\}$ ou dans certains flux d'événements (voir 6.6.5.2) qui induisent, par leur "queue lourde" ou leur récurrence, une structure fractale.

Possibilités de Recherche et Perspectives

Étude de la dimension fractale d'un SCN.

Une piste intéressante consiste à étudier la **dimension fractale** du **Synergistic Connection Network**. On peut imaginer calculer une "dimension de boîte" du graphe $\{\omega_{i,j}\}$ à chaque itération ou pour chaque palier (micro, macro). La **stabilité** de cette dimension fractale, ou sa croissance/diminution, indiquerait dans quelle mesure le réseau tend à s'**auto-similariser**. Un SCN dont la **dimension** fractale s'avère stable (par ex. ≈ 1.6) pourrait être interprété comme "parfaitement" organisé en lois de puissance. À l'inverse, un SCN dont la dimension fractale fluctue beaucoup ou tend vers un régime gaussien montrerait l'absence d'un patron fractal durable.

Indicateur de cohérence ou de maturité.

Sur un plan **conceptuel**, la fractalité peut servir d'indicateur de "maturation" du réseau. On peut envisager un "score fractal" $F \in [0,1]$: plus ce score est élevé, plus la **répétition** du même schéma à diverses échelles est manifeste. Cela traduirait qu'on a atteint un état d'**auto-organisation** stable où la structure multi-niveau se reproduit à chaque palier. D'un point de vue pratique, un tel indicateur aiderait à diagnostiquer si le SCN est en phase transitoire (score fractal bas) ou en phase aboutie (score fractal élevé).

Heuristiques favorisant la fractalité.

Certains algorithmes DSL pourraient être **modulés** pour encourager la formation de structures fractales, par exemple en sélectionnant des paramètres η, τ, γ (pour l'inhibition) qui renforcent la tendance à créer quelques hubs majeurs et une multitude de petits clusters, typiques des graphes scale-free. Un tel choix pourrait améliorer la **robustesse** : un réseau fractal résiste en général mieux à la perturbation ou à la panne de certains noeuds, les autres niveaux restant intacts (car la structure globale recopie la même organisation). Dans le cas des flux événementiels (bourse, climat, etc.), mettre en évidence la fractalité signifie gérer plus facilement les extrêmes (événements "catastrophiques") et donc gagner en **résilience**.

Conclusion

La fractalité ne constitue pas un **incontournable** de tout SCN : il s'agit d'une **propriété** émergeant dans certains **contextes**, typiquement lorsque la distribution ou l'organisation des entités suit des **lois de puissance** et que la même dynamique s'exprime autosimilairement. Les apports **multi-niveau** du DSL (flux bottom-up, feedback top-down, agrégation Ψ) autorisent, mais ne garantissent pas, ce mode d'organisation fractal. Il reste du **travail** de recherche pour formaliser davantage les conditions exactes sous lesquelles un SCN devient "fractal", comment mesurer sa **dimension** fractale, et comment, éventuellement, tirer profit d'une **structure** fractale pour améliorer la robustesse ou l'efficacité de l'auto-organisation (heuristiques de paramétrage, stratégies pour favoriser ou contrer la formation de hubs dominants, etc.).

6.7.4. Liens avec les Chapitres Suivants

Chapitre 7 (Optimisations).

Le présent chapitre a mis en avant la puissance du **DSL** (Deep Synergy Learning) dans la gestion multi-niveau (micro→macro) et la possibilité de **fractraliser** la structure d'auto-organisation. Pour rendre cette hiérarchie plus stable et plus efficace, on recourt à des mécanismes d'**optimisation**, tels que le **recuit simulé** (injection de bruit pour sortir des minima locaux) ou l'**inhibition** (terme freinant la croissance ubiquitaire des poids ω). Dans une configuration **multi-niveau**, ces techniques peuvent elles-mêmes se décliner à plusieurs paliers : un recuit "local" au niveau micro, un recuit "global" au palier macro, ou encore des stratégies d'inhibition hiérarchique. Le **chapitre 7** détaillera la

manière dont cette logique se met en place, et montrera comment, lorsqu'on sait qu'une structure fractale se réplique d'un palier à l'autre, on peut optimiser localement, puis répliquer le même schéma au palier macro, entraînant une **réduction** notable de la complexité.

D'un point de vue **mathématique**, on adaptera les paramètres η, τ, γ (taux d'apprentissage, facteur de décroissance, amplitude d'inhibition) en jouant sur la récurrence fractale pour accélérer la convergence ou améliorer la robustesse à la sur-segmentation (trop de clusters) ou à la sous-segmentation (regroupement trop vaste).

Chapitre 8 (Multimodal).

La fractalité prend encore plus de sens lorsque l'on gère des **flux très hétérogènes** (vision, audio, texte, signaux divers) dans un cadre véritablement **multimodal**. Le **chapitre 8** décrira comment le **DSL** s'applique pour **fusionner** des sources de données de nature différente, tout en exploitant des patterns récurrents à diverses résolutions (par exemple, motifs d'objets en vision, motifs de forme sonore en audio, motifs lexicaux en texte). Un système multimodal peut observer des invariants d'échelle si la même logique d'auto-organisation apparaît dans chaque modalité, puis se recopie lors de la fusion inter-modale. Le **DSL** fractal détectera et tirera parti de ces symétries à différents niveaux (micro = détail sensoriel, macro = concept global) avec un gain considérable dans l'interprétation d'environnements complexes.

Chapitre 9 (Temps réel).

Dans la plupart des applications réelles, les données (événements, segments, mesures) arrivent **en flux continu**. La fractalité, déjà introduite dans ce chapitre (6.7.3), suppose une auto-similarité potentielle. En **temps réel**, cela signifie que la **dimension fractale** ou la "signature scale-free" du réseau $\{\omega_{i,j}\}$ peut évoluer au fur et à mesure que de nouvelles entités sont incorporées. Le **chapitre 9** décrira précisément comment adapter la **clusterisation** micro et macro dans un **cadre** en perpétuel changement. Si une structure fractale était stabilisée, l'arrivée de nouveaux segments ou événements peut la perturber, forçant le **DSL** à réorganiser ses liens ω . On peut ainsi mesurer la fractalité au fil du temps (log-log plots, dimension fractale dynamique) ou se fixer un **objectif** de maintenir un certain degré d'auto-similarité, marque de l'équilibre ou de la maturité du SCN.

En somme, les chapitres suivants s'appuieront sur les **contributions** du présent chapitre en approfondissant les techniques d'optimisation (chap. 7), en illustrant les scénarios multimodaux (chap. 8) et en finalisant la perspective "flow" continu (chap. 9). L'**idée** demeure de faire interagir la structure multi-niveau (micro→macro) et la **dynamique** d'auto-organisation (ascendant, descendant), tout en tenant compte des **lois** ou **patterns** fractals, lorsque ceux-ci se manifestent naturellement dans les données.

6.7.5. Vision Globale

La **démarche** du **Deep Synergy Learning (DSL)** ne se limite pas à un **traitement** local de paires d'entités : elle met en avant un **apprentissage multi-échelle** qui, du niveau **micro** (regroupement local) au niveau **macro** (agrégation de clusters plus vastes), structure la **complexité** de manière hiérarchique. Il en résulte une organisation claire de la **synergie** $\{\omega_{i,j}\}$, où chaque palier se construit sur le précédent au lieu de traiter toutes les entités dans un **graphe monolithique**. Cette **stratification** apporte de multiples avantages, qu'il s'agisse de la lisibilité, de l'adaptation ou de la possibilité d'exploiter des propriétés fractales.

La notion de **fractalité** occupe une place centrale dans la **vision globale** du DSL. Il s'agit d'**observer** et d'**exploiter** l'**auto-similarité** potentielle : lorsque les mêmes patterns ou lois d'organisation émergent d'un niveau à l'autre, on parle de processus fractal. Autrement dit, si un phénomène s'instancie à l'échelle micro (quelques entités se renforçant mutuellement) et que la même forme de distribution ou de lien réapparaît à une échelle supérieure (super-nœuds se reliant selon des lois semblables), on reconnaît une **autosimilarité** d'échelle. Dans ce contexte, la fractalité prend la forme d'une **répétition** du même schéma d'auto-organisation, conduisant par exemple à des distributions en lois de puissance ou à des "hubs" dominants dans le graphe.

En pratique, l'**apprentissage multi-échelle** proposé par le DSL s'avère particulièrement puissant pour plusieurs raisons :

459. **Lecture améliorée de la dynamique interne (micro) et des schémas (macro) :**

Le modèle permet de **circonscrire** des petits groupements de données (patchs visuels, tokens textuels, capteurs locaux, transactions ponctuelles) afin d'en extraire des **clusters** cohérents. Simultanément, les **super-nœuds** obtenus aux niveaux supérieurs (macro) font ressortir des “grands schémas” (catégories visuelles, intentions conversationnelles, comportements robotiques, tendances de marché) qui, autrement, seraient noyés dans la masse. Cette complémentarité rend possible une *navigation* entre le micro (analyses très localisées) et le macro (vision plus globale ou sémantique), sans se perdre dans la complexité.

460. **Ordonnancement de la synergie** $\{\omega\}$:

La **hiérarchie** rend plus robuste et plus interprétable la mise à jour des pondérations $\omega_{i,j}$. Au lieu de maintenir un énorme graphe plat où les connexions foisonnent de manière incontrôlée, on regroupe, on agrège, on divise si besoin, et on définit des ponts clairs entre les niveaux. Cela **stabilise** souvent le processus, réduit les illusions de sur-connexion et permet d'**isoler** des incohérences au bon palier (micro ou macro).

461. **Résilience et plasticité** :

Le fait de disposer d'une **organisation** à plusieurs échelles confère au réseau la possibilité de s'**adapter** localement à des bouleversements (nouveaux segments, panne d'un capteur, brusque hausse d'une action, etc.) tout en conservant au niveau **macro** une **cohésion** d'ensemble. Les flux ascendants (bottom-up) et descendants (top-down) s'équilibrivent : si un petit groupe change de structure, cela reste contenu ; si un macro-nœud se scinde, cela reconfigure le palier global sans démanteler toute l'organisation micro.

La **fractalité** s'avère être un **levier** précieux dans cette vision globale. Lorsqu'on constate un patron fractal, cela signifie que les *mêmes* lois d'agrégation ou de distribution se retrouvent d'un niveau (micro) à l'autre (macro). On en tire deux conséquences principales :

- **Exploitation** des invariants d'échelle : on peut analyser un phénomène (visuel, conversationnel, boursier, etc.) à petite échelle et en extrapoler des **lois** semblables à grande échelle. Le DSL se charge, par la règle $\omega(t+1) = \omega(t) + \eta[S - \tau\omega(t)]$, de *conserver* cette structure lorsqu'on assemble les clusters.
- **Compréhension** plus profonde du système : une fractalité marquée indique une **auto-similarité** persistante, susceptible d'être utilisée pour la **prédiction** (p. ex. distribution heavy-tailed) ou pour l'**optimisation** (cibler seulement quelques “hubs” dominants). Si la structure fractale est souhaitable, on peut d'ailleurs concevoir des paramètres ou heuristiques (chap. 7) favorisant sa stabilisation.

En définitive, la **coexistence** d'un apprentissage multi-échelle (ancré dans la philosophie DSL) et d'une possible fractalité (auto-similarité à plusieurs paliers) consolide l'**architecture** : on **voit** mieux la dynamique interne, on **dégage** de grands patterns, on maintient plus aisément la **résilience** face à l'arrivée de nouvelles entités ou aux fluctuations extrêmes. Les sections suivantes (p. ex. chap. 7, 8, 9) approfondiront divers **cas** (optimisations, multimodalité, temps réel) où cette démarche se concrétise, mais la synthèse de ce chapitre demeure : l'**apprentissage multi-niveau** et la **fractalité** ne sont pas juste des éléments décoratifs, mais bien des **piliers** fondateurs de la manière dont le **DSL** s'adapte et se structure dans des contextes complexes ou “scale-free.”

Chapitre 7 : Algorithmes d'Optimisation et Méthodes d'Adaptation Dynamique

Chapitre 7 : Algorithmes d'Optimisation et Méthodes d'Adaptation Dynamique	957
7.1. Introduction	959
7.1.1. Contexte	959
7.1.2. Objectifs	962
7.1.3. Structure du Chapitre	969
7.2. Principes Généraux de l'Optimisation dans le DSL	974
7.2.1. Énergie ou Fonction Potentielle	974
7.2.3. Équilibre entre Complexité et Qualité	984
7.3. Recuit Simulé et Perturbations Stochastiques	990
7.3.1. Recuit Simulé : Fondements	990
7.3.2. Protocole de Température	994
7.3.3. Injection de Bruit Aléatoire	998
7.3.4. Exemples d'Implémentation	1004
7.4. Inhibition Avancée et Contrôle de la Compétition	1009
7.4.1. Inhibition Dynamique	1009
7.4.2. Ajustement Automatique de γ	1012
7.4.3. Méthodes de Seuil Adaptatif	1018
Exemples Numériques	1022
7.5. Méthodes Hybrides et Heuristiques	1024
7.5.1. Sparsification Contrôlée	1024
7.5.2. Approches de Type Genetic Algorithm	1027
7.5.3. Recherche Multi-Début ou Multi-Run	1032
7.6. Adaptation Incrémentale et Apprentissage Continu	1039
7.6.1. Mise à Jour en Ligne	1039
7.6.3. Évasion des Minima Locaux à Long Terme	1048
7.7. Couplage avec des Approches de Renforcement	1055
7.7.1. Gestion d'un Signal de Récompense	1055
7.7.2. Multi-Agents : DSL comme Réseau de Coopération	1058
7.7.3. Exemples de Collaboration RL–DSL	1062

7.8. Comparaison Expérimentale et Paramétrages	1068
7.8.1. Étude Comparative (Sans Recuit, Avec Recuit, etc.)	1068
7.8.2. Impact des Paramètres η, τ, γ	1079
A. Rappel du Mécanisme d’Inhibition	1082
7.8.3. Mesures de Performance (Énergie, Cohésion, Temps de Convergence)	1083
7.9. Études de Cas	1088
7.9.1. Cas de Simulation Numérique.....	1088
7.9.2. Mise en Application Robotique ou Multi-Agent	1093
7.9.3. Scénario Symbolique + Sub-Symbolique	1099
7.10. Conclusion et Ouverture	1105
7.10.1. Récapitulatif du Chapitre.....	1105
7.10.2. Liens vers Chapitres 8 et 9	1108
7.10.3. Perspectives pour l’Optimisation Avancée.....	1112

7.1. Introduction

Dans ce chapitre 7, nous nous intéressons aux **algorithmes d'optimisation** et aux **méthodes d'adaptation dynamique** susceptibles d'améliorer la performance et la robustesse du **SCN** (Synergistic Connection Network). Avant de plonger dans ces approches (recuit simulé, inhibition avancée, apprentissage continu, etc.), il convient de rappeler brièvement comment les **chapitres précédents** ont préparé le terrain : nous avons successivement exploré la **représentation** (chap. 3), la **dynamique** (chap. 4), l'**architecture** SCN (chap. 5), et l'**apprentissage multi-échelle** (chap. 6).

7.1.1. Contexte

7.1.1.1. Rappel des Chapitres Précédents : Représentation (Chap. 3), Dynamique (Chap. 4), Architecture (Chap. 5), Multi-Echelle (Chap. 6)

Le propos du **chapitre 7** s'inscrit dans la continuité des fondations établies aux **chapitres 3 à 6**, qui ont explicitement défini la logique du **Deep Synergy Learning (DSL)**, tant au niveau de la représentation des entités qu'au niveau de la dynamique d'auto-organisation, de l'architecture logicielle et de la gestion multi-niveau. Cette section rappelle brièvement ces éléments, soulignant leurs implications en matière d'**optimisation** et de **paramétrages** pour le DSL.

1. Chapitre 3 : Représentation des Entités d'Information

Le **chapitre 3** avait posé les **bases de la représentation** des entités $\{\mathcal{E}_i\}$ manipulées par le DSL. Sur un plan purement **mathématique**, on peut se situer dans un **espace vectoriel** (embedding dans \mathbb{R}^d), dans un **espace symbolique** (ontologies, règles), ou dans une structure hybride (symbolique-subsymbolique). Le **DSL** se concentre sur la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$:

$$S(\mathcal{E}_i, \mathcal{E}_j)$$

qui, selon la nature des entités, peut provenir d'une **similitude** cosinus, d'une **corrélation** statistique, ou d'un **score** symbolique (ontologie, sémantique). Les chapitres consacrés à la **représentation** ont illustré à quel point la **qualité** (fidélité, expressivité) de ce $\{\mathcal{E}_i\}$ affecte directement la **convergence** ou la **stabilité** de la dynamique DSL. Des représentations trop bruyantes ou mal définies peuvent entraîner un comportement chaotique ou des clusters peu pertinents.

En clair, ce **chapitre 3** mettait l'accent sur l'idée qu'une **bonne** ou **mauvaise** représentation se répercute dans les pondérations $\{\omega_{i,j}\}$. Si la fonction S ne reflète pas la "vraie" similarité ou corrélation, la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ pousse le DSL vers un arrangement peu cohérent.

2. Chapitre 4 : Dynamique d'Auto-Organisation

Le **chapitre 4** s'est focalisé sur la **formule** DSL régissant l'évolution des liaisons $\omega_{i,j}$. Sous sa forme la plus générique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Il décrivait aussi des **variantes** telles que l'inhibition compétitive, la dynamique multiplicative, la saturation ou la division top-down. Conceptuellement, c'est dans ce chapitre qu'ont été introduits les **clusters** émergents, la notion d'attracteurs multiples, et la possibilité pour le réseau de se figer dans des minima locaux ou de connaître des oscillations s'il n'est pas bien paramétré.

On y voit déjà poindre un **besoin** : comment **échapper** aux configurations sous-optimales ? Comment éviter que des clusters se figent trop tôt ? Comment gérer les zones d'incertitude ? Ces questions ouvrent la voie à l'introduction de **techniques** de recuit, d'inhibition avancée, ou d'autres heuristiques stochastiques qui seront traitées dans le **chapitre 7**.

3. Chapitre 5 : Architecture Générale du SCN

Dans le **chapitre 5**, l'accent a été mis sur la **conception** même du Synergistic Connection Network (SCN) à travers une **architecture modulaire** :

- Un **noyau** gérant la matrice $\{\omega_{i,j}\}$,
- Des **modules** dédiés au calcul de la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ (pouvant être divers selon qu'on manipule vecteurs, symboles, probabilités),
- Des **sous-systèmes** chargés de l'inhibition, de l'insertion de bruit (recuit), ou de la réorganisation top-down.

Cette structuration a permis de souligner que, pour **optimiser** ou ajuster la dynamique DSL, il est utile de pouvoir paramétrier chaque module indépendamment : par exemple, on peut injecter un bruit local dans le module “mise à jour ω ”, ou on peut activer un process d'inhibition globale depuis un module superviseur. Autrement dit, le chap. 5 expliquait comment organiser le **logiciel** ou le **système** complet pour qu'il soit apte à recevoir des stratégies d'optimisation localisées (ce qui sera approfondi en chap. 7).

4. Chapitre 6 : Apprentissage Multi-Échelle

Enfin, le **chapitre 6** a mis en évidence le fait que le **DSL** n'opère pas sur un unique plan, mais gère plusieurs **palières** : un **niveau micro** (patches visuels, segments de conversation, actionneurs élémentaires, événements ponctuels) et un **niveau macro** (classes sémantiques, intentions conversationnelles, comportements robot, mégapatterns). Nous avons découvert la **coexistence** de flux bottom-up (les entités micro se combinent pour former de plus gros nœuds) et de flux top-down (le macro-nœud oriente ou scinde les clusters micro).

En toile de fond, on a introduit le **concept** de fractalité, c'est-à-dire l'**auto-similarité** à différents paliers, se traduisant par des distributions en lois de puissance ou des patterns répétés. Sur le plan **optimisation**, une telle structure fractale apporte un éclairage sur la façon dont on peut paramétrier (ou exploiter) la récurrence de motifs d'organisation à chaque échelle. En parallèle, si la multi-échelle est mal calibrée, le DSL peut se retrouver piégé dans des configurations statiques ou divergentes, montrant, là encore, la nécessité d'**heuristiques** correctrices.

Synthèse du Contexte

Au terme de ces **quatre** chapitres :

462.**Représentation** (Chap. 3) : la capacité à **encoder** fidèlement les entités \mathcal{E}_i façonne la **qualité** de la fonction $S(i, j)$, donc de la convergence $\{\omega_{i,j}\}$.

463.**Dynamique** (Chap. 4) : l'équation de mise à jour ω et ses **variantes** (inhibition, multiplicatif) peuvent aboutir à des **clusters** souhaitables... ou se bloquer (minima, oscillations).

464.**Architecture** (Chap. 5) : la **modularité** du SCN autorise l'injection de mécanismes d'**optimisation** (bruit, recuit, inhibition) dans divers sous-systèmes.

465.**Multi-échelle** (Chap. 6) : l'**auto-organisation** s'effectue à plusieurs paliers (micro→macro), avec parfois une **fractalité** rendant le système plus riche, mais plus sensible au paramétrage.

Le chapitre 7 va donc s'emparer de ces **demandes** d'optimisation, introduisant des **algorithmes** pour gérer la stagnation, échapper aux minima locaux, rendre l'auto-organisation plus rapide, plus stable, et plus apte à traiter de grands ensembles de données ou des configurations "scale-free". L'enjeu est d'exposer différentes **méthodes** stochastiques, heuristiques, ou analytiques, pour affiner la dynamique $\omega_{i,j}$ et aboutir à des configurations plus satisfaisantes selon l'objectif (robustesse, vitesse de convergence, lisibilité).

7.1.1.2. Positionnement : ce chapitre se consacre aux approches pour optimiser et adapter la dynamique du SCN (Synergistic Connection Network)

Les chapitres précédents (3 à 6) ont posé les bases théoriques et méthodologiques du **Deep Synergy Learning** (DSL) en précisant la **représentation** des entités (Chap. 3), la **dynamique** d'auto-organisation (Chap. 4), l'**architecture** logicielle et modulaire (Chap. 5), et enfin le fonctionnement **multi-échelle** (Chap. 6). Malgré ces fondations, l'expérience et la théorie montrent qu'un **SCN** (Synergistic Connection Network) peut présenter plusieurs **difficultés** : un risque de **stagnation** dans des minima locaux, des **oscillations** non désirées, ou encore une **surcharge** de liens inutiles. Ce constat amène à un **besoin d'optimisation** et d'**adaptation** plus fine. C'est précisément ce que le **chapitre 7** va aborder les **stratégies** pour perfectionner le comportement du SCN, échapper à des configurations sous-optimales, accélérer la convergence, ou maintenir la robustesse en conditions évolutives.

A. Approches pour Optimiser la Dynamique du SCN

La règle de mise à jour du DSL

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

et ses **variantes** (par ex. inhibition, mise à jour multiplicative) s'apparentent conceptuellement à un **processus** de descente de gradient local (ou de montée selon l'interprétation). Si la structure de la fonction d'énergie ou de coût est non convexe, le SCN peut se **bloquer** dans un **minimum local**, aboutissant à une partition (ou clusterisation) non pertinente ; une sorte de configuration stable, mais sous-optimale. Pour y remédier, on introduit des **méthodes stochastiques** (recuit simulé, bruit contrôlé), des **stratégies** de saut global (algorithmes génétiques, par exemple) ou d'autres **heuristiques** plus spécifiques (inhibition ciblée, resets partiels) pour maintenir la **souplesse** de la dynamique. Le chapitre 7 détaillera ces algorithmes et justifiera comment ils s'intègrent dans l'architecture modulaire du DSL. Même lorsque les minima locaux ne constituent pas un problème majeur, il demeure souhaitable d'**optimiser** la vitesse ou l'efficacité de convergence, surtout lorsque le réseau compte de très nombreuses entités $\{\mathcal{E}_i\}$. On peut alors recourir à l'**inhibition dynamique** pour éviter la prolifération de liens de poids moyen, ou à la **sparsification** (ex. conserver seulement les k plus grandes $\omega_{i,j}$ par noeud). Ces procédés allègent le calcul, limitent l'inflation du graphe, et permettent au SCN d'atteindre plus rapidement un arrangement stable et lisible. Sur le plan **algorithmique**, ce besoin d'efficacité se traduit par l'ajout de routines "post-traitement" (seuil, décroissance additionnelle de liens, merges ou splits systématiques) qui contrôlent la structure globale.

B. Méthodes d'Adaptation Dynamique

L'**apprentissage continu** s'impose dans les situations où le **SCN** n'est pas figé mais soumis à un flot de nouvelles **entités** et d'événements, comme cela se produit dans maints environnements évolutifs (voir chap. 9). Afin de préserver la cohérence et la performance du **DSL**, il est essentiel d'intégrer ces ajouts et changements en flux continu sans repartir d'un réapprentissage exhaustif. On peut ainsi actualiser localement les liens $\omega_{i,j}$ lors de l'arrivée d'un nouveau noeud \mathcal{E}_{n+1} , veillant à éviter qu'une perturbation isolée ne dérègle l'ensemble du réseau. Un petit **recuit local** peut être déclenché si un déséquilibre menace d'émerger, tandis qu'une **inhibition latérale** veille à ne pas surcharger les connexions avec des liens superflus. L'objectif est de maintenir l'équilibre entre la réactivité nécessaire à la création de liens pour les entités fraîchement insérées et la stabilité requise pour ne pas affecter négativement la structure déjà établie.

Le contrôle top-down et feedback (voir section 6.4) renforce encore cette logique. Au palier macro, un **macro-nœud** ou module de plus haut niveau peut valider ou corriger la configuration hiérarchique si une incohérence est détectée. Dans ce cadre, on surveille la **cohésion** globale et on peut déclencher une inhibition sélective si un super-bloc se révèle trop vaste ou hétérogène. On peut aussi activer un recuit plus global, ouvrant la voie à des fluctuations contrôlées permettant de sortir d'un état localement stable mais non optimal. Ainsi, le **DSL** bénéficie d'une double plasticité : localement, on garde la dynamique micro et l'apprentissage continu ; globalement, on s'appuie sur un signal descendant si la cohésion globale se dégrade. Cette **dualité** s'inscrit dans l'architecture modulaire décrite (voir chap. 5) et dans la perspective multi-niveau (voir chap. 6), mais sous un angle “optimisation” qui préserve la fluidité du réseau. On obtient donc un système qui demeure alerte aux perturbations, sait incorporer de nouveaux nœuds et gère la hiérarchie en modulant l'auto-organisation locale, tout en préservant un degré de performance et de robustesse optimal vis-à-vis de l'évolution continue du réseau.

C. Fil Conducteur de ce Chapitre

Au regard du bilan dressé, le **chapitre 7** va :

- **Décrire** des **stratégies** stochastiques ou heuristiques pour **échapper** aux minima locaux :
 - **Recuit simulé** (injection de bruit à une “température” contrôlée),
 - **Inhibition avancée** (pour restreindre les liaisons moyennes, forcer une différenciation plus nette),
 - **Algorithmes** globaux (mélange d'approches génétiques, random restarts, etc.).
- **Proposer** des **mécanismes** pour **améliorer** la dynamique DSL :
 - Ajustement auto-adaptatif des paramètres η, τ ,
 - Règles de “trimming” (couper les liens trop faibles),
 - K-out-of-n liaisons autorisées par nœud (sparsification).
- **Montrer** l'**intégration** de ces méthodes dans un **SCN** fonctionnel, à différents paliers :
 - **Local** (micro) : injection de bruit, inhibition ciblée,
 - **Global** (macro) : scission top-down, feedback contextuel,
 - **Flux continu** : comment insérer des entités en cours de route sans bloquer l'évolution.

Cet exposé servira de pont vers les chapitres **8** (multimodal) et **9** (temps réel), où la gestion du DSL dans des environnements complexes (données hétérogènes, flux en perpétuel renouvellement) requiert ces **outils** d'optimisation et d'adaptation pour maintenir la **pertinence** et la **stabilité** de la structure multi-niveau. En somme, il s'agit là du **noyau** méthodologique pour débrider le DSL, face aux défis posés par les minima locaux, la sur-segmentation, la sur-connexion, ou tout autre frein à une auto-organisation fluide.

7.1.2. Objectifs

Afin de perfectionner la dynamique du **SCN** (Synergistic Connection Network) décrite aux chapitres précédents, ce chapitre se fixe pour **objectifs** de :

Exposer les **stratégies** permettant d'**échapper** aux minima locaux (recuit simulé, heuristiques globales ou locales, etc.).

Montrer comment mettre en œuvre des **mécanismes de compétition** avancée (inhibition dynamique, saturation) pour éviter l'excès de liens moyens ou les oscillations incontrôlées.

Évoquer la thématique de l'**apprentissage continu** et de l'**adaptation en temps réel**, donnant au SCN la capacité de s'ajuster à l'arrivée ou au retrait d'entités (ou de flux de données) sans devoir tout “réapprendre” depuis le début.

7.1.2.1. Exposer les Stratégies pour Échapper aux Minima Locaux (Recuit Simulé, Heuristiques)

Dans la dynamique du **Deep Synergy Learning** (voir chapitre 2.2 pour le cadre général), la mise à jour des pondérations $\omega_{i,j}$ suit en règle générale l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ désigne la **synergie** sub-symbolique entre les entités \mathcal{E}_i et \mathcal{E}_j , η un taux d'apprentissage et τ un coefficient de décroissance (voir section 2.2.2 et chapitre 7.1 pour plus de détails). Cette équation se comporte comme une **descente locale** sur un paysage d'énergie $J(\{\omega_{i,j}\})$. Si ce paysage comprend de multiples vallées ou **minima locaux**, la règle DSL risque de **stagner** dans un puits sous-optimal. Pour échapper à ce phénomène et permettre au *Synergistic Connection Network* (SCN) d'atteindre une configuration de meilleure qualité (clustering plus net, auto-organisation plus pertinente), on recourt à des **stratégies** globales ou stochastiques.

Les sections suivantes décrivent plusieurs approches : le **recuit simulé**, certaines **heuristiques** inspirées des algorithmes génétiques ou des “shakes” contrôlés, et la façon de les **fusionner** avec la mise à jour DSL. Ces techniques visent à créer des **sauts** dans l'espace des pondérations, empêchant le SCN de se figer dans un attracteur local.

A. Recuit Simulé (Simulated Annealing)

Le **recuit simulé** s'inspire des procédés de métallurgie, où un **métal** chauffé se liquéfie et permet à ses molécules de se réorganiser, puis où le refroidissement progressif consolide cette configuration, souvent plus stable ou plus “parfaite” qu'un refroidissement brutal. Dans le contexte DSL, on modifie la mise à jour $\omega_{i,j}(t+1)$ en **injectant** un **terme stochastique**. À la formule classique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

on ajoute une **température** $\sigma(t)$ qui décroît dans le temps, plus un bruit $\xi_{i,j}(t)$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t).$$

La **température** $\sigma(t)$ est élevée au début de l'apprentissage, permettant une exploration large (les $\omega_{i,j}$ peuvent varier de manière importante, sautant par-dessus les barrières d'énergie locales), puis on la fait **décroître** progressivement ($\sigma(t) \rightarrow 0$ quand $t \rightarrow \infty$). Divers choix s'offrent pour $\sigma(t)$:

$$\sigma(t) = \frac{\sigma_0}{\log(t+2)}, \quad \text{ou} \quad \sigma(t) = \alpha^t \quad (\alpha < 1),$$

avec σ_0 un paramètre initial.

Ce **recuit simulé** prévient la stagnation dans un **minimum local** en autorisant des “mouvements ascendants” sur le paysage de coût J . Au début ($\sigma(t)$ grande), les perturbations sont fréquentes, brisant d'éventuels piégeages. Par la suite, la chaleur diminue, et le SCN converge lentement vers une configuration stable, idéalement de qualité supérieure. Le **paramétrage** (vitesse de refroidissement, amplitude du bruit, etc.) se décide souvent de manière empirique. Les avantages sont nets en présence de topologies d'énergie complexes, tandis qu'une température mal calibrée (trop basse trop vite, ou trop élevée trop longtemps) peut rallonger inutilement la convergence ou perturber l'auto-organisation finale.

B. Heuristiques Globales ou Locales (Génétiques, “Shakes”)

Un premier ensemble de **métaheuristiques** mobilisables au-delà du recuit simulé sont les **algorithmes génétiques** (GA). Dans une telle perspective, l'**état** du SCN, c'est-à-dire la configuration globale des pondérations $\{\omega_{i,j}\}$, se mue en un “individu” évoluant au sein d'une population. Une opération de **croisement** revient à échanger partiellement des blocs de matrices ω entre différents individus, tandis qu'une **mutation** consiste à modifier aléatoirement certaines pondérations $\omega_{i,j}$. La fonction de **coût** $J(\{\omega_{i,j}\})$ ou la fonction d'**énergie** y joue le rôle de **fitness**, ce qui permet de sélectionner les meilleures configurations dans un cycle itératif de sélection-croisement-mutation. L'intérêt principal réside dans la capacité d'un GA à franchir des barrières d'énergie ou de coût que de simples descentes locales ne peuvent surmonter. Cependant, cette approche implique un coût de calcul conséquent, car chaque individu doit être évalué, rendant la mise en place d'un GA parfois lourde pour des SCN de grande dimension.

Une seconde famille de procédés, plus légère, consiste à conserver la **dynamique DSL** classique tout en introduisant des “**shakes**” ponctuels. On procède alors à des secousses périodiques ou conditionnelles : par exemple, toutes les k itérations, on ajoute un bruit aléatoire à un ensemble de $\omega_{i,j}$, ou l'on met à zéro un pourcentage de liaisons. Cette intervention “secoue” la configuration courante, l'écartant du minimum local où elle pourrait stagner. Il est possible de conditionner cette manœuvre à l'apparition d'un **plateau** dans la convergence (lorsque la norme $\|\Delta\omega\| \approx 0$ indique que le réseau ne progresse plus). On augmente alors localement le degré de liberté, relançant la recherche d'une organisation plus favorable. Les **shakes** répondent ainsi à l'objectif de “redynamiser” le réseau lorsqu'il est bloqué, sans nécessiter la gestion complète d'une population, comme dans un algorithme génétique, ni la maîtrise continue d'un paramètre de température, comme dans le recuit simulé.

C. Mélange DSL + Perturbations Globales

Il existe une **forme générale** pour intégrer dans la **dynamique DSL** un **terme supplémentaire** qui recouvre un bruit de recuit, une mutation génétique, un “shake” aléatoire, ou tout autre mécanisme global. On écrit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}(i,j) + \Delta_{\text{global}}(i,j),$$

où $\Delta_{\text{DSL}}(i,j)$ se limite à la variation usuelle associée à la **descente** contrôlée par la synergie locale :

$$\Delta_{\text{DSL}}(i,j) = \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Le terme $\Delta_{\text{global}}(i,j)$ incarne quant à lui la **perturbation** choisie. On peut lui donner la forme d'un **bruit stochastique** $\sigma(t) \xi_{i,j}(t)$ pour le **recuit simulé**, d'une “**mutation**” pour un algorithme génétique, ou d'un “shake” appliqué à intervalles réguliers si on souhaite relancer la dynamique en cas de stagnation. L'idée cruciale est de **fusionner** la **descente locale** pilotée par $S(i,j)$ et τ avec la **capacité d'exploration** qu'offre la perturbation globale, laquelle évite les minima locaux et procure une plus grande robustesse.

En guise d'**exemple**, si l'on pratique le **recuit simulé**, on définit $\Delta_{\text{global}}(i,j) = \sigma(t) \xi_{i,j}(t)$. La “température” $\sigma(t)$ décroît au fil des itérations pour réduire progressivement l'amplitude des fluctuations et consolider ainsi la structure émergente. À l'opposé, on peut choisir de n'appliquer qu'une perturbation périodique : toutes les K itérations, on “secoue” aléatoirement 10 % des liaisons pour sortir d'un éventuel état bloqué. Dans chaque cas, la formulation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}(i,j) + \Delta_{\text{global}}(i,j)$$

garantit que la **dynamique** DSL reste au centre de l'**auto-organisation**, tandis que la **perturbation** injecte de l'aléatoire ou de la variabilité pour explorer l'espace des configurations et surmonter les limitations inhérentes à la simple descente locale.

D. Avantages, Limites et Conclusion

L'**introduction** de stratégies **stochastiques** ou **globales** dans le **DSL** (telles que le recuit simulé, les algorithmes génétiques ou les “shakes”) présente plusieurs **avantages** majeurs. D'une part, l'ajout d'une perturbation contrôlée permet de **diminuer** sensiblement le risque de se retrouver piégé dans des **minima locaux** : l'espace des pondérations $\{\omega_{i,j}\}$ demeure ouvert à l'exploration, de sorte qu'un cluster ou un macro-bloc peut se transformer ou se dissoudre s'il s'avère que la configuration courante n'est pas optimale. D'autre part, l'**exploration** de l'espace des partitions ou de la matrice ω s'en trouve renforcée, offrant la possibilité de découvrir de nouveaux arrangements, qu'il s'agisse de “clustering” plus net ou de sous-structures plus expressives. De surcroît, l'on peut graduer l'intensité de ces perturbations (recuit, mutation, shakes) selon la phase du processus : des valeurs plus hautes pour l'exploration initiale, puis un décrément progressif afin de stabiliser la convergence en phase finale.

Cependant, ces méthodes ne sont pas exemptes de **limites**. En premier lieu, la **complexité** de paramétrage peut se révéler élevée : il est nécessaire de régler la “température” dans le recuit, la fréquence ou l'amplitude des shakes, ou bien les taux de mutation et de croisement dans un algorithme génétique. De plus, la **charge de calcul** s'accroît souvent, notamment lorsque l'on manipule une population d'individus $\{\omega\}$ pour un algorithme génétique, chaque individu devant être évalué via une fonction de coût J . Par ailleurs, même avec la présence de bruit, l'évolution peut rester piégée dans des attracteurs complexes si, par exemple, la température décroît trop rapidement (dans le recuit) ou si les mutations s'avèrent trop peu fréquentes.

En **conclusion** (voir 7.1.2.1), l'**injection** de recuit, de mécanismes de “shakes” ou de **métaheuristiques** globales demeure une **composante** précieuse pour un SCN qui navigue dans un **paysage** d'énergie potentiellement riche en minima locaux. La **descente** pilotée par la logique DSL s'enrichit ainsi d'une **capacité d'exploration** supplémentaire, ce qui accroît la qualité de l'**auto-organisation** et évite une stagnation prématuée. Les prochaines sections (7.1.2.2, 7.1.2.3) approfondissent les liens entre ces méthodes et l'**inhibition compétitive** (cf. chap. 2.2.2.2) ou l'apprentissage continu (cf. chap. 9), tout en fournissant des **exemples** pratiques d'implémentation et d'évaluation des performances.

7.1.2.2. Montrer la Compétition Avancée (Inhibition, Saturation Dynamique)

Au-delà du **risque** de minima locaux (section 7.1.2.1), un **Synergistic Connection Network** (SCN) peut connaître d'autres dérives, comme la **prolifération** de liaisons moyennes, la **confusion** entre multiples clusters, ou la **dominance** excessive de quelques liens. Pour y remédier, plusieurs **mécanismes** de régulation existent, regroupés sous le terme de **compétition**. Deux stratégies reviennent souvent : l'**inhibition dynamique** (qui dissuade un noeud de maintenir trop de liens en parallèle) et la **saturation** (clipping) des pondérations (qui empêche quelques liens de devenir incommensurablement plus grands que les autres). L'objectif est de **préserver** la netteté de la structure et d'accélérer la convergence.

A. Inhibition Dynamique

L'idée d'**inhibition** avait déjà été évoquée dans différents chapitres antérieurs (voir par exemple chap. 4.4), et elle prend souvent la forme d'une modification de la mise à jour DSL. Plus précisément, on enrichit la formule

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

en y ajoutant un terme inhibiteur, conduisant à

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Ce terme d'inhibition, noté $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$, induit une compétition latérale : plus un noeud \mathcal{E}_i accumule de liens élevés vers divers \mathcal{E}_k , plus il est “pénalisé” dans sa connexion avec \mathcal{E}_j . Sur le plan conceptuel, il s'agit de forcer un **contraste** dans les pondérations $\omega_{i,j}$, car \mathcal{E}_i est dans l'impossibilité de se relier fortement à tous simultanément. Cette logique renvoie à l'idée de “compétition synaptique” en neurosciences, où un neurone ne peut maintenir qu'un nombre limité de synapses robustes.

La **motivation** derrière cette inhibition est de prévenir la formation d'un état globalement médiocre où un vaste réseau de liaisons "moyennes" domine, sans vrai pôle fort ni séparation claire. En imposant à chaque nœud \mathcal{E}_i de "payer un coût" lorsqu'il multiplie les connexions, on l'incite à investir dans quelques liens prioritaires. Le résultat est un **raffermissement** des connexions réellement soutenues par une synergie $S(i,j)$ élevée, tandis que les liens périphériques sont affaiblis et s'étiolent. Cette approche rehausse la **lisibilité** de la structure, car les clusters apparaissent alors avec un **contraste** plus marqué : on repère vite les liaisons solides et on ne conserve pas un entrelacs dense d'arêtes moyennes.

Dans un contexte multi-niveau ou plus vaste, le **palier** supérieur peut activer une inhibition sélective pour empêcher un super-nœud de s'étendre artificiellement. Cela rejoint la démarche de **division** top-down (voir chap. 6.5.2) : si un bloc devient trop massif et hétéroclite, un signal descendant peut s'attaquer à ses pondérations internes, favorisant la scission future. Le paramètre γ peut par ailleurs varier au fil du temps, selon que l'on recherche une forte sélectivité ou, au contraire, une plus grande tolérance à des liens intermédiaires. Quand le réseau se densifie à l'excès, on augmente γ pour durcir la compétition, forçant ainsi l'affaiblissement de connexions en surabondance. Quand on juge le graphe trop fragmenté, on diminue γ de façon à autoriser un plus grand nombre de liaisons modérées. L'inhibition dynamique constitue donc un instrument essentiel pour régler la **densité** et la **sélectivité** de la matrice ω , évitant les stagnations et assurant un meilleur niveau de **contraste** qui facilite l'**auto-organisation** et la détection de clusters.

B. Saturation Dynamique (Clipping)

La **saturation** (appelée aussi *clipping*) consiste à imposer une borne supérieure ω_{\max} aux pondérations. Concrètement, une fois le calcul de $\omega_{i,j}(t+1)$ effectué, on applique

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Cette contrainte borne empêche toute liaison de croître indéfiniment, quel que soit l'enthousiasme de la synergie $S(i,j)$. Du point de vue **mathématique**, le seuil ω_{\max} agit comme un véritable "cap" sur la valeur prise par $\omega_{i,j}$, de sorte que si la mise à jour $\Delta\omega_{i,j}$ propulse $\omega_{i,j}(t+1)$ au-dessus de ω_{\max} , on la ramène simplement à ω_{\max} .

Le **bénéfice** premier est d'empêcher un ou deux liens exagérément forts de "monopoliser" la structure, ce qui se produirait si une synergie $S(i,j) \approx 1$ se maintenait en permanence et qu'aucun mécanisme ne la contenait. Sans borne supérieure, on risquerait de voir $\omega_{i,j}$ se rapprocher de $1/\tau$ ou plus, au détriment d'autres connexions. En imposant un "cap" à $\omega_{i,j}$, on favorise une **meilleure distribution** des ressources au sein d'un nœud \mathcal{E}_i , car même un lien très pertinent ne dépassera pas ω_{\max} . Sur le plan **algorithmique**, ce clipping empêche par ailleurs l'apparition d'**instabilités** ou la divergence de $\omega_{i,j}$ si l'on combine la formule DSL à un bruit (recuit) élevé.

Dans une **version dynamique**, on autorise ω_{\max} à évoluer au fil des itérations, par exemple en démarrant avec un cap bas (limitant la croissance initiale des pondérations) puis en l'augmentant progressivement si la dynamique le justifie. L'on peut aussi choisir l'option inverse : d'abord laisser les liens s'établir librement pour identifier rapidement les plus forts, puis imposer un seuil ω_{\max} dans un second temps de manière à stabiliser la structure et à éviter qu'un petit groupe de connexions ne devienne excessivement dominant. Cette flexibilité dans le choix et la progression du "cap" rend la saturation dynamique, permettant de s'adapter aux différentes phases de l'**auto-organisation** dans le SCN.

C. Synergie entre Inhibition et Clipping

Il est possible de **combiner** l'inhibition et le clipping au sein d'un **SCN** pour contraindre la distribution des pondérations $\{\omega_{i,j}\}$ à la fois en limitant le nombre de liaisons moyennes et en empêchant l'émergence de quelques liaisons trop dominantes. L'**inhibition** agit sur la composante "somme" au sein d'un même nœud \mathcal{E}_i , conduisant à

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

ce qui constraint un nœud à ne pas maintenir un trop grand nombre de liens d'intensité moyenne. La **saturation** ou clipping, quant à elle, impose un seuil ω_{\max} empêchant toute pondération de croître indéfiniment : après mise à jour, on applique

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Ceci borne la valeur des liaisons les plus fortes, évitant qu'un petit nombre d'entre elles ne devienne hyperdominant. Ainsi, on agit sur deux plans complémentaires (cf. chap. 7.2, 7.3) : on inhibe les liens moyens pour encourager la spécialisation locale, et on limite la croissance des liens forts pour prévenir une hégémonie de quelques connexions. Cette **double contrainte** renforce la structure globale du réseau, assurant un ratio sain de connexions faibles contre fortes et évitant la confusion que pourrait engendrer un enchevêtrement de poids intermédiaires ou, à l'inverse, la monopolisation par un seul lien. Le **SCN** y gagne en **lisibilité** (les clusters s'individualisent mieux) et en **robustesse** (la dynamique DSL ne se laisse pas déborder par des extrêmes).

D. Implications pour la Dynamique

Dans un **SCN** comprenant à la fois l'**inhibition** et le **clipping**, la **stabilité** de la structure de clusters s'en trouve accrue. Les entités, au lieu de développer un grand nombre de liaisons d'intensité moyenne, sont amenées à sélectionner certaines connexions réellement significantes : si les pondérations s'accumulent trop sur un même nœud, l'inhibition réduit la prolifération de ces liaisons moyennes, tandis que la saturation borne les liens extrêmes pour empêcher un unique couple (i, j) d'atteindre une valeur trop démesurée. D'un point de vue mathématique, la formule de base de la dynamique DSL

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta[S(i,j) - \tau \omega_{i,j}(t)]$$

devient

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t)$$

puis est suivi d'un **clipping** :

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Cette combinaison engendre un **réseau** qui se clarifie plus vite, puisqu'un grand nombre de liaisons superflues tombent proches de zéro et qu'aucun lien isolé ne s'envole trop haut. Il apparaît ainsi une amélioration de la **vitesse** d'évolution : on part d'un réseau initialement dense où la répartition des poids peut être confuse et, grâce à l'inhibition et au clipping, on converge rapidement vers quelques clusters dominants et nettement délimités. Sur le plan algorithmique, cette sélectivité accrue facilite la détection des groupes principaux.

En ce qui concerne l'**interaction avec un recuit** ou toute autre heuristique stochastique (voir section 7.1.2.1), l'inhibition et le clipping servent de **régulateurs** pendant la phase "chaude" du processus : même si la température ou le bruit suscitent des variations notables dans les $\omega_{i,j}$, on restreint la prolifération de liaisons moyennes (grâce à l'inhibition) et on empêche l'explosion de quelques liens trop forts (via le cap ω_{\max}). Quand la température chute (phase "froide"), le SCN se stabilise autour de liaisons suffisamment sélectives, tout en étant protégé d'une éventuelle "monopolisation" par un unique sous-groupe.

En conclusion (7.1.2.2), il apparaît que l'**inhibition** et la **saturation** ou clipping constituent deux composantes majeures pour un **SCN** discipliné et facilement lisible. Ces deux mécanismes stimulent la **compétition** entre liens à des degrés complémentaires : l'inhibition chasse les connexions moyennes parasites, le clipping limite l'envolée des liens dominants. Leur paramétrage dynamique (variation adaptative de γ , ajustement de ω_{\max} selon le stade de l'apprentissage) offre une maîtrise plus fine de la **stabilité** et de la **qualité** de l'**auto-organisation**. À l'issue de cette compétition avancée, la suite du chapitre (7.1.2.3) examinera la complémentarité de ces approches avec les techniques d'apprentissage continu (chap. 9) et la modularité multi-niveau (chap. 6), pour renforcer encore la résilience du **DSL** face à la variabilité temporelle ou aux perturbations exogènes.

7.1.2.3. Évoquer l'Apprentissage Continu et l'Adaptation en Temps Réel

Les approches d'**optimisation** (7.1.2.1) et de **compétition avancée** (7.1.2.2) permettent de stabiliser et de réguler un SCN. Cependant, dans la plupart des **contextes** pratiques, le flux de données n'est pas figé : de nouveaux événements ou de nouvelles entités peuvent apparaître, et des conditions peuvent changer au fil du temps. Le **Deep Synergy Learning (DSL)**, pour être complet, doit alors intégrer la **possibilité** d'apprendre **en continu** (incrémental) ou de **s'adapter en temps réel**, sans repasser par un processus d'entraînement intégral. Cette section met en lumière ces mécanismes de **mise à jour incrémentale** et de **feedback** dynamique, qui garantissent l'évolution permanente de la matrice $\{\omega_{i,j}\}$ et la plasticité du réseau.

A. Mise à Jour Incrémentale et Flux de Données

Dans de nombreux SCN, il arrive que de nouvelles entités $\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \dots$ surviennent, ou qu'au contraire certaines soient retirées lorsqu'elles deviennent obsolètes. Sur le plan mathématique, on souhaite alors insérer ou supprimer ces nœuds \mathcal{E}_m sans devoir recomputer l'intégralité de la matrice de pondérations $\{\omega_{i,j}\}$ (de taille $O(n^2)$), ce qui serait coûteux. Lorsque survient une **insertion**, on initialise par exemple $\omega_{(n+1),j}(t)$ à 0 ou à de petites valeurs aléatoires pour tous les j existants. On applique ensuite la mise à jour locale, par exemple sous la forme :

$$\omega_{(n+1),j}(t+1) = \omega_{(n+1),j}(t) + \eta [S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau \omega_{(n+1),j}(t)].$$

Les mécanismes d'**inhibition** et de **clipping** (voir chap. 7.1.2.2) peuvent intervenir à ce stade pour prévenir un gonflement excessif des liens. De la sorte, la nouvelle entité \mathcal{E}_{n+1} prend place dans le réseau sans déclencher une reconfiguration totale. On se limite souvent à un **voisinage** $N(n+1) \subset \{1, \dots, n\}$, c'est-à-dire à un ensemble restreint de nœuds j pour lesquels on estime que la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ pourrait être significative.

Dans le cas d'un **retrait**, on peut au contraire faire décroître progressivement $\omega_{m,j}(t)$ vers zéro pour l'entité \mathcal{E}_m sur quelques itérations, ou bien opérer un effacement direct de ces liens si l'on sait que \mathcal{E}_m est définitivement inutile (capteur en panne, objet supprimé, etc.). Dans tous les cas, la logique **DSL** demeure : on ajuste les pondérations $\omega_{m,j}$ via un procédé local, évitant de relancer toute la boucle d'apprentissage.

Cette méthode illustre un **apprentissage “ouvert”**. Le réseau demeure “vivant” : on ne se limite pas à un bloc d'itérations clos, puis à un arrêt, mais on laisse $\omega(t)$ continuer à évoluer sur un temps potentiellement illimité. De nouveaux événements peuvent ainsi façonner la structure au fil du temps, la synergie $S(i, j)$ évoluant, et les poids $\omega_{i,j}$ s'adaptant en conséquence. Sur le plan conceptuel, on se représente alors la dynamique $\omega(t)$ comme un **processus** incrémental (comparable aux flux de données en chap. 9), ce qui confère au **SCN** un caractère adaptatif permanent et favorise une intégration plus souple des entités nouvelles sans perturber gravement la configuration existante.

B. Adaptation en Temps Réel : Rôle du Flux Top-Down

Dans des applications robotiques, conversationnelles ou de recommandation, le **contexte** et le **but** peuvent se modifier au fil du temps. Un robot peut changer de **mission** (transport, exploration, saisie), un chatbot peut voir apparaître un nouveau sous-thème de discussion. Le **DSL**, muni d'un niveau macro (cf. chap. 6), peut alors réorienter la **synergie** au niveau micro afin de répondre aux impératifs globaux. Pour y parvenir, on introduit un **terme de feedback** $\Delta_{\text{down}}(i, j)$ dans la mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)] + \Delta_{\text{down}}(i, j).$$

Si le macro-nœud (ou super-nœud) détecte qu'un cluster local n'est plus pertinent à la tâche courante, $\Delta_{\text{down}}(i, j)$ peut être négatif afin d'**inhiber** ces liens et forcer la réorganisation. À l'inverse, s'il juge nécessaire de renforcer rapidement une liaison, le terme $\Delta_{\text{down}}(i, j)$ devient positif, accélérant la convergence au niveau micro. Grâce à cette **cohérence** entre paliers, le SCN n'est pas contraint de réapprendre entièrement dès qu'un paramètre global change. Les pondérations $\omega_{i,j}$ évoluent sous l'influence du **feedback** descendant tout en conservant l'auto-organisation locale, donnant un **réseau** flexible, apte aux ajustements continus.

C. Avantages Mathématiques et Opérationnels

L'**apprentissage continu** confère la **continuité** de l'organisation : contrairement à un mode batch/arrêt classique qui fige le réseau après un certain nombre d'itérations, la structure $\{\omega_{i,j}\}$ demeure en évolution permanente, s'adaptant à l'ajout d'entités ou au feedback macro signalant un changement de contexte. Cette souplesse se révèle cruciale dans un environnement dynamique. Dans un paradigme neuronal classique, on exigerait souvent de relancer l'entraînement pour intégrer de nouveaux capteurs ou de nouvelles classes. Ici, l'insertion d'une entité \mathcal{E}_{n+1} se fait à un coût bien moindre car on ne perturbe pas la globalité des liens déjà en place. Les **mécanismes d'optimisation** (7.1.2.1 pour recuit ou heuristiques globales, 7.1.2.2 pour inhibition ou clipping) s'étendent naturellement à ce mode flux : on peut déclencher un “shake” lors de l'arrivée d'un lot important d'entités, ou activer une **inhibition** renforcée si l'afflux de nouveaux nœuds crée une abondance de liens moyens. Ce fonctionnement garantit une **dynamique** stable, même lorsque la taille du réseau augmente ou qu'il se reconfigure vers un nouveau but.

7.1.3. Structure du Chapitre

Après avoir présenté le **contexte** (7.1.1) et les **objectifs** (7.1.2) poursuivis par ce chapitre, nous proposons une organisation en plusieurs sections (7.2 à 7.10), chacune abordant un **volet** particulier des **algorithmes d'optimisation** et des **méthodes d'adaptation dynamique** dans le SCN (Synergistic Connection Network). Cette vue d'ensemble permettra de comprendre l'agencement logique du contenu et d'anticiper les liaisons avec les **chapitres** suivants.

7.1.3.1. Vue d'Ensemble des Sections (7.2 à 7.10)

Le **chapitre 7** a pour objectif principal de présenter un **large éventail** de méthodes et d'**approches** visant à **optimiser** et à **adapter** la dynamique d'un **Synergistic Connection Network (SCN)** dans le cadre du **Deep Synergy Learning (DSL)**. Les sections 7.2 à 7.10 se succèdent de manière progressive, partant des principes généraux de l'optimisation dans un contexte DSL pour aboutir à des démonstrations pratiques (études de cas et comparaisons expérimentales). Cette structuration reflète la diversité des **enjeux** : échapper aux minima locaux, gérer la compétition et la saturation, incrémenter le réseau en continu, fusionner des heuristiques globales, etc. La brève synthèse qui suit clarifie la **finalité** de chaque section et leur rôle dans l'ensemble du chapitre.

1. Section 7.2 : Principes Généraux de l'Optimisation dans le DSL

Dans cette section, on revoit comment la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

peut être comprise dans un **cadre** plus large de minimisation ou de maximisation d'une **fonction d'énergie** $J(\Omega)$. Une analogie est faite avec des procédés de “descente de gradient” locaux, ce qui éclaire les raisons pour lesquelles un SCN peut se retrouver piégé dans des **minima locaux**. On y présentera également la contrainte de complexité $O(n^2)$ (ou plus) lorsque le nombre d'entités n est grand, justifiant la recherche de méthodes pour **réduire** la densité du réseau ou **simplifier** certaines phases de calcul.

2. Section 7.3 : Recuit Simulé et Perturbations Stochastiques

Cette section se focalise sur le **recuit simulé**, un protocole inspiré des modèles physiques où l'on applique une “température” $\sigma(t)$ pour autoriser des “sauts” stochastiques dans la configuration. On décrit en détail comment **injecter** un bruit aléatoire $\sigma(t) \xi_{i,j}(t)$ dans la mise à jour ω , et quels schémas de décroissance de σ (ex. $\alpha^t, \frac{1}{\log(t+2)}$) s'emploient le plus souvent. On présente en outre le **rôle** d'un tel bruit : **libérer** le réseau d'un attracteur local, **élargir** la recherche de configurations et faciliter la **découverte** de partitions plus globalement optimales.

3. Section 7.4 : Inhibition Avancée et Contrôle de la Compétition

Après avoir vu comment **relancer** la dynamique par du bruit (ou d'autres perturbations), on se penche sur la nécessité d'**imposer** une **compétition** dans le SCN, afin d'éviter la prolifération des liens "moyens" ou la domination écrasante de quelques liaisons "hyper-fortes".

- On explique le **principe** de l'inhibition latérale (ou globale), par lequel $\omega_{i,j}$ se voit soustraire un terme en fonction de la somme $\sum_k \omega_{i,k}$.
- On illustre aussi la **saturation** (clipping) qui borne $\omega_{i,j}$ à une valeur ω_{\max} .

L'objectif est de **limiter** l'inflation et de promouvoir une **sélectivité** du réseau (peu de liens forts, beaucoup de liens faibles ou nuls), renforçant la clarté des clusters et la robustesse face à l'excès de connexions.

4. Section 7.5 : Méthodes Hybrides et Heuristiques

Dans cette partie, l'accent se porte sur des **algorithmes** plus variés que le simple recuit, comme les **algorithmes génétiques**, la **sparcification** par k-NN, les **colonies de fourmis**, etc. Le point clé est de **montrer** comment la dynamique DSL peut **coexister** avec de telles heuristiques, chacune d'elles pouvant explorer l'espace des configurations $\{\omega_{i,j}\}$ de manière plus radicale ou plus globale. La section discute aussi de la **cohabitation** des coûts de calcul (répartition de la charge), de la possibilité de "multi-run" pour affiner la solution, et de la souplesse offerte par une **architecture** SCN modulaire (voir chap. 5).

5. Section 7.6 : Adaptation Incrémentale et Apprentissage Continu

Cette section renoue avec l'idée (déjà esquissée en 7.1.2.3) d'**apprentissage continu** et d'**adaptation en flux**. Elle étudie comment, concrètement, on **insère** une nouvelle entité \mathcal{E}_{n+1} dans le réseau sans tout recalculer, comment on "oublie" ou "prune" certains nœuds devenus obsolètes, et comment on peut planifier des **mini-perturbations** (recuit ou inhibition plus forte) à intervalles réguliers pour contrer la stagnation à long terme. Cette logique rend le **DSL** "vivant" et perpétuel, plutôt qu'un simple algorithme batch.

6. Section 7.7 : Couplage avec des Approches de Renforcement

Cette partie montre comment, en plus de la synergie $S(i, j)$ calculée localement, on peut **introduire** un **signal de récompense** global ou extrinsèque, typique de l'apprentissage par renforcement (RL). Ainsi, la **dynamique** DSL se voit modifiée pour tenir compte non seulement des corrélations internes, mais aussi d'un retour externe (ex. réussite d'une tâche robotique, satisfaction utilisateur). On envisage alors un **réseau** où ω s'ajuste à la fois selon les synergies micro et selon une "récompense" fournie par un environnement ou par un module RL multi-agent.

7. Section 7.8 : Comparaison Expérimentale et Paramétrages

Après avoir présenté toutes ces **techniques** (recuit, inhibition, heuristiques globales, apprentissage continu, etc.), il s'avère nécessaire de **comparer** leurs effets en pratique. Cette section propose des scénarios simples ou moyennement complexes pour **évaluer** comment la dynamique DSL de base se comporte face à un DSL enrichi d'un recuit simulé ou d'une inhibition dynamique. Les mesures portent sur la **qualité** (énergie finale, modularité du partitionnement), la **vitesse** (nombre d'itérations pour converger) ou la **robustesse** (résistance aux perturbations). On discute aussi du **paramétrage** (calibrage des températures, de γ , etc.) et de la sensibilité de ces paramètres.

8. Section 7.9 : Études de Cas

Pour **concrétiser** davantage, on passe à plusieurs **démonstrations** ou **mini-applications** :

466. **Simulation** d'un SCN de petite taille (20–50 entités) avec des patterns artificiels, observant la convergence avec/sans recuit.

467. **Exemple** robotique (multi-capteurs, multi-tâches) : on voit comment l'inhibition aide à sélectionner un sous-ensemble de connexions utiles pour chaque scénario.

468. **Cas** symbolique-subsymbolique ou "conceptuel", où le couplage RL + DSL montre l'apport du signal de récompense pour clarifier la structure.

Ces **études de cas** éclairent la **mise en œuvre** concrète des méthodes d'optimisation.

9. Section 7.10 : Conclusion et Ouverture

Enfin, la dernière section **synthétise** les diverses **stratégies** exposées, en pointant leurs avantages et leurs limites. On discute aussi de pistes d'amélioration, tels que :

- Des heuristiques plus fines pour l'inhibition,
- Des mécanismes de recuit multi-phase,
- Des intégrations plus approfondies avec l'apprentissage par renforcement.

On fait **le pont** vers les chapitres suivants (8 et 9) où ces techniques trouvent un terrain d'application privilégié :

- **Multimodal** (Chap. 8), où la diversité de flux (vision, texte, audio) accroît la nécessité d'un SCN robuste aux collisions et conflits,
- **Temps réel** (Chap. 9), où la mise à jour en **flux continu** s'avère cruciale, et où l'adaptation rapide du SCN est de mise.

Rôle de ces Sections

La progression (7.2 → 7.10) illustre la **richesse** des solutions d'**optimisation** et d'**adaptation** disponibles pour le DSL. Chacune des **sections** se concentre sur un pan spécifique : principes d'optimisation, recuit, heuristiques globales, inhibition, apprentissage continu, etc., avant de proposer des **comparaisons** expérimentales et des **cas pratiques**. Cette organisation **unifie** les problèmes soulevés aux chapitres précédents (minima locaux, sur-densité, dynamique lente, etc.) et montre comment y **répondre** grâce à des **algorithmes** et **paramétrages** soigneusement choisis.

En somme, ce **chapitre 7** constitue la **boîte à outils** qui, s'appuyant sur les fondations du DSL, va rendre le **Synergistic Connection Network** plus **efficace**, plus **flexible** et plus **adapté** à des applications réalistes (grande échelle, flux évolutionnaire, environnements changeants).

7.1.3.2. Liens avec Chapitres Futurs (8 : Multimodal, 9 : Évolutions Temps Réel...)

Les **méthodes** d'optimisation, de régulation et d'adaptation décrites dans ce **chapitre 7** ne sont pas de pures techniques **isolées**. Elles constituent en réalité un **socle** transversal qui viendra enrichir des scénarios plus spécifiques, lesquels seront abordés dans les chapitres suivants. Le **chapitre 8** se consacre à la fusion **multimodale**, où coexistent plusieurs modalités (vision, texte, audio, capteurs...), tandis que le **chapitre 9** traite des **évolutions en temps réel** et de l'**apprentissage continu**. Dans l'un et l'autre, les **approches** ici présentées (recuit simulé, heuristiques globales, inhibition et clipping, apprentissage incrémental) trouvent un **terrain** privilégié, renforçant leur utilité et leur portée.

A. Chapitre 8 : DSL Multimodal

Le cadre **multimodal** dans un **Synergistic Connection Network (SCN)** implique la gestion simultanée de plusieurs **modalités** telles que la vision, le texte, l'audio ou d'autres flux de capteurs. Cette diversité de canaux accroît la dimension du réseau, puisque le nombre d'entités, souvent noté n , peut devenir considérable. Il en résulte une structure de taille $O(n^2)$ pour la matrice de pondérations $\omega_{i,j}$.

Du point de vue **mathématique**, on observe que la logique d'**auto-organisation** se complexifie dès lors que chaque couple (i,j) peut être associé à diverses **synergies** selon la modalité considérée. Il peut s'agir de similarité d'images, de corrélation audio, de distances textuelles, voire de correspondances cross-modales. Il faut alors maintenir la cohérence globale de l'ensemble des pondérations ω tout en respectant les mécanismes DSL (renforcement ou inhibition).

Lorsque ces sources multimodales se combinent, il est fréquent qu'elles génèrent un réseau très **dense**. Certains canaux peuvent dominer si la synergie d'un type est souvent plus élevée, conduisant à des liens massivement renforcés dans une modalité, et à une négligence partielle des autres. Les méthodes décrites antérieurement (chapitre 7) sur l'**optimisation** et la **compétition** prennent alors toute leur importance.

Le **recuit simulé**, par exemple, peut aider à franchir les barrières d'énergie, surtout dans un contexte où l'on se retrouve confronté à des minima locaux. L'**inhibition** latérale empêche un nœud de conserver trop de liaisons d'intensité moyenne, forçant une spécialisation par modalité ou un choix sélectif de quelques synergies fortes. Le **clipping** (saturation) borne la valeur de certains liens, évitant ainsi qu'une modalité unique ne monopolise la quasi-totalité des connexions et ne fasse gonfler démesurément les pondérations.

Le résultat escompté est un **réseau multimodal** où la concurrence et la coopération entre canaux s'équilibrivent. Les flux texte, image, audio ou sensoriels ont chacun l'occasion de voir leurs synergies respectées lorsqu'elles sont réellement significatives, mais sans tomber dans l'excès. Un système DSL de ce type est, par essence, **scalable** : il peut accueillir un large volume de données hétérogènes, dès lors que l'on applique convenablement les mécanismes de compétition, d'inhibition et d'heuristiques globales de stabilisation.

Au **chapitre 8**, il sera donc question de voir concrètement comment la fusion multimodale s'effectue dans un SCN, comment la hiérarchie de niveaux (chapitre 6) se raccorde à diverses modalités, et comment les techniques d'optimisation ou de compétition (chapitre 7) se déploient dans un univers où la matrice ω reflète des interactions multiples. Cette stratégie garantit un **DSL** apte à traiter, de manière conjointe, des sources variées (vision, texte, audio) sans perdre en lisibilité ni en performance.

B. Chapitre 9 : Évolutions Temps Réel et Apprentissage Continu

L'**incrémantation permanente** s'avère cruciale dans un **SCN** hiérarchique lorsque le flux de données ou d'entités ne cesse de croître au fil du temps. Le **chapitre 9** s'intéresse à cette notion d'**apprentissage continu**, où le réseau n'est jamais figé : de nouvelles entités $\{\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \dots\}$ apparaissent de manière régulière, tandis que d'autres sont retirées ou que les objectifs du système évoluent. Dans un tel contexte, le SCN fonctionne comme un **réseau vivant** dont les pondérations $\omega_{i,j}$ s'actualisent en continu pour refléter l'évolution des synergies locales ou des changements de mission au niveau macro.

L'**adaptation** s'explique d'abord par la possibilité d'insérer chaque entité \mathcal{E}_{n+1} sans relancer un apprentissage exhaustif : on initialise des valeurs de $\omega_{(n+1),j}$ à un niveau faible, puis on applique la dynamique DSL classique $\Delta\omega = \eta[S - \tau\omega]$. Les mécanismes décrits en section 7.1.2.3, tels que le **feedback descendant** ou la surveillance de la norme d'évolution, facilitent l'intégration progressive de ces entités au palier local. Le but est de limiter l'impact global sur la structure déjà formée tout en autorisant une éventuelle reconfiguration si la nouvelle venue perturbe fortement la synergie globale.

Cette logique s'étend au **contrôle top-down** lorsque le macro-nœud (ou un sous-système de niveau supérieur) identifie la nécessité d'une reconfiguration. Un changement de contexte, comme l'arrivée d'un nouvel objectif pour un groupe d'entités, peut déclencher un **signal** négatif ou positif dans la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau\omega_{i,j}(t)] + \Delta_{\text{down}}(i,j)$. Si ce signal est négatif, il inhibe des liens devenus obsolètes ; s'il est positif, il soutient de nouvelles interconnexions. De cette manière, le SCN préserve un **équilibre** entre la cohérence locale (chap. 7 sur l'optimisation et la compétition) et la cohérence globale (chap. 6 sur la hiérarchie et la logique top-down).

Cette capacité à évoluer **en temps réel** se montre indispensable pour des applications comme la robotique multi-agent, où chaque robot peut changer de mission, ou pour un chatbot, où la conversation dérive progressivement sur de nouveaux sujets. Dans tous ces cas, le DSL se met en synergie avec les **méthodes** telles que le recuit simulé ou le “shake” (voir 7.1.2.1) : en cas de stagnation ou de décalage entre le local et le global, un bruit modéré ou une inhibition plus marquée peuvent être introduits pour réorienter la structure. Il s’agit de garantir une **stabilité** tout en ménageant une **liberté** d’évolution permanente. L’**apprentissage continu** ainsi conçu offre un avantage décisif par rapport aux approches statiques : il évite le re-entraînement systématique du réseau, diminue la complexité de mise à jour et assure que le SCN reste fonctionnel et pertinent face à un environnement changeant.

Conclusion sur les Liens avec Chapitres 8 et 9

Les **outils** développés ici (recuit, heuristiques globales, inhibition, clipping, apprentissage continu) sont destinés à être **intégrés** dans des scénarios plus complexes :

- **Multimodal (chap. 8)** : où les flux hétérogènes (vision, texte, audio, capteurs variés) accroissent la dimension et la nature du SCN. Les techniques d’optimisation et de compétition gardent le réseau gérable et discriminant malgré la profusion de signaux.
- **Évolutions temps réel (chap. 9)** : où des entités nouvelles arrivent en flux, où le contexte se redéfinit à la volée. Les stratégies d’adaptation et d’apprentissage continu permettent au DSL de rester pertinent sans re-entraînement global ni effondrement de la structure.

Par conséquent, ce chapitre 7 se situe au **cœur** de la mise en œuvre du DSL dans des environnements complexes : il donne la **boîte à outils** pour corriger, booster, filtrer, scinder et réorganiser le SCN, afin que les chapitres suivants (8 et 9) puissent exploiter cette puissance dans des **cas** multimodaux et dynamiques, correspondant à nombre d’applications pratiques (IA cognitive, robotique, analyse de flux de données, etc.).

7.2. Principes Généraux de l'Optimisation dans le DSL

Dans la dynamique du **Deep Synergy Learning (DSL)**, la mise à jour des pondérations $\omega_{i,j}(t+1)$ s'effectue localement au fil des itérations. Toutefois, on peut aussi comprendre cette évolution comme un **processus d'optimisation** (ou de recherche d'un minimum) dans l'espace de liaisons $\{\omega_{i,j}\}$. Autrement dit, si l'on définit une **énergie ou fonction potentielle** \mathcal{J} , la règle DSL correspond à une forme de **descente** implicite de \mathcal{J} — à ceci près que la synergie $S(i,j)$ peut elle-même être non linéaire et évoluer au cours du temps.

L'**objectif** de ce paragraphe (7.2) est de poser les **fondations** de ce point de vue “optimisation” :

- Expliquer comment la **notion** de fonction d'énergie $\mathcal{J}(\Omega)$ (développée en Chap. 2.4.3) offre une **lecture** mathématique de la dynamique,
- Montrer le **conflit** potentiel entre **descente locale** (risques de minima locaux) et **recherche globale**,
- Soulever les **contraintes** liées à la **taille** n du réseau et le besoin d'**approximations** pour maintenir un **équilibre** entre complexité de calcul et qualité de la solution.

7.2.1. Énergie ou Fonction Potentielle

Le concept d'**énergie** (ou de **fonction potentielle**) \mathcal{J} joue un rôle central pour relier la **dynamique DSL** à des principes d'optimisation classiques. L'idée est la suivante : on tente de **définir** un critère global $\mathcal{J}(\Omega)$ dont la **minimisation** correspond, au moins en partie, à la **formation** de clusters et à la **cohérence** entre entités.

7.2.1.1. Étude Très Déttaillée : Rappel de la Notion de $\mathcal{J}(\Omega)$ (Référence Chap. 2.4.3)

Le **Deep Synergy Learning (DSL)** se prête à une **formulation** en termes d'**énergie** (ou de **fonction potentielle**) \mathcal{J} . Cette approche, déjà esquissée au **Chap. 2.4.3**, relie la **dynamique** du Synergistic Connection Network (SCN) à des principes d'**optimisation** plus classiques. L'idée centrale consiste à construire un critère $\mathcal{J}(\Omega)$ (où Ω désigne la **matrice** des pondérations $\{\omega_{i,j}\}$) dont la **minimisation** correspond à une configuration de réseau favorisant la **cohérence** (ou les **clusters**) entre entités fortement synergiques, tout en contrôlant la **densité** ou l'**amplitude** des liaisons.

Dans le **Chapitre 2.4.3**, un **prototype** de cette énergie avait été introduit :

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} (S(i,j)) + \mathcal{R}(\Omega).$$

La présente sous-section (7.2.1.1) rappelle et développe ce concept, en explicitant la signification de chaque terme et en étudiant la façon dont la **mise à jour** locale (cf. chapitres sur le DSL) peut être interprétée comme une **descente** (implicite ou approximative) de \mathcal{J} .

Forme Générale de $\mathcal{J}(\Omega)$

On considère un **réseau** de n entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Chacune possédant potentiellement un **poids** $\omega_{i,j}$ avec d'autres entités, la **matrice** $\Omega \in \mathbb{R}^{n \times n}$ (ou un ensemble de $\omega_{i,j}$ si on ne suppose pas nécessairement la symétrie $\omega_{i,j} = \omega_{j,i}$) décrit la configuration du SCN. On se propose de définir une “**énergie**” $\mathcal{J}(\Omega)$ selon deux composantes principales :

$$\mathcal{J}(\Omega) = - \sum_{(i,j)} \omega_{i,j} S(i,j) + \mathcal{R}(\Omega).$$

Le premier terme, $-\sum_{(i,j)} \omega_{i,j} S(i,j)$, encourage des liaisons $\omega_{i,j}$ **fortes** lorsque la **synergie** $S(i,j)$ est élevée. D'un point de vue énergétique, on “récompense” la mise en place d'un **lien** $\omega_{i,j}$ important, si et seulement si $S(i,j)$ est grand ; autrement dit, on abaisse \mathcal{J} (et donc on se rapproche d'un minimum) quand on dote le réseau de fortes connexions

$\omega_{i,j}$ pour les paires (i,j) fortement synergiques. Cette **somme** ne fait qu'accumuler l'**utilité** ou l'**affinité** exprimée par S .

Le second terme, $\mathcal{R}(\Omega)$, constitue une **régularisation**, sans cette composante, on risquerait de voir $\omega_{i,j}$ croître sans entrave, dès lors que $S(i,j) > 0$. Ce **terme** \mathcal{R} peut prendre différentes formes :

$$\mathcal{R}(\Omega) = \lambda_1 \sum_{(i,j)} (\omega_{i,j})^2, \quad (\text{ex. terme quadratique})$$

pour modérer la croissance illimitée, ou bien

$$\mathcal{R}(\Omega) = \lambda_2 \sum_{(i,j)} \omega_{i,j} \quad \text{ou} \quad \lambda_3 \sum_{(i,j)} |\omega_{i,j}|,$$

pour **penaliser** la taille/densité globale du réseau (voir le **Chap. 2.4.3** où des mécanismes de parsimonie apparaissent). On peut aussi introduire un **terme d'inhibition** compétitive (voir **Section 2.2.2.2**), réécrivant \mathcal{R} comme une fonction qui \uparrow si trop de liaisons connectées à un même nœud s'avèrent très fortes.

En posant $\mathcal{J}(\Omega) = -\sum \omega_{i,j} S(i,j) + \mathcal{R}(\Omega)$, on peut considérer la **configuration** Ω qui **minimise** \mathcal{J} . Cela revient à :

$$\min_{\Omega} \mathcal{J}(\Omega) = \min_{\Omega} \left[- \sum_{(i,j)} \omega_{i,j} S(i,j) + \mathcal{R}(\Omega) \right].$$

En l'absence de \mathcal{R} , on chercherait à **maximiser** $\sum \omega_{i,j} S(i,j)$. Si $S(i,j)$ est symétrique et positive, cette maximisation favoriserait la croissance illimitée de $\omega_{i,j}$ pour toutes paires (i,j) où $S(i,j) > 0$. D'où la nécessité d'un terme \mathcal{R} (par exemple $1/2 \tau \sum (\omega_{i,j})^2$) forçant un équilibre. La solution "compromis" (minimiser \mathcal{J}) aboutit alors à la mise en avant des **couples** (i,j) bénéficiant d'une forte synergie, sans laisser les pondérations diverger.

Sous certaines conditions (comme l'existence d'un unique minimum local ou un aspect convexe/quasi-convexe), on peut assimiler la **mise à jour** en DSL à une **descente de gradient** implicite. En effet, la mise à jour additive

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

correspond, en omettant d'autres nuances, à

$$-\frac{\partial}{\partial \omega_{i,j}} (-\omega_{i,j} S(i,j) + 1/2 \tau (\omega_{i,j})^2) = S(i,j) - \tau \omega_{i,j}.$$

Ainsi, la descente de $\nabla \mathcal{J}$ sur ce **terme** est similaire à la **dynamique** DSL pour la pondération $\omega_{i,j}$. Bien entendu, on peut enrichir \mathcal{R} pour refléter des inhibitions, des couplages compétitifs, voire du pruning (Chap. 2.4.3).

Le fait de **minimiser** \mathcal{J} revient donc à favoriser des liaisons $\omega_{i,j}$ hautes pour les paires (i,j) dont $S(i,j)$ l'est aussi, tout en imposant une **penalité** si le réseau devient trop "dense" ou si certaines connexions outrepassent leurs limites. Sur le plan algébrique, ce type de **problème** d'optimisation a souvent pour solution un arrangement en **clusters** ; on regroupe naturellement les entités cohérentes (hauts scores S internes au cluster), et on maintient des connexions plus modestes (voire nulles) avec l'extérieur. D'où la lecture en **partition** ou **sous-réseaux** fortement connectés.

Le **Chap. 2.4.3** exposait déjà l'intérêt de cette **vision** en "énergie" pour expliquer pourquoi la **dynamique** DSL conduit à des structures auto-organisées. L'**attraction** induite par $-\sum \omega_{i,j} S(i,j)$ (qui cherche à rapprocher les entités synergiques) se trouve contrebalancée par la **régularisation** \mathcal{R} , prévenant la croissance illimitée et assurant la parcimonie ou un certain contrôle des amplitudes. Ce cadre "pseudo-énergétique" s'apparente à la **minimisation** d'une **énergie libre** en physique statistique ou au **principe** de couplage local-structure globale en théorie des graphes.

Plus formellement, on peut chercher à **relier** la règle locale de mise à jour à une **équation** $\dot{\omega}_{i,j} = -\partial \mathcal{J} / \partial \omega_{i,j}$ en temps continu, montrant que la convergence d'un SCN correspond à l'obtention d'un état stationnaire au sens d'un **gradient**

nul. La prise en compte des **variantes** (inhibition compétitive, mécanismes multiplicatifs) donne des formes plus complexes de \mathcal{J} . Les **chapitres** suivants (Chap. 7, Chap. 12) approfondiront l'analyse mathématique de la **stabilité** et de la **structure** minimale d'énergie, mettant en évidence comment la clusterisation émerge comme un **minimum local** (ou plusieurs minima) dans l'espace Ω .

Conclusion : Vers un Cadre d'Optimisation Global

En guise de synthèse, la notion de fonction potentielle $\mathcal{J}(\Omega) = -\sum \omega_{i,j} S(i,j) + \mathcal{R}(\Omega)$ offre une **lecture** globale de la **dynamique** locale : rechercher la configuration de Ω qui **minimise** \mathcal{J} consiste à lier fortement les entités dont la synergie S est grande, tout en évitant la prolifération anarchique de liens via la **régularisation**. Les **mises à jour** locales (sections 2.2.2 et 2.2.3) peuvent dès lors se comprendre comme une tentative de **descente** (au moins partielle ou approximative) de ce critère global, conduisant dans de nombreux cas à la formation de **clusters** cohérents. C'est ce socle qu'approfondit le **Chapitre 7**, introduisant davantage de **théorèmes** de convergence, de méthodes d'**optimisation** et de considérations sur la **complexité** algorithmique, confortant ainsi la robustesse du **DSL** en tant que **méthode** d'auto-organisation orientée par une **fonction** pseudo-énergétique.

7.2.1.2. Étude Très Déttaillée : Cas Simple de l'Énergie $\mathcal{J} = -\sum \omega_{i,j} S(i,j) + \tau/2 \sum (\omega_{i,j})^2$

Dans la **Section 7.2.1.1**, on a rappelé comment la notion d'**énergie** (ou **fonction** potentielle) $\mathcal{J}(\Omega)$ vient éclairer la **dynamique** du **Deep Synergy Learning (DSL)**. Le présent sous-chapitre (7.2.1.2) illustre un **cas** classique, en se limitant à une forme quadratique de **régularisation** et à une hypothèse de **synergie** $S(i,j)$ fixée. On montre que la **descente de gradient** sur cette énergie coïncide alors, terme à terme, avec la **règle** de mise à jour en DSL (telle que décrite notamment dans les **Sections 2.2.2 et 4**). Ce résultat clarifie pourquoi le **DSL** peut être considéré, au moins dans ce cadre simplifié, comme une **procédure** d'optimisation implicite conduisant à la formation de **clusters** cohérents dans un Synergistic Connection Network (SCN).

Considérons un **réseau** de n entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. On introduit une **matrice** Ω rassemblant les pondérations $\omega_{i,j}$, et on se donne une **synergie** $S(i,j)$ censée représenter l'affinité (distance inversée, co-information, etc.). Le **cas simple** consiste alors à prendre

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} [\omega_{i,j}]^2,$$

où $\tau > 0$ constitue un **paramètre** de décroissance. L'interprétation est la suivante. Le premier terme $-\sum_{i,j} \omega_{i,j} S(i,j)$ “récompense” le fait d'augmenter $\omega_{i,j}$ quand $S(i,j)$ est grand : en rendant $\omega_{i,j}$ élevé, on **diminue** (négativement) l'énergie. On se rapproche donc du minimum chaque fois qu'on amplifie les liaisons entre entités jugées fortement synergiques. Le second terme, $\tau/2 \sum_{i,j} \omega_{i,j}^2$, **pénalise** la croissance excessive de $\omega_{i,j}$. Il s'apparente à un **régularisateur** quadratique imposant un certain amortissement (cf. **Chapitre 2.4.3** sur les termes de régularisation).

L'équation ci-dessus incarne un **compromis** : on souhaite pousser $\omega_{i,j}$ à être grand si $S(i,j)$ l'est (pour créer un cluster fort), mais on limite l'expansion via $\tau \omega_{i,j}^2$. Il s'ensuit, mathématiquement, un **point d'équilibre** auquel la pondération $\omega_{i,j}$ cesse de grandir (ou décroît), aboutissant à un agencement final de poids reflétant un certain équilibre entre “affinité” et “parcimonie”.

Pour mieux voir que la mise à jour en DSL peut se lire comme une **descente de gradient** sur \mathcal{J} , on dérive \mathcal{J} par rapport à une pondération $\omega_{i,j}$. Le **gradient** s'écrit :

$$\frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = -S(i,j) + \tau \omega_{i,j}.$$

Le premier terme $-S(i,j)$ provient de la dérivée de $-\sum \omega_{i,j} S(i,j)$ ($S(i,j)$ étant supposé **constant** par rapport à $\omega_{i,j}$), et le second $\tau \omega_{i,j}$ vient du terme $\tau/2 \sum \omega_{i,j}^2$.

Dans une **descente de gradient** explicite, on aurait :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}(\omega_{i,j}(t)).$$

En substituant,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta[-S(i,j) + \tau \omega_{i,j}(t)] = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Cette **formule** coïncide exactement avec la **règle** du DSL (voir **Section 4** et **Section 2.2.2.1**). Le facteur η tient lieu de **pas** de la descente de gradient. Ainsi, dans ce cadre restreint (synergie $S(i,j)$ fixe, coût quadratique $\tau/2 \sum \omega^2$), la dynamique DSL s'avère **équivalente** à une descente de gradient standard sur la fonction d'énergie $\mathcal{J} = -\sum \omega S + \tau/2 \sum \omega^2$.

La **minimisation** de \mathcal{J} via cette descente de gradient locale pousse à :

469. **renforcer** $\omega_{i,j}$ si $S(i,j)$ est élevé, pour diminuer $-\sum \omega S$,

470. **brider** la croissance par le terme $\tau/2 \sum \omega^2$.

Cette dynamique conduit naturellement à la **constitution** de sous-groupes d'entités présentant des synergies fortes entre elles et des liaisons moins développées vers l'extérieur, c'est-à-dire à une **formation** de **clusters** (voir aussi **Chap. 4** pour l'analyse de la stabilité).

Bien que ce cas soit **simplifié**, il illustre clairement pourquoi la **règle** DSL est assimilable à une **descente** partielle ou approximative d'une **énergie**. Dans la pratique, il existe de nombreuses variantes plus complexes :

- La **synergie** $S(i,j)$ peut dépendre de $\omega_{i,j}$ lui-même, rendant \mathcal{J} non linéaire ou non stationnaire (voir **Section 7.2.1.3**).
- D'autres **termes** de régularisation, comme l'inhibition compétitive ou les couplages n-aires, rendent \mathcal{J} plus riche et parfois non convexe.
- La **dynamique** DSL peut incorporer un **bruit** stochastique, des coupes de liens $\omega < \omega_{\min}$, ou des mécanismes multiplicatifs. L'**interprétation** en termes d'énergie devient alors plus délicate, souvent on aboutit à une "descente de gradient" combinée à d'autres heuristiques (ex. recuit simulé, heuristiques globales).

Malgré ces limites, le **cas simple** $\mathcal{J} = -\sum \omega_{i,j} S(i,j) + \tau/2 \sum \omega_{i,j}^2$ offre un **modèle** de référence. Il explique mathématiquement l'idée que la **règle** DSL favorise la mise en place de **fortes liaisons** au sein de paires ou de groupes synergiques, tout en évitant l'emballement via le facteur τ . Cette correspondance explicite entre la **règle** de mise à jour et la **descente** de $\nabla \mathcal{J}$ légitime la lecture pseudo-énergétique du **DSL**, où les **clusters** se comprennent comme des **états** d'énergie plus basse.

Conclusion

Lorsque l'on retient la forme $\mathcal{J}(\Omega) = -\sum_{(i,j)} \omega_{i,j} S(i,j) + \tau/2 \sum (\omega_{i,j})^2$ et que $S(i,j)$ demeure **constant** (ou indépendant de $\omega_{i,j}$), la **descente de gradient** sur \mathcal{J} se confond avec la **règle** de mise à jour DSL $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$. Ce cas illustre, de façon particulièrement nette, la **logique** d'équilibre entre "attirer" les entités synergiques (par un terme $-\omega S$) et "contrôler" l'ampleur des poids (par un terme $\tau \omega^2$). Les **variantes** plus réalistes ou plus complexes (synergie dépendante de ω , inhibition compétitive, synergies multiples) prolongent cette **vision** en y introduisant des non-linéarités ou des degrés de liberté supplémentaires, ce qui sera discuté plus avant dans les **Chapitres** à venir (par ex. **7.2.1.3** et **Chap. 12**).

7.2.1.3. Étude Très Déttaillée : Limites liées à la Synergie Évolutive ou aux Couplages Non Linéaires

La **Section 7.2.1.2** a montré comment, dans un cadre simplifié, l'on peut relire la mise à jour du **Deep Synergy Learning (DSL)** comme une **descente** de gradient sur une fonction potentielle $\mathcal{J} = -\sum_{i,j} \omega_{i,j} S(i,j) + \tau/2 \sum \omega_{i,j}^2$.

Cette modélisation clarifie les mécanismes d'**auto-organisation** lorsqu'on suppose le **score** de synergie $S(i,j)$ fixe et le **terme** de régularisation quadratique. La présente section (7.2.1.3) nuance ce tableau en soulignant deux **limitations** majeures : (1) la **synergie** elle-même peut évoluer avec la configuration du réseau ou au fil du temps, et (2) la relation entre ω et \mathcal{J} peut ne plus être simplement quadratique ou additive, ce qui rend la descente de gradient plus complexe. On conclut en replaçant l'exemple $\mathcal{J} = -\sum \omega S + \tau/2 \sum \omega^2$ dans un cadre plus général, justifiant l'emploi de techniques stochastiques ou globales quand les couplages se compliquent.

Dans la **pratique** du DSL, la **synergie** $S(i,j)$ n'est pas forcément **constante** ni indépendante de $\omega_{i,j}$. Il arrive que deux entités \mathcal{E}_i et \mathcal{E}_j développent une synergie plus élevée au fil de leur histoire commune, par exemple si elles ont déjà coopéré dans le passé ou si elles partagent un flux sensoriel grandissant. De même, le **flux** de données peut **évoluer**, modifiant les distances, les similarités ou les co-informations. On obtient ainsi un **SCN** dont l'énergie potentielle $\mathcal{J}(\Omega)$ dépend, d'itération en itération, du contexte : au moment où l'on effectue la dérivée de \mathcal{J} , la **fonction** \mathcal{J} elle-même a pu changer.

Mathématiquement, cela signifie que $\mathcal{J}(\Omega; t)$ ou $\mathcal{J}(\Omega, \Omega)$ n'est plus un **critère** statique. On parle alors d'une **énergie** "instable" ou en tout cas "non stationnaire", se modifiant selon la chronologie du système, ce qui ruine l'analogie directe avec une descente de gradient sur un **paysage** fixe. L'**exemple** simple $\mathcal{J} = -\sum \omega S + \tau/2 \sum \omega^2$ devient partiellement inopérant : si S dépend de la configuration Ω ou d'autres variables internes, la dérivation $\nabla_\omega \mathcal{J}$ se complexifie, imposant des **termes** de rétroaction additionnels.

Dans ces conditions, la **règle** locale $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ n'est plus strictement la descente d'un **gradient** classique, mais un mouvement adaptatif sur un **paysage** lui-même en mouvement. Le système peut néanmoins **auto-organiser** ses liens, au prix d'une lecture plus subtile de la convergence (on peut parvenir à un **régime** dynamique stable plutôt qu'à un minimum statique).

Une autre restriction du schéma $\mathcal{J} = -\sum \omega S + \tau/2 \sum \omega^2$ tient à la **forme** même de la régularisation et de la dépendance. Il existe des versions du DSL où la **relation** entre $\omega_{i,j}$ et le "coût" total \mathcal{J} ne se borne pas à un terme quadratique $\tau/2 \omega_{i,j}^2$. On peut avoir des **couplages** plus complexes :

- **Termes n-aires** : Si la synergie $S(\mathcal{E}_i, \mathcal{E}_j, \mathcal{E}_k)$ intervient (cf. **Chap. 12**), ou si l'on définit des interactions triples ou quadruples, alors \mathcal{J} dépendra de produits $\omega_{i,j} \omega_{j,k} \omega_{k,i}$, rendant la **descente** de gradient plus ardue, parfois non convexe.
- **Inhibitions saturantes** : Certaines **inhibitions** (voir **Section 2.2.2.2**) imposent que la somme $\sum_k \omega_{i,k}(t)$ reste en deçà d'un certain seuil ou qu'un terme $\gamma \sum_k \omega_{i,k}^2$ se rajoute de manière non linéaire. Sur le plan analytique, on ne peut plus écrire $\partial \mathcal{J} / \partial \omega_{i,j} = S(i,j) - \tau \omega_{i,j}$; la dérivée comprend des couplages "croisés" compliqués.
- **Non-séparabilité** : Dans le cas quadratique, on traite $\sum (\omega_{i,j})^2$ comme une somme d'éléments indépendants. Mais si la "régularisation" prenait la forme $\sum (\omega_{i,j} \omega_{i,k})$ (compétition pour un nœud \mathcal{E}_i), la fonction $\mathcal{R}(\Omega)$ ne se séparerait plus en termes de $\omega_{i,j}$ isolés ; on accède alors à un **paysage** encore plus non trivial.

On obtient donc, sous l'angle mathématique, une "**descente de gradient implicite**" sur un **critère** $\mathcal{J}(\Omega)$ qui intègre des **termes** non linéaires d'ordre supérieur ou saturants. Ces systèmes peuvent connaître plusieurs minima locaux, des phasages, voire des comportements oscillatoires. On perd, dès lors, la garantie d'un **maximum** global de $\sum \omega S - \mathcal{R}$ ou d'un **minimum** global pour \mathcal{J} .

Conclusion

Bien que le **cas** $\mathcal{J} = -\sum_{i,j} \omega_{i,j} S(i,j) + \tau/2 \sum_{i,j} \omega_{i,j}^2$ soit très instructif pour saisir la relation entre la **règle** de mise à jour en DSL et une **descente** d'énergie (voir **Section 7.2.1.2**), il ne recouvre pas toute la **richesse** du DSL. Dans de nombreux scénarios pratiques, la **synergie** peut évoluer (dépendre du temps ou de l'histoire), faisant de la fonction d'énergie un **objet** non stationnaire. Par ailleurs, on peut rencontrer des **couplages** non linéaires (termes n-aires, inhibitions complexes), rendant la fonction \mathcal{J} non convexe, parfois non séparable, et conduisant à des **dynamiques** plus difficilement assimilables à une simple descente de gradient.

Cette observation justifie l'usage, dans les **cas réels**, de **techniques** stochastiques ou globales (telles que le recuit simulé, les heuristiques multi-phase, ou même des algorithmes évolutionnaires) pour éviter de se retrouver piégé dans un **attracteur** local si la fonction potentielle s'avère très irrégulière. On préserve néanmoins l'esprit général : la **règle** DSL demeure **localement** assimilable à un mouvement tendant à augmenter $\omega_{i,j}$ quand $S(i,j)$ est fort, et à le diminuer dans le cas contraire, aboutissant à la formation spontanée de **clusters** ou de **groupes** synergiques. Les prochains chapitres examineront la stabilité et la complexité de ces systèmes, rappelant que, dès qu'on introduit des **interactions** ou des inhibitions plus subtiles, la fonction \mathcal{J} se rapproche plus d'un **paysage** non linéaire, réclamant des heuristiques plus sophistiquées pour dévoiler la structure ultime de l'**auto-organisation**.

7.2.2.1. Étude Très Détailée : Descente de Gradient Implicite via $\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}$

Dans le **Deep Synergy Learning (DSL)**, la mise à jour des pondérations $\omega_{i,j}$ s'apparente à une **descente de gradient** locale dans l'espace $\{\omega_{i,j}\}$. Cette perspective provient du fait qu'on peut postuler l'existence d'une **énergie** (ou fonction potentielle) $\mathcal{J}(\Omega)$. Dans le cas le plus simple (tel que décrit en **Section 7.2.1.2**), cette \mathcal{J} prend la forme :

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2,$$

mais dans des situations plus générales, \mathcal{J} peut inclure des termes additionnels ou des dépendances complexes. La **règle** de mise à jour DSL, lorsqu'elle est examinée sous l'angle de la **descente** de $\nabla \mathcal{J}$, se formule de manière implicite comme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}(\Omega(t)).$$

La présente section (7.2.2.1) précise la teneur de cette **descente de gradient implicite** et explique en quoi elle confère au DSL sa structure d'**auto-organisation** locale, tout en exposant les limites liées à un tel schéma purement local (analysées en **7.2.2.2** et **7.2.2.3**).

Supposons qu'on ait une énergie $\mathcal{J}(\Omega)$ sur l'espace $\{\omega_{i,j}\}_{(i,j)}$. Si on recherche une **minimisation** de \mathcal{J} par une descente de gradient explicite, on écrit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}(\Omega(t)),$$

où $\eta > 0$ joue le rôle de **taux d'apprentissage** (ou "pas" de gradient). Dans cette optique, la composante

$$-\frac{\partial \mathcal{J}}{\partial \omega_{i,j}}$$

indique la direction de plus forte **descendante** dans l'espace $\{\omega_{i,j}\}$.

Dans le **DSL**, on constate que la **règle** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

peut se réécrire :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta [-S(i,j) + \tau \omega_{i,j}(t)].$$

Cela coïncide exactement avec :

$$-\eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}} \quad \text{si} \quad \frac{\partial \mathcal{J}}{\partial \omega_{i,j}} = \tau \omega_{i,j}(t) - S(i,j).$$

Cet exemple fut détaillé en **7.2.1.2** pour le **cas quadratique**. Plus généralement, dès qu'on pose :

$$\mathcal{J}(\Omega) = -\sum \omega_{i,j} S(i,j) + (\text{termes de régularisation sur } \omega_{i,j}),$$

on aboutit à la **même** structure de mise à jour, la dérivée partielle $\partial \mathcal{J} / \partial \omega_{i,j}$ rassemblant un “terme” $-S(i,j)$ (qui pousse à augmenter $\omega_{i,j}$) et un “terme” provenant de la régularisation (qui agit comme **frein**).

Cette formulation indique que la **mise à jour** en DSL est **locale** : chaque $\omega_{i,j}$ se modifie en fonction de son gradient $\partial \mathcal{J} / \partial \omega_{i,j}$. Dans un système de grande taille $(i,j) \in \{1, \dots, n\}^2$, on applique simultanément la règle, ce qui revient à une **descente** parallèle sur toutes les composantes. Cela crée la notion d'**auto-organisation** : aucun label ou feedback global n'est nécessaire, tout repose sur l'estimation locale de $\partial \mathcal{J} / \partial \omega_{i,j}$.

Cependant, cette **descente** reste “implicite” : la synergie $S(i,j)$ est généralement introduite comme une **constante** (ou dépendant faiblement de ω). Dans la pratique, si $S(i,j)$ varie en fonction du temps ou de l'historique (voir **7.2.1.3**), le **paysage** $\mathcal{J}(\Omega)$ se reconfigure simultanément ; on ne suit plus exactement la descente d'un **paysage** figé, mais d'un paysage qui peut bouger (ce qui n'invalidé pas la logique, mais la rend plus dynamique).

On peut schématiser la règle DSL comme un **schéma** :

$$\Delta \omega_{i,j}(t) = -\eta \nabla_{\omega_{i,j}} \mathcal{J}(\Omega(t)) \Rightarrow \omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta \omega_{i,j}(t).$$

Ce schéma exprime le fait que la variation $\Delta \omega_{i,j}$ est alignée sur l'opposé de la dérivée partielle de \mathcal{J} à l'instant t . Autrement dit, on descend localement la fonction \mathcal{J} . Dès lors qu'un poids $\omega_{i,j}$ se trouve trop grand par rapport à la synergie (au sens du terme de régularisation), la dérivée $\tau \omega_{i,j} - S(i,j)$ devient positive, poussant $\omega_{i,j}$ à décroître. À l'inverse, quand la synergie dépasse la portion de pénalisation, la dérivée s'avère négative, et $\omega_{i,j}$ monte. Ce “jeu” local conduit naturellement à la **formation** de clusters : des liaisons $\omega_{i,j}$ deviennent plus importantes lorsque $S(i,j)$ l'exige, et d'autres se réduisent pour respecter la contrainte d'amortissement.

Le **principal** mérite de cette vision par descente de gradient est son **caractère** très simple et local. Chaque lien applique la même formule, et la structuration en sous-groupes (clusters) apparaît comme l'issue naturelle d'une minimisation. On obtient alors un algorithme $O(n^2)$ qui, step après step, régule l'ensemble des pondérations sans nécessiter de label global. C'est la clé de l'**auto-organisation** : l'émergence d'une structure de réseau à partir d'une simple consigne “descendre \mathcal{J} ”.

En **revanche**, cette descente de gradient locale n'évite pas les **pièges** de minima locaux multiples ou d'attracteurs variés, comme discuté en **7.2.2.2**. On peut ainsi tomber sur une configuration stable mais sous-optimale. De surcroît, si la synergie dépend de Ω elle-même, on ne suit plus un **paysage** stationnaire : l'approximation “descente implicite” reste valable localement, mais la topologie de \mathcal{J} se modifie au fil du temps.

Conclusion

L'identification de la **règle** DSL

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}(\Omega(t))$$

à une **descente** de gradient (implicite) sur la fonction $\mathcal{J}(\Omega)$ confirme le **caractère** local et les **fondements** d'optimisation du DSL. Chacune des liaisons $\omega_{i,j}$ se met à jour en fonction du **gradient** partiel, offrant une mécanique d'**auto-organisation** décentralisée. Cette force fait aussi la **faiblesse** de la descente locale : dans un espace de grande dimension, peu ou pas de mécanismes permettent d'éviter de tomber dans un **minimum local**. Les sections **7.2.2.2** et **7.2.2.3** discuteront les conséquences de ces minima multiples et la nécessité de méthodes plus **globales** (recuit, algorithmes évolutifs, etc.) pour explorer l'espace $\{\omega_{i,j}\}$ au-delà de la simple dynamique de gradient.

7.2.2.2. Risque de Minima Locaux et Attracteurs Multiples

Lorsque l'on assimile la **dynamique du Synergistic Connection Network (SCN)** à une **descente** d'une fonction d'**énergie** $\mathcal{J}(\Omega)$ (voir la Section 7.2.1), on se heurte à un **phénomène** classique de toute descente de gradient locale : l'existence possible de **minima locaux** ou de **multiples attracteurs** dans l'espace des pondérations $\{\omega_{i,j}\}$. Malgré la convergence du réseau vers un **point fixe**, rien ne garantit qu'il s'agisse du **minimum global** : on peut aboutir à un attracteur localement stable, mais sous-optimal sur le plan global. La présente section (7.2.2.2) analyse cette contrainte et illustre pourquoi, en pratique, la **descente** pure peut “emprisonner” le SCN dans une configuration moyenne au lieu de la disposition la plus favorable.

A. Minima Locaux : Définition et Manifestations

Dans les systèmes de grande dimension ou présentant des **couplages** non linéaires, la fonction $\mathcal{J}(\Omega)$ se révèle souvent **non convexe**. Cela signifie qu'elle peut comporter divers “puits” d'énergie, appelés **vallées** ou “bassins” dans le **paysage**. Un **minimum local** est alors un point Ω^* tel que :

$$\nabla \mathcal{J}(\Omega^*) = 0 \quad \text{et} \quad \mathcal{J}(\Omega^*) > \mathcal{J}(\Omega^{(\text{global min})}).$$

En d'autres termes, on ne peut plus abaisser \mathcal{J} en effectuant un petit pas local autour de Ω^* , mais il existe malgré tout un **point** $\Omega^{(\text{global min})}$ où l'énergie est plus faible (plus “profitable”).

Ce phénomène se traduit **concrètement** dans un **SCN** par un agencement de liens $\{\omega_{i,j}\}$ qui semble “cohérent” mais qui, en réalité, n'est pas l'agencement **optimal** au regard d'un critère global (maximisation de la somme $\sum_{i,j} \omega_{i,j} S(i,j)$, prise en compte d'une régularisation, etc.). C'est ce que l'on appelle un **enfermement** dans un attracteur local : la **descente** de gradient locale (ou la mise à jour DSL) ne permet plus de franchir la “barrière” d'énergie séparant ce minimum local d'un état plus satisfaisant.

Exemple simple : supposons qu'il existe deux configurations de **clusters** presque équivalentes. Le système, selon son **initialisation** ou des perturbations précoces, favorise la formation du cluster A plutôt que du cluster B, se stabilisant en conséquence. Si la dynamique du DSL n'injecte aucun mécanisme pour “quitter” cette vallée locale, le réseau restera enfermé dans cette **solution** (qui peut ne pas être la meilleure).

B. Multiplicité d'Attracteurs

Au-delà des minima locaux, un **SCN** de forte dimension peut afficher **plusieurs** attracteurs, chacun correspondant à un arrangement stable de poids $\{\omega_{i,j}\}$. Un **attracteur** se définit comme un état (ou petit voisinage) vers lequel la **descente** locale aboutit si l'on part d'une région déterminée de l'espace :

$$\mathcal{A}_k = \{\Omega(0) \mid \lim_{t \rightarrow \infty} \Omega(t) = \Omega^{(k)}\},$$

où $\Omega^{(k)}$ est le point fixe (minimum local) correspondant. Chaque \mathcal{A}_k est appelé **bassin** d'attraction de $\Omega^{(k)}$. Cela signifie que le **résultat** final dépend **fortement** de la condition initiale $\Omega(0)$.

D'un point de vue **clusters**, deux attracteurs différents peuvent représenter deux différents “clustering” possibles, peut-être d'égale qualité ou d'inégal niveau d'énergie. Par exemple, si le **paysage** \mathcal{J} recèle plusieurs **vallées** quasi symétriques, le système peut basculer dans l'une ou l'autre selon de faibles écarts initiaux. On se retrouve alors avec un **SCN** stable, mais pas forcément unique, soulignant la **fragilité** (ou la **richesse**) de la descente DSL.

C. Illustration Mathématique Minimale

Pour fixer les idées, supposons un micro-réseau de 3 entités {1,2,3}. On définit des synergies hypothétiques $S(1,2), S(2,3), S(1,3)$ et un paramètre τ . Il n'est pas rare de pouvoir configurer ces S de façon à créer deux configurations stables de $\{\omega_{1,2}, \omega_{2,3}, \omega_{1,3}\}$. Par exemple :

- **Configuration A :**

$\omega_{1,2}$ solide, $\omega_{2,3}$ moyen, $\omega_{1,3}$ quasi nul.

- **Configuration B :**

$\omega_{1,3}$ solide, $\omega_{2,3}$ moyen, $\omega_{1,2}$ quasi nul.

Imaginons que l'**énergie** \mathcal{J} attachée à ces deux configurations ne diffère que de Δ . Si l'on part avec $\omega_{1,2}(0) > \omega_{1,3}(0)$, la dynamique aura tôt fait de “pousser” $\omega_{1,2}$ vers un lien fort, par un effet boule de neige, menant à la configuration A. À l'inverse, si $\omega_{1,3}(0)$ devance $\omega_{1,2}(0)$, on finit dans la configuration B. Chaque situation représente un **minimum local** où le SCN se fige, illustrant concrètement la **coexistence** d'attracteurs multiples.

D. Nécessité d'une Recherche Globale

La **descente locale** (au sens strict) ne peut franchir la barrière séparant un minimum local d'une configuration plus profitable. D'autres **méthodes** doivent être introduites pour :

471. **Secouer** la configuration (stochastiquement, recuit simulé, etc.),
472. **Explorer** plusieurs initialisations (multi-run),
473. **Combiner** DSL avec des heuristiques globales (colonies de fourmis, algorithmes génétiques) qui “tournent autour” de différents attracteurs.

Cette problématique, décrite en détail en **7.2.2.3**, est courante dans l'**optimisation** non convexe : on dispose d'une dynamique locale efficace, mais on veut aussi s'assurer d'explorer plus largement l'espace. Le DSL, sans adjonction de bruit ou de contrôle multi-run, risque de se “bloquer” dans une configuration correcte mais non optimale.

Conclusion

Le **risque** de minima locaux et d'attracteurs multiples, inhérent à la descente de gradient dans un paysage **non convexe**, s'applique pleinement au DSL. Même si le SCN converge vers un certain **arrangement** stable de poids $\{\omega_{i,j}\}$, cette solution peut ne pas être l'**optimum** global en termes d'énergie \mathcal{J} et donc ne pas correspondre au **clustering** le plus pertinent. La **section** suivante (7.2.2.3) abordera les **méthodes** stochastiques et heuristiques globales susceptibles de surmonter ces limitations, en permettant au SCN de s'**évader** d'un attracteur local et d'explorer davantage l'espace de solutions.

7.2.2.3. Étude Très Détallée : Motivation pour des Méthodes Stochastiques ou Heuristiques Globales

Les Sections **7.2.2.1** et **7.2.2.2** ont mis en évidence la **descente locale** effectuée par le **Deep Synergy Learning** (DSL) et le **risque** qui en découle : la dynamique peut se retrouver confinée dans un **minimum local** ou un **attracteur** spécifique, au lieu de parvenir à une configuration globalement plus avantageuse. Cette section (7.2.2.3) explique pourquoi des **méthodes stochastiques ou heuristiques globales** (ex. recuit simulé, algorithmes évolutionnaires, etc.) s'avèrent essentielles pour **franchir** les barrières d'énergie et **explorer** plus en profondeur l'espace des solutions $\{\omega_{i,j}\}$. L'objectif est d'éviter de se figer dans un attracteur médiocre et d'espérer atteindre (ou se rapprocher de) des configurations de **meilleure** qualité.

A. Caractéristiques des Méthodes Stochastiques ou Heuristiques

Les **méthodes** stochastiques/globales se distinguent par leur capacité à réaliser des “**sauts**” non purement déterministes et à **explorer** plus largement l'espace $\{\omega_{i,j}\}$.

474. Sauts Non Localisés

Les algorithmes de **descente** (gradient ou variations) ne s'autorisent que de petits pas dans la direction locale de $-\nabla\mathcal{J}(\Omega)$. Les méthodes globales, au contraire, incluent :

- **Bruit** injecté dans la variation $\Delta\omega_{i,j}$ (p. ex. recuit simulé),

- **Opérateurs** “mutation/crossover” (algorithmes génétiques), ou “déplacement aléatoire” (méthodes multi-démarrage),
- **Perturbations** structurées (colonies de fourmis, swarm intelligence) pour échantillonner des régions éloignées du paysage.

Cette démarche évite que la configuration reste prisonnière d'une “vallée” proche de son point de départ.

475. Franchissement de Barrières

Sur le plan **énergétique**, un **minimum local** se sépare souvent d'autres minima par une “barrière” plus ou moins élevée. Dans une descente locale, on ne peut franchir cette barrière, car toute perturbation de faible amplitude reste dans le même bassin. Les **méthodes** stochastiques (bruit, “température” de recuit) autorisent au réseau un certain nombre de “tentatives” de franchissement. *Mathématiquement*, même si $\nabla J(\Omega)$ annule la progression en un point, un “coup” aléatoire suffisant peut pousser Ω en dehors du puits, d'où une chance d'atteindre un minimum plus profond.

476. Recherche Multi-run

Une autre tactique consiste à **redémarrer** la descente DSL de multiples fois depuis des **conditions initiales** distinctes, ou à gérer en parallèle plusieurs “populations” de solutions (comme dans les algorithmes génétiques). Cela **multiplie** les opportunités de contourner des vallées locales différentes.

Sur le plan pratique, on compare les configurations finales obtenues par chaque run et on sélectionne la plus satisfaisante selon $J(\Omega)$. On peut combiner ce multi-run avec du **bruit** pour que chaque descente ait un chemin différent.

B. Recuit Simulé, Inhibition Dynamique et Approches Hybrides

Parmi les stratégies destinées à **globaliser** la recherche, on retrouve notamment :

477. Recuit Simulé

L'idée (détaillée en **Chap. 7.3**) consiste à assimiler la configuration Ω à un “état” physique, et la fonction J à une “énergie”. Au début, on impose une **température** élevée, permettant des **movements** aléatoires de grande amplitude (p. ex. ajout d'un bruit $\sigma(t) \xi_{i,j}$ sur $\omega_{i,j}$), de sorte que la dynamique DSL ne se bloque pas trop tôt dans un puits local. Puis on **refroidit** progressivement ($\sigma(t) \rightarrow 0$), réduisant les sauts pour stabiliser la solution dans un minimum qu'on espère plus global. *Mathématiquement*, cela se traduit par un terme additionnel dans l'évolution $\Delta\omega_{i,j}$, typiquement :

$$\Delta\omega_{i,j}(t) = -\eta \frac{\partial J}{\partial \omega_{i,j}}(\Omega(t)) + \sigma(t) \xi_{i,j}(t),$$

où $\sigma(t)$ décroît en fonction de t et $\xi_{i,j}(t)$ est un bruit (souvent gaussien).

478. Inhibition Dynamique Avancée

Bien que l'**inhibition** vise d'abord à **sparsifier** ou rendre plus sélective la structure de liens (cf. §7.1.2.2), elle peut aussi aider à “débloquer” le réseau si trop de liaisons se maintiennent à des niveaux intermédiaires stables. Une **augmentation** du paramètre d'inhibition γ (par ex. $\gamma \sum_k \omega_{i,k}$) incite certaines pondérations concurrentes à décroître, ce qui peut briser un attracteur local où tous les liens sont moyennement stables.

Algorithmique : À intervalles réguliers, on peut rehausser γ pour forcer des réaffectations de poids plus radicales, puis réduire γ pour regagner la stabilité.

479. Heuristiques Globales (Colonies de Fourmis, Algorithmes Génétiques, etc.)

Section 7.5 évoquera des méthodes comme les algorithmes évolutionnaires, où l'on manipule plusieurs “matrices” Ω simultanément (“population de solutions”), leur appliquant des opérateurs de **mutation** (changement aléatoire de certains $\omega_{i,j}$), de **croisement** (combinaison partielle de deux solutions), et de **sélection** (on ne garde que les solutions présentant la meilleure \mathcal{J}).

D'un point de vue mathématique, on n'est plus limité à la descente locale ; on “sauter” régulièrement dans l'espace, évitant l'enfermement dans un attracteur unique. On peut ainsi échantillonner différentes configurations, détecter lesquelles sont globalement plus prometteuses et favoriser leur reproduction.

C. Pourquoi et Quand les Employer ?

Ces outils de **recherche globale** s'imposent particulièrement lorsque :

480. Le Paysage \mathcal{J} est Très Non Convexe

Chaque **synergie** $S(i,j)$ peut dépendre de variables complexes (interactions n-aires, etc.). L'espace $\{\omega_{i,j}\}$ atteint des dimensions élevées, multipliant les **basins** d'attraction. Un simple gradient “myope” risque alors de s'arrêter tôt, manquant le “vrai” optimum.

481. Robustesse Nécessaire

Dans des applications (vision multimodale, robotique, clustering d'entités hétérogènes) où la **solution** n'est pas unique ou où les données sont bruyantes, on veut s'assurer de trouver un agencement relativement stable, peu sensible aux conditions initiales. Les approches stochastiques renforcent la **robustesse** : elles peuvent secouer le système pour le rendre moins dépendant de perturbations initiales.

482. Apprentissage Continu ou Évolutif

Dès que de nouvelles entités (ou données) surgissent (cf. **Chap. 7.6**), le SCN peut se révéler bloqué dans un ancien attracteur. Injecter du **bruit**, déclencher un **mini-recuit** ou recourir à des heuristiques globales permet de “reconstruire” partiellement les liaisons pour prendre en compte les entités ou synergies inédites, sans repartir de zéro.

D. Conclusion sur la Recherche Globale

Le **DSL** “pur” repose sur une dynamique “locale” (la descente de gradient sur \mathcal{J}). Cette localité ouvre la voie aux **minima** locaux ou aux attracteurs multiples (voir **Section 7.2.2**). Les **méthodes** stochastiques (recuit simulé, injection de bruit) et les **heuristiques** globales (algorithmes génétiques, colonies de fourmis, etc.) constituent une **extension** incontournable pour :

483. **Élargir** la recherche de solutions dans l'espace $\{\omega_{i,j}\}$,

484. **Franchir** les barrières d'énergie qui empêchent la descente locale d'accéder à un bassin plus profitable,

485. **Gagner** en flexibilité et résilience (scénarios multi-démarrage, adaptation aux changements de synergie, etc.).

Les chapitres ultérieurs (en particulier **7.3** et **7.5**) détailleront ces **approches** (recuit, heuristiques) et montreront comment elles s'intègrent dans le SCN pour améliorer la qualité des solutions. Ainsi, on complète la **descente DSL** par des “mouvements” plus globaux, élevés ou aléatoires, gage d'une auto-organisation moins vulnérable aux puits locaux et plus susceptible de dégager une **structure** globale plus satisfaisante.

7.2.3. Équilibre entre Complexité et Qualité

Au fil des sections précédentes (7.2.1, 7.2.2), nous avons souligné d'une part l'**intérêt** de concevoir la dynamique du SCN (Synergistic Connection Network) comme une forme de descente d'énergie, et d'autre part la **nécessité** d'employer des approches plus globales ou stochastiques pour échapper aux minima locaux. Il reste néanmoins une

autre question cruciale : dans un système où le **nombre** d'entités n peut devenir très élevé, comment gérer la **complexité** algébrique ou algorithmique tout en **préservant** la qualité des liaisons $\omega_{i,j}$?

C'est l'objet de la section 7.2.3 : comprendre l'**équilibre** à trouver entre la **complexité** du calcul (ou sa durée) et la **qualité** obtenue pour la structure du SCN. Nous verrons qu'en l'absence de stratégies d'approximation, un SCN complet impose d'évaluer $O(n^2)$ synergies, ce qui devient prohibitif pour de grands n . Des **compris** et des **pratiques** d'approximation (k -NN, ϵ -radius, etc.) se révèlent alors indispensables pour atteindre des solutions "suffisamment bonnes" en un temps raisonnable.

7.2.3.1. Étude Mathématique Détailée : Coût Computationnel lorsque n Entités, $O(n^2)$ Synergies

Dans un **Synergistic Connection Network** hiérarchique, on examine la situation où chaque entité \mathcal{E}_i , pour $i \in \{1, \dots, n\}$, possède la possibilité de former un lien avec toute autre entité \mathcal{E}_j . Les poids de ces liaisons sont notés $\omega_{i,j}$ et évoluent selon la dynamique du **Deep Synergy Learning** (DSL). Le nombre total de paires (i,j) grimpe alors en $O(n^2)$. La présente section (7.2.3.1) offre un exposé analytique de la manière dont cette croissance quadratique influe sur le **coût** en temps de calcul et en mémoire, et montre pourquoi cette "densité" constitue un enjeu majeur lorsque n devient grand.

A. Le Problème d'Échelle : $O(n^2)$ Liaisons.

Dans un graphe complet où tous les couples (i,j) peuvent potentiellement être connectés, la **composante** $\omega_{i,j}$ s'évalue et se met à jour à chaque itération. D'un point de vue **combinatoire**, cela signale $n(n - 1)/2 \approx O(n^2)$ liaisons. Dans la formulation DSL standard, la règle de mise à jour

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

s'applique, *a priori*, à toutes les paires (i,j) . À chaque pas temporel $t \mapsto t + 1$, il faut donc itérer sur $O(n^2)$ liens, en effectuant un calcul d'actualisation pour chacun. Cela induit un **coût** par itération qui est directement proportionnel à n^2 . Pour un réseau de taille modérée, cela reste gérable, mais dès que n se chiffrera à plusieurs dizaines ou centaines de milliers, ce simple balayage devient prohibitif.

B. Exemple Chiffré : Explosion en Temps et Mémoire.

Lorsque $n = 10^5$, le nombre total de couples (i,j) se situe autour de 10^{10} . L'opération de mise à jour $\omega_{i,j}(t + 1) \leftarrow \omega_{i,j}(t) + \dots$ imposerait alors de procéder à 10^{10} manipulations ou davantage à chaque itération. On en déduit un **temps** de calcul considérable pour la moindre passe de DSL, souvent incompatible avec des ressources de calcul ou des délais raisonnables. Par ailleurs, **stocker** tous ces coefficients $\omega_{i,j}$ sous forme d'une matrice dense occupe des dizaines de gigaoctets de mémoire, ce qui peut se révéler impossible ou hautement onéreux dans un environnement standard.

Même à l'échelle $n = 10^4$, on approche déjà 10^8 liaisons. Les itérations multiples (par exemple quelques centaines) entraînent alors un total potentiel de 10^{10} à 10^{11} calculs élémentaires, ce qui demeure fort lourd et ralentit significativement la convergence. Cette **double contrainte** (temps de traitement exponentiel et consommation mémorielle) se pose dès lors qu'on manipule un graphe complet, sans parcimonie ni structure éparsse.

C. Implications Dynamiques : Réévaluation Continue.

Le **DSL** ne s'arrête pas après une itération unique, car la philosophie même de l'auto-organisation requiert un **processus** progressif où la pondération $\omega_{i,j}$ évolue jusqu'à la stabilisation ou le repérage de clusters cohérents. Ce raffinement successif, sur plusieurs pas $t = 0, 1, 2, \dots, T$, instaure un coût cumulé de $O(n^2 \times T)$ opérations s'il faut recalculer chaque lien à chaque fois. Sur un T modérément grand (de 100 à 1000 itérations), on aboutit rapidement à des sommes astronomiques d'opérations, hors de portée d'une exécution normale.

Ainsi, un simple DSL complet, voulu pour dénicher des **clusters** ou structurer des entités, se heurte au mur des " $O(n^2) \times T$ ". Cet état de fait **annonce** la nécessité, explorée plus loin (chap. 7.2.3.2, 7.2.3.3), de recourir à des **approches** restreignant la densité du graphe, comme la parcimonie, le filtrage top- k , ou des structures de voisinage limité, afin de rendre le déploiement du SCN **scalable** pour de larges n .

D. Deux Grands Enjeux : Temps de Traitement et Mémoire.

En guise de synthèse, cette croissance en $O(n^2)$ fait peser deux charges essentielles :

Dans un **premier** temps, le **temps de traitement** grimpe en raison de la boucle sur tous les liens (i,j) . Plus précisément, à chaque itération, on traite $O(n^2)$ liens, et on ambitionne généralement de conduire plusieurs itérations (parfois des centaines). L'usage d'algorithmes plus complexes (inhibition, recuit) renchérit encore le coût de traitement par lien, rendant l'ensemble exponentiellement cher en n .

Dans un **second** temps, la **mémoire** constitue un frein si l'on veut conserver dans un tableau dense toutes les $\omega_{i,j}$. Pour 8 octets par pondération, on excède rapidement plusieurs dizaines de gigaoctets dès que n franchit la barre des dizaines de milliers. Outre la place disque, la mémoire vive peut s'avérer insuffisante et la charge d'accès alourdit le calcul.

Ces constats justifient la réflexion, amorcée dans les sections suivantes (7.2.3.2, 7.2.3.3), qui aborde la **parcimonie** (éliminer les liens faibles, ne garder que les k plus forts) ou d'autres techniques de **structure** (réseau épars, topologie imposée) pour contourner l'explosion quadratique du SCN complet. Cet aménagement s'avère indispensable pour mettre en œuvre le DSL à grande échelle, sans sacrifier la possibilité d'une auto-organisation hiérarchique ni l'évolution itérative qui caractérise ce paradigme.

Conclusion

Le **coût** $O(n^2)$ inhérent à un SCN complet rend la dynamique DSL difficile à déployer lorsqu'on aborde de **très grands** ensembles d'entités. Mathématiquement, l'algorithme se confronte à la fois à un **temps** d'exécution $O(n^2 \times T)$ sur de multiples itérations et à une **mémoire** $O(n^2)$, potentiellement gigantesque. D'où l'importance, dans les sections suivantes (7.2.3.2, 7.2.3.3), de proposer des **solutions** pour réduire la densité du graphe ou pour limiter la fréquence (ou l'étendue) des mises à jour. Il s'agit de **compromettre** la rigueur du SCN "idéal" (où tout est relié) au profit d'une **faisabilité** algorithmique acceptable. Ainsi, l'évolution vers un SCN **sparse** ($O(n k)$ au lieu de $O(n^2)$) ou vers des schémas plus parcimonieux en calcul s'impose souvent comme le **seul** moyen de traiter efficacement les grandes échelles.

7.2.3.2. Trouver un Compromis : Liaisons "Suffisamment Bonnes" sans Explosiver la Durée de Calcul

Il est courant, dans la formulation complète d'un **Deep Synergy Learning (DSL)**, de vouloir gérer pour chaque paire (i,j) de l'ensemble $\{1, \dots, n\}$ une pondération $\omega_{i,j}$. Le calcul systématique et la mise à jour de ces $\omega_{i,j}$ dans une descente de gradient ou dans tout autre algorithme d'optimisation induit un **coût** potentiel en $O(n^2)$. Dans un cadre idéal, on pourrait se permettre de prolonger la descente jusqu'à minimiser de façon quasi-exhaustive la fonction

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + (\text{termes d'inhibition ou de régularisation}),$$

mais, en pratique, dès que n grandit, le temps et la mémoire requis explosent de manière prohibitive.

D'un point de vue **mathématique**, la recherche d'un **minimum** global pour une telle \mathcal{J} (généralement non convexe) peut relever d'une complexité NP-difficile : à chaque itération, évaluer ou mettre à jour l'ensemble $\omega_{i,j}$ se chiffre en $O(n^2)$ opérations, et si l'on poursuit sur T itérations, on aboutit à un coût global $O(n^2 \times T)$. Pour des valeurs de n de l'ordre de 10^5 (voire même 10^4), ce volume de calcul excède largement ce que l'on peut allouer dans un contexte temps réel ou en big data, d'autant plus si des méthodes globales (recuit simulé, algorithmes évolutionnaires) s'ajoutent, multipliant encore les évaluations ou stockant une population de solutions.

Il apparaît alors rationnel de restreindre l'ambition d'un **optimum** strict et de viser plutôt une solution "suffisamment bonne" dans un temps calculé raisonnable. Sur le plan de la **descente** ou de la dynamique DSL, cela revient souvent à interrompre le processus avant qu'il ne sature complètement la courbe $\mathcal{J}(\Omega)$, ou à adopter des **approches** qui évitent de manipuler la totalité des liens à chaque step. Par exemple, on peut réduire la boucle de mise à jour à un sous-ensemble $\Omega_{(i,j)}$ sélectionné parmi les plus prometteurs, ou on peut imposer un schéma "k plus proches voisins" qui ramène la densité à $O(n k)$ au lieu de $O(n^2)$. Mathématiquement, on décrit alors un DSL "sparse", où l'on met à jour seulement

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

mais pour un ensemble restreint de paires (i, j) . Le coût par itération s'établit en $O(n k)$ au lieu de $O(n^2)$, ce qui devient bien plus abordable pour de grands n . On y perd la certitude de prendre en compte toutes les paires d'entités, mais on économise de façon drastique la mémoire et le calcul.

Cette idée d'un "**compromis**" entre la **qualité** et la **durée** est explicite dans de nombreuses applications réelles. Un SCN "assez performant" pour distinguer des **clusters** ou coopérer dans un problème de robotique collective peut surpasser un SCN "idéalement optimisé" qui mettrait trop de temps à converger, ou exigerait une architecture matérielle démesurée. Sur le plan de l'**auto-organisation**, on peut fixer un critère d'arrêt $\|\Omega(t+1) - \Omega(t)\| < \varepsilon$ sur quelques itérations, ou une borne T_{\max} au-delà de laquelle l'itération s'interrompt.

Il est de surcroît possible de coupler le DSL "truncation" avec des **méthodes** stochastiques (recuit, heuristiques globales) pour franchir des barrières d'énergie majeures, tout en maintenant un nombre limité de phases intensives. On effectue alors un certain nombre de "paliers" ou "températures" de recuit, sans aller jusqu'à un refroidissement extrême. Formellement, on peut exprimer la mise à jour comme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \nabla_{\omega_{i,j}} J(\Omega(t)) + \sigma(t) \xi_{i,j}(t),$$

mais en limitant la durée totale et en insérant un sous-ensemble restreint (i, j) . Le résultat final n'est pas l'**optimum** théorique, mais un arrangement localement stable, qualitativement satisfaisant, et calculé dans un budget prévisible.

En somme, la **rationalité** de ce compromis relève d'un raisonnement asymptotique : si l'on cherche à gérer des millions d'entités, aucune approche "complètement exhaustive" n'est possible en pratique. On adopte donc un DSL "rapide", parfois approximatif, qui converge vers une structure stable en $O(n k)$ ou en un temps $O(n^2)$ très partiellement employé (puisque l'on ignore une fraction énorme de paires). Les éléments non exploités (ou pas encore mis à jour) peuvent progressivement être revisités par un algorithme incrémental ou par un multi-run stochastique, réparti dans le temps. C'est là la philosophie du "**suffisamment bon**" : l'**auto-organisation** s'effectue sans la rigidité d'un optimum global, tout en assurant une cohérence globale acceptable dans des délais réalistes. Les sections suivantes approfondiront les techniques de sparsification (7.2.3.3) ou d'inhibition, montrant comment elles concilient le fonctionnement auto-organisé du DSL avec des budgets de calcul plus réalistes.

7.2.3.3. Pratiques d'Approximation (k-NN, etc.) : Étude Mathématique Plus Détaillée

Lorsqu'un **Deep Synergy Learning (DSL)** fait face à un **nombre élevé** d'entités, noté n , la gestion d'un Synergistic Connection Network (SCN) complet implique $O(n^2)$ liaisons $\omega_{i,j}$. Chaque itération de l'algorithme exige alors une mise à jour ou une réévaluation de tous ces liens, ce qui peut rapidement conduire à un coût $O(n^2 \times T)$ si l'on effectue T itérations. Dans cette section (7.2.3.3), on analyse des **pratiques** d'approximation — telles que la méthode **k-NN** ou le seuil ϵ — qui permettent d'**abaisser** la complexité tout en maintenant une **qualité** acceptable des liens. Sur le plan **mathématique**, il s'agit de limiter la dimension effective du réseau (réduire le nombre de liens manipulés) afin de rendre le DSL réalisable pour des valeurs de n très grandes, en s'appuyant sur l'idée qu'un **sous-ensemble** restreint de liaisons capture l'essentiel de la dynamique d'auto-organisation.

A. k-NN : Ne Conserver que les k Plus Proches Voisins

L'approche k-NN consiste à n'activer ou n'actualiser que les liens relatifs aux "k plus forts" ou "k plus grandes synergies" pour chaque entité \mathcal{E}_i . On définit, pour chaque $i \in \{1, \dots, n\}$, un ensemble $N_k(i)$ de **cardinalité** k correspondant aux entités j qui maximisent la synergie $S(i, j)$ ou la pondération $\omega_{i,j}(t)$. On force ainsi :

$$\omega_{i,j}(t) = 0 \quad \text{si} \quad j \notin N_k(i).$$

Le coût total chute alors de $O(n^2)$ à $O(n k)$, car chaque entité ne conserve que k liens "actifs". Sur le plan **algorithmique**, on maintient ces k voisins (ou k liaisons) à chaque itération ou toutes les T' itérations ; si, au cours de la mise à jour DSL, certains liens $\omega_{i,j}$ faiblement actifs deviennent soudain plus pertinents, ils peuvent "entrer" dans le top-k en évincant un lien moins fort. Mathématiquement, cela requiert de réévaluer, au moins périodiquement, les

synergies potentiellement exclues, ce qui se fait parfois via un mécanisme d'indexation (KD-tree, LSH) pour trouver plus efficacement les k plus fortes valeurs.

Cette **k -NN** locale favorise la constitution de grappes ou clusters plus denses, chaque nœud \mathcal{E}_i se limitant à un “voisinage” restreint. On obtient un SCN “semi-sparse” qui, selon les observations empiriques, saisit l’essentiel des interactions dominantes sans sacrifier trop la cohérence. Du point de vue **mathématique**, il s’agit d’un **troncage** partiel de la solution idéale, censé préserver les liens porteurs de synergie significative.

B. ϵ -radius : Ne Conserver que les Liens au-dessus d'un Seuil

Une autre méthode repose sur le choix d’un **seuil** $\epsilon > 0$. On écarte tout lien $\omega_{i,j}$ (ou toute synergie $S(i,j)$) demeurant sous ce seuil. On peut formaliser :

$$\omega_{i,j}(t) = \begin{cases} \omega_{i,j}(t), & \text{si } \omega_{i,j}(t) > \epsilon, \\ 0, & \text{sinon.} \end{cases}$$

On obéit ainsi à un filtrage “en coup de hache”, supposant qu’en deçà de ϵ , les liaisons apportent un **gain** négligeable par rapport à leur coût de calcul. Cette idée s’étend directement à $S(i,j)$: seuls les paires (i,j) dont la similarité dépasse ϵ sont retenues, puis on évolue localement selon la mise à jour DSL. L’opération ménage un **graphe** plus clairsemé, et donc un coût $O(m)$ où m est le nombre de paires réellement actives (typiquement bien inférieur à n^2 si ϵ est choisi de façon adéquate).

Il faut toutefois prévoir un mécanisme de **réactivation** : des liens initialement sous ϵ peuvent s’élever via la dynamique DSL (surtout si on recalcule la synergie). En théorie, on doit donc ponctuellement re-tester certains liens inactifs pour voir si leur pondération dépasse la barre ϵ . Sans cela, le risque est de fixer le graphe trop tôt, empêchant l’émergence de nouvelles connexions potentiellement utiles.

C. Autres Formes de Parcimonie et “Voisinage Restreint”

Au-delà du k -NN ou du ϵ -radius, on trouve la notion de “voisinage restreint”. On assigne à chaque entité \mathcal{E}_i un sous-ensemble $C(i)$ de candidats (quelques centaines parmi n) où la liaison $\omega_{i,j}$ est autorisée ou recalculée. On obtient alors un coût $O(\sum_i |C(i)|) \approx O(n k)$. L’**objectif** est de balayer un sous-graphe localement pertinent : l’entité \mathcal{E}_i n’entretient d’interactions que dans $C(i)$. Si, à un moment donné, $\omega_{i,j}$ semble monter ou descendre, on réajuste. Cette solution se rapproche de l’idée de k -NN, mais la liste $C(i)$ peut être déterminée par des méthodes d’approximate nearest neighbors (ANN) ou par une contrainte de géométrie (par exemple, si les entités sont distribuées dans un espace métrique).

On trouve aussi le concept de mise à jour partielle ou **batch** stochastique, consistant à ne ré-actualiser qu’une fraction des nœuds (ou des liens) à chaque itération. Sur le plan **mathématique**, on élimine la boucle $O(n^2)$ en transformant la descente DSL en un algorithme de type “échantillonnage stochastique”. Le réseau avance moins vite dans sa convergence, mais on diminue grandement le temps par iteration.

D. Équilibre Coût–Qualité grâce à ces Pratiques

La logique d’approximations (k -NN, ϵ -radius, voisinage restreint) répond à la préoccupation de faire fonctionner le DSL sur de grands ensembles d’entités, en réduisant la densité du réseau. D’un point de vue théorique, on considère que la majorité des liens (i,j) sont peu porteurs (synergie faible ou contrainte superflue). Il vaut alors mieux dédier les ressources de calcul aux liaisons réellement utiles. Un graphe clairsemé de taille $O(n k)$ ou $O(m)$ (selon le seuil ϵ) suffit à véhiculer l’**information** essentielle pour la formation de clusters ou pour l’émergence d’une coopération stable dans le DSL.

Sur le plan **mathématique**, on peut écrire :

$$\tilde{\omega}_{i,j}(t) = \begin{cases} \omega_{i,j}(t), & \text{si } (i,j) \in \Lambda, \\ 0, & \text{sinon,} \end{cases}$$

où $\Lambda \subset \{(i,j) \mid i < j\}$ désigne l’ensemble de paires “autorisées” (top- k , ϵ -radius, ou autre). La dynamique DSL se restreint alors à $\{\tilde{\omega}\}$. Les analyses expérimentales montrent que ce filtrage ne dégrade pas trop la qualité globale, car les liens omis étaient faiblement contributifs à la baisse d’énergie \mathcal{J} . Sur un plan formel, on peut définir un “**erreur**”

d'approximation ΔJ entre la solution restreinte $\tilde{\omega}$ et la solution hypothétique $\omega^{(\text{complet})}$. Tant que ΔJ demeure faible, la qualité du SCN final reste dans une zone “suffisamment bonne”.

Enfin, ces techniques s'inscrivent dans le continuum des “approximations polynomiales” : on troque l'exhaustivité $O(n^2)$ contre un graphe plus “sparse” $O(n k)$. Ce faisant, on retrouve le **compromis** exposé en 7.2.3.2 : le DSL peut fonctionner efficacement pour de grands n , quitte à sacrifier une petite part de performance théorique.

Conclusion

Dans la pratique de l'**auto-organisation** à grande échelle, les **pratiques d'approximation** (k -NN local, ϵ -radius, voisinage restreint, batch stochastique) forment un ensemble d'outils destinés à maintenir le SCN dans une complexité plus modeste que $O(n^2)$. Les entités ne conservent que les liens les plus pertinents, réduisant draconiquement la mémoire et le temps d'itération, et ce sans trop nuire à la performance globale. Ces méthodes incarnent la philosophie “suffisamment bonne” : on préfère un DSL sparse, aisément manipulable en temps requis, plutôt qu'une descente exhaustive inapplicable pour un grand n . On préserve la dynamique auto-organisée essentielle, tout en assurant la **faisabilité** algorithmique du DSL dans des conditions réelles ou des environnements dimensionnellement élevés.

7.3. Recuit Simulé et Perturbations Stochastiques

Le **recuit simulé** (Simulated Annealing) s'inspire du procédé métallurgique consistant à **chauffer** un métal (puis à le refroidir lentement) pour obtenir une **structure** plus homogène et plus solide. Appliquée aux réseaux **DSL** (Deep Synergy Learning) et à la minimisation d'une **énergie** $J(\Omega)$, le recuit simulé ajoute un **bruit** contrôlé sur $\omega_{i,j}$, qu'on **réduit** progressivement. Ainsi, on se donne la possibilité de “quitter” un minimum local, puis de **se stabiliser** plus tard lorsqu'on abaisse la “température”.

7.3.1. Recuit Simulé : Fondements

7.3.1.1. Inspiration Métallurgique : Chauffer (Fort Bruit) puis Refroidir (Finesse Locale)

Dans le cadre du **recuit simulé**, on s'inspire du procédé métallurgique où l'on *chauffe* un métal ou un alliage pour mobiliser les molécules, puis on le *refroidit* lentement afin de le stabiliser dans une configuration énergétique plus favorable. La transposition au **Deep Synergy Learning** (DSL) consiste à injecter un **terme stochastique** dans la mise à jour $\omega_{i,j}$, modulé par une « température » qui diminue au fil des itérations. Le présent exposé (7.3.1.1) décrit la **logique** de cette analogie (chauffer → refroidir) puis en donne la **formulation mathématique** au sein du DSL, en expliquant comment cette stratégie aide à sortir de minima locaux et à préserver la **finesse** du réglage final.

Dans la métallurgie, on **chauffe** le matériau à haute température, rendant les atomes très mobiles et évitant qu'ils se figent prématurément dans une structure sous-optimale. Ensuite, on **refroidit** graduellement, autorisant l'alliage à converger vers un état plus stable. La clé réside dans la **diminution progressive** de la température pour empêcher la cristallisation dans un mauvais minimum et favoriser l'exploration initiale, puis la fixation finale.

Lorsque l'on applique ce raisonnement à un **SCN** (Synergistic Connection Network), on complète la formule de mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(T(t)) \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ désigne un **bruit** (gaussien, uniforme...) d'amplitude gouvernée par la température $T(t)$. Lorsque $T(t)$ est **élèvée**, les fluctuations aléatoires autour du schéma déterministe sont conséquentes ; cela correspond à la phase de **chauffe**. Au fur et à mesure que l'on **refroidit** ($T(t) \downarrow$), on diminue la force du bruit, autorisant la dynamique DSL à se figer progressivement dans une configuration stable.

La **phase de chauffe** ou de “haute température” se caractérise par une variance du bruit $\sigma(T)$ suffisamment importante pour permettre aux pondérations $\omega_{i,j}$ de faire de grands “sauts”. Ces fluctuations peuvent être perçues comme la possibilité de **monter** en énergie locale, donc de s'extirper de certains minima. Dans un langage de descente de gradient, cela équivaut à « valider » des pas contraires au simple gradient négatif, et donc potentiellement utiles pour découvrir des vallées plus profondes (minima plus globaux).

La **phase de refroidissement** consiste à diminuer $T(t)$, réduisant le bruit et stabilisant la configuration. À mesure que la température approche zéro, la dynamique aléatoire se retrouve étouffée ; la mise à jour $\omega_{i,j}(t+1) \approx \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ tend vers la descente locale sans perturbation, figeant le SCN autour d'une solution jugée « optimisée ». Un refroidissement trop **rapide** comporte toutefois le risque de replonger dans un minimum local, tandis qu'un refroidissement trop **lent** induit un surcroît de calcul (d'autant plus que le SCN peut être de taille considérable).

Dans la **physique statistique**, la méthode de Metropolis, souvent considérée à l'origine du recuit simulé, consiste à accepter ou non les transitions d'une configuration vers une autre en se fondant sur la probabilité $\exp(-\Delta J/(k_B T))$. À **haute** température, on accepte de nombreuses transitions « défavorables » pour échapper aux puits d'énergie ; puis, en phase « froide », on accentue la descente locale. Dans un **SCN**, on ne manipule pas directement les acceptations/rejets de transitions mais on incorpore un **bruit** dans la formule DSL. À haute température, l'amplitude de ce bruit contrecarrera parfois la descente de $\omega_{i,j}$, permettant de

sortir de configurations loc. À basse température, le **bruit** tend à s'estomper, le réseau se figeant dans un état quasi déterministe dicté par la synergie $S(i,j)$ et la décroissance $\tau \omega_{i,j}$.

Dans bien des cas, on modélise $\xi_{i,j}(t)$ comme un **bruit gaussien** $\mathcal{N}(0,1)$ et on écrit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] + \sqrt{\alpha T(t)} Z_{i,j}(t),$$

où $\alpha > 0$ est un coefficient d'échelle, $Z_{i,j}(t) \sim \mathcal{N}(0,1)$, et $T(t)$ la température. Lorsque $T(t)$ est élevée, la variance $\alpha T(t)$ du bruit est grande, provoquant de fortes oscillations de $\omega_{i,j}$. À mesure que $T(t) \downarrow 0$, on retrouve presque la pure descente locale $\omega_{i,j} \leftarrow \omega_{i,j} + \eta[S(i,j) - \tau \omega_{i,j}]$.

Ce **grand** bruit initial permet d'**explorer** l'espace des configurations. Les liaisons $\omega_{i,j}$ peuvent franchir des barrières énergétiques et sauter en dehors d'une structure sous-optimale. Cette phase "chaude" agit comme un remue-ménage global, élargissant la zone de recherche. Sans ce levier, la dynamique DSL — basée sur un gradient local $[S(i,j) - \tau \omega_{i,j}]$ — peut s'emprisonner définitivement dans un minimum local (un faux cluster stable mais non optimal).

Après cette exploration, on **réduit** la température pour laisser la configuration $\{\omega\}$ se **fixer**. On arrête alors d'accepter de trop gros écarts et l'on affine localement. Le **paramétrage** de la loi de décroissance de T (ex. $T(t) = T_0 \alpha^t$ ou $T(t) = \frac{T_0}{\ln(t+2)}$) reste crucial :

- Trop rapide, on retombe dans du local,
- Trop lent, on multiplie le coût et la durée de convergence.

En pratique, on choisit souvent une décroissance géométrique α^t , ou un autre schéma, puis on fixe un nombre d'itérations par palier, gardant un œil sur l'ampleur du bruit résiduel.

Conclusion

Le **recuit simulé** introduit la **philosophie** "chauffer → refroidir" dans la mise à jour DSL, insérant un bruit fort en début (grande amplitude σ) pour **explorer** un large espace de configurations, puis un refroidissement lent pour **fixer** un état final de plus basse énergie. Cette technique, directement inspirée des procédés **métallurgiques**, s'avère précieuse pour **échapper** aux minima locaux et garantir une solution plus globalement satisfaisante à la **dynamique** du Synergistic Connection Network. La suite (7.3.2) précisera les **détails** de $\sigma(t)$ et (7.3.3) évoquera les *protocoles* formels ou pseudo-codes types pour intégrer ce recuit dans la boucle DSL.

7.3.1.2. Avantage : Échapper aux Minima Locaux

La **dynamique** d'un **Synergistic Connection Network** (SCN), lorsqu'elle est purement locale (sans bruit), peut se retrouver **piégée** dans des minima locaux ; la mise à jour de $\omega_{i,j}$ suit un schéma déterministe cherchant à "descendre" une fonction d'énergie $J(\Omega)$. Dans bien des systèmes non convexes, on risque un "**blocage**" dans une configuration stable sans être la plus avantageuse. C'est précisément là que le **recuit simulé** intervient : il permet d'effectuer des mouvements transitoires "contre le gradient" (augmentant localement J) afin de **franchir** des barrières d'énergie et de **découvrir** des vallées plus profondes. Le principe est de "**chauffer**" le réseau (phase bruitée) puis de le "**refroidir**" (phase de stabilisation).

Sans recuit, la règle DSL $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ se comporte comme une **descente** locale, on n'accepte pas de sauts qui augmentent l'énergie $J(\Omega)$. Or, pour sortir d'un **minimum local**, il faut parfois, ne serait-ce que momentanément, **monter** en énergie. Le recuit simulé l'autorise en injectant un **terme** stochastique dont l'ampleur est contrôlée par la "température" $T(t)$.

Historiquement, on définissait :

$$P(\text{accepter } \Delta \Omega) = \min(1, \exp[-\Delta J/T(t)]),$$

ce qui signifie que si $\Delta J > 0$ (hausse de l'énergie), on accepte **parfois** ce mouvement, surtout si $T(t)$ est élevé. Dans un **SCN**, on n'exécute pas forcément ce test explicite : on ajoute un bruit $\sigma(T(t)) \xi_{i,j}$ et on laisse la dynamique "monter" localement. Plus $T(t)$ est grand, plus la probabilité de franchir un "col" d'énergie est haute.

Dans l'espace Ω (toutes les $\omega_{i,j}$), la fonction d'énergie $J(\Omega)$ forme un **paysage** accidenté, avec plusieurs vallées (minima) séparées par des crêtes. Une descente de gradient locale reste dans la vallée initiale. Le **recuit** fournit un "tremplin" aléatoire pour franchir ces crêtes, atteignant potentiellement une vallée plus profonde.

On initie le recuit avec $T(t)$ **élevé**, d'où de fortes fluctuations permettant des "sauts" hors de la zone stable initiale. Puis on **refroidit** lentement $T(t)$: on espaces ou on limite les grands sauts, et on se stabilise dans la cuvette d'énergie la plus prometteuse. Le **choix** du planning $T(t)$ (décroissance logarithmique, exponentielle, etc.) importe : trop rapide, on replonge vite dans un minimum local ; trop lent, le coût en temps de calcul grandit.

Sans recuit (ou autre perturbation stochastique), le **SCN** s'en tient à une "descente" univoque. Le recuit "brouille" la mise à jour, autorisant un **sursaut** dans des configurations a priori moins favorables. Cela démultiplie la **capacité** d'exploration du réseau.

Dans la **dynamique** DSL, on sait (chap. 4.4) que plusieurs attracteurs se forment. L'absence de bruit fort rendra la transition d'un attracteur à l'autre quasi impossible. Le recuit permet de changer de bassin d'attraction au besoin, **augmentant** la chance de parvenir à un "clusterement" globalement plus intéressant.

Les mécaniques d'**inhibition** (chap. 7.1.2.2) ou d'**apprentissage continu** (chap. 7.1.2.3) peuvent se combiner au recuit : on peut, par exemple, enclencher un "réchauffement" soudain si le SCN se stabilise trop vite après l'arrivée de nouvelles entités. Le bruit élevé agit alors comme un "reset partiel" tout en préservant la structure acquise.

Conclusion

En injectant un **bruit** proportionnel à la température $T(t)$, le recuit simulé donne au **SCN** la **latitude d'augmenter** localement $J(\Omega)$ quand c'est nécessaire pour **sortir** d'un puits d'énergie. Cette démarche, inspirée du recuit métallique, se révèle essentielle dans bien des environnements complexes ou non convexes : elle assure un **équilibre** entre la descente locale (renforcement déterministe) et la **probabilité** d'explorer d'autres "vallées" (attracteurs) potentiellement plus favorables. Une fois la "phase chaude" (fort bruit) achevée, on **refroidit** progressivement, permettant au réseau de **converger** dans la configuration la plus stable découverte. De cette manière, le **DSL** gagne en **flexibilité** et en **robustesse** vis-à-vis de minima locaux — enjeu central dans l'**optimisation** du Synergistic Connection Network.

7.3.1.3. Inconvénient : Paramétrage Délicat (Planning de Température)

Le **recuit simulé** constitue un levier essentiel pour **échapper** aux minima locaux dans la dynamique d'un **Synergistic Connection Network (SCN)**. Il n'en demeure pas moins qu'il souffre d'un écueil caractéristique : la **détermination** d'un *planning* (ou *schedule*) de température $T(t)$ judicieux se révèle complexe. Trop souvent, un mauvais **calibrage** de la courbe $T(t)$ ruine soit l'effet d'exploration (si l'on "refroidit" trop rapidement), soit la rapidité de convergence (si la température reste élevée trop longtemps). On détaille ici les **défis** de ce paramétrage, les écueils qui en découlent et quelques heuristiques pour y pallier.

Le **recuit simulé** démarre habituellement avec un niveau de bruit élevé (forte "phase chaude"). Si T_0 est **trop faible**, le réseau ne "bougera" presque pas de son attracteur local, faute de grands sauts stochastiques. À l'inverse, si T_0 est **trop grand**, la mise à jour $\omega_{i,j}$ subit de vastes fluctuations, et le SCN se comporte de façon quasi chaotique ; toute ébauche de cluster pourrait se dissoudre avant d'avoir pris forme. Le choix du "juste milieu" pour T_0 — fonction des valeurs de η , τ , $\max(S(i,j))$, la taille du réseau, etc. — est déjà **non trivial** et peut exiger des expérimentations empiriques.

Passée l'initialisation, on applique un **schéma** de décroissance $T(t+1) = \alpha T(t)$ (refroidissement géométrique), ou $T(t) = T_0/\ln(t+2)$ (refroidissement logarithmique), ou encore d'autres formes (polynomiale, adaptative...). Ce taux de décroissance **détermine** la durée effective de la **phase chaude** et le moment où l'on entre en **phase froide** (temps où le bruit est faible). D'un point de vue **mathématique**, il s'agit d'un compromis typique de recuit :

- α trop proche de 1 \Rightarrow refroidissement **très lent** \Rightarrow exploration riche, mais temps de convergence potentiellement prohibitif,
- α trop bas \Rightarrow on “refroidit” trop vite \Rightarrow on perd le bénéfice du recuit, retombant dans un algorithme quasi local.

La phase “**chaude**” se termine au moment où la température $T(t)$ devient suffisamment **faible** ($\approx \varepsilon$). On se retrouve alors avec un **DSL** quasi déterministe : la probabilité de franchir des barrières d’énergie s’amenuise. Il est donc primordial de **ne pas** atteindre cette phase trop tôt, sous peine de se retrouver dans un minimum local, ni de la prolonger au-delà du raisonnable, sous peine de flotter longtemps dans un “bain” stochastique. Ce choix de l’instant critique t_{cool} fait partie du **planning de température**.

Dans un **SCN** volumineux (plusieurs milliers ou millions d’entités \mathcal{E}_i), la *phase chaude* doit être assez longue pour que le bruit ait l’opportunité de “visiter” un espace Ω de taille potentiellement gigantesque ($O(n^2)$ connexions). Sinon, le recuit n’exploré qu’un sous-espace insignifiant, et on tombe dans un *pseudo* minimum local. À l’inverse, rester trop longtemps en phase chaude maintient un bruit massif qui “détruit” toute tentative de stabilisation.

Dans **n’importe** quel recuit, on cherche la synergie entre exploration (phase chaude) et convergence (phase froide). Un **SCN** souffrant de bruit **persistent** tardif échoue à structurer nettement ses clusters ; en revanche, un **SCN** dans lequel la température chute trop précocement se **fige** sans atteindre une organisation satisfaisante. Le ratio “temps passé en exploration” versus “temps passé en affinement” doit donc se calibrer, souvent par **essais empiriques** ou via heuristiques basées sur la variation $\Delta\mathcal{J}$.

Si $S(i, j)$ varie d’un ordre de grandeur à l’autre, on doit veiller à ce que le bruit $\sigma(T)$ ne soit ni trop minime (inutile si $\max S(i, j)$ est grand) ni trop énorme (noyant toutes les différences de synergie). Sur le plan **pratique**, on peut normaliser $\{S(i, j)\}$ pour garder un intervalle commun, ou adapter le bruit $\sigma(t)$ à la **distribution** de synergies.

Plutôt que de recourir à un planning rigide (géométrique ou logarithmique), on peut envisager un **pilotage** adaptatif. Par exemple, si l’on remarque que la fonction d’énergie $\mathcal{J}(\Omega)$ stagne, on ré-augmente légèrement T pour relancer l’exploration ; si au contraire la configuration Ω bouge trop, on abaisse plus vite la température.

Une variante consiste à exécuter un recuit “multi-phase” ou “intermittent” : par périodes, on remonte la température, puis on redescend, etc. On enchaîne ainsi plusieurs “saisons” de recuit, assurant une exploration par à-coups. Cette démarche se montre plus **empirique** mais parfois plus souple, associant des phases déterministes à des re-chauffes ponctuelles.

Conclusion

Le recuit simulé **facilite** l’évasion hors des minima locaux, mais exige un **planning** de température bien pensé. Plusieurs **risques** se présentent :

- 486. **Refroidissement trop rapide** : on n’exploré pas assez, la dynamique revient à une quasi-descente locale,
- 487. **Refroidissement trop lent** : le réseau demeure longtemps dans un état chaotique, repoussant la convergence.

Ce **dilemme** contraint à une mise en place réfléchie de :

- la **température** initiale T_0 ,
- la **loi** de décroissance α^t ou $1/\ln(t)$,
- la durée de la **phase chaude** avant que $\sigma \approx 0$.

En pratique, on recourt souvent à des **approches** semi-empiriques ou adaptatives (monitoring de \mathcal{J} , réchauffes intermittentes) pour trouver un **compromis** entre la puissance d’exploration et la vitesse de stabilisation.

7.3.2. Protocole de Température

Lorsque l'on applique le **recuit simulé** au **DSL** (Deep Synergy Learning), on introduit une **température** $T(t)$ qui gouverne le **niveau** du bruit stochastique. Ainsi, la mise à jour habituelle de $\omega_{i,j}$ (cf. chap. 4) se voit complétée par un **terme** $\xi_{i,j}(t)$ proportionnel à $T(t)$. L'idée est de **chauffer** le système en début d'apprentissage (bruit élevé) pour échapper aux minima locaux, puis de **refroidir** (bruit décroissant), afin de stabiliser la solution.

7.3.2.1. Équation : $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t)$

Le **recuit simulé** dans un **Synergistic Connection Network (SCN)** se formalise en **complétant** la mise à jour habituelle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ par un **terme stochastique** $\xi_{i,j}(t)$. Ce terme, souvent appelé "bruit thermique", se rattache à la **température** $T(t)$ pour régler son amplitude. La formule générale est donc :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t).$$

Cette petite modification rend la **dynamique** du SCN stochastique. Elle permet, lorsqu'on "chauffe" (température haute), de **monter** localement en énergie $J(\Omega)$, offrant à la structure la possibilité de **quitter** un minimum local (cf. 7.3.1.2). On présente ici la logique détaillée et les avantages de ce **terme** $\xi_{i,j}(t)$.

En l'absence de recuit, la mise à jour DSL suit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Cette dynamique détermine une forme de "descente locale" (le SCN se rapproche d'un minimum énergétique s'il n'existe pas de barrière insurmontable).

Pour pouvoir franchir les barrières d'énergie, on ajoute $\xi_{i,j}(t)$. La mise à jour devient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t).$$

On spécifie alors que $\xi_{i,j}(t)$ est un bruit dont la **variance** dépend d'une **température** $T(t)$ (voir 7.3.2.2 et 7.3.2.3). Cela fait passer la descente pure à un **processus** de "bonds" stochastiques, autorisant des variations positives ou négatives indépendamment de la pente locale. D'un point de vue "énergie" (chap. 7.2.1), la descente **classique** $\Delta\omega \propto -\nabla J$ interdit d'aller vers un état de plus haute énergie. Or, franchir un "col" requiert momentanément $\Delta J > 0$. Le bruit $\xi_{i,j}(t)$ autorise de tels **sauts** avec une probabilité dictée par l'amplitude du bruit (qui elle-même dépend de $T(t)$).

Dans la méthode de Metropolis (recuit simulé "classique"), on calcule ΔJ puis on décide d'accepter ou non la variation. Dans la version "continue", on injecte directement $\xi_{i,j}(t)$. Lorsque $\xi_{i,j}(t)$ est grand, il peut vaincre la "barrière" correspondante, augmentant temporairement $J(\Omega)$. Ainsi, **plus** $T(t)$ est élevé, **plus** la probabilité d'accepter un saut énergétiquement défavorable est grande. On choisit souvent :

$$\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma^2(T(t))),$$

c'est-à-dire un **bruit gaussien** de variance $\sigma^2(T)$, avec $\sigma \propto T$. On peut aussi utiliser des distributions uniformes ou autres, l'important étant de **faire varier** $\sigma(t)$ (ou $T(t)$) pour passer d'une phase d'exploration large (T élevé) à une phase plus fine (T faible). Grâce à $\xi_{i,j}(t)$, même si la mise à jour locale $\eta[S(i,j) - \tau \omega_{i,j}(t)]$ ne permet pas de s'échapper d'un puits, on a un bruit aléatoire pouvant "faire grimper" $\omega_{i,j}$ (ou la faire baisser) assez fort pour "sauter" la barrière. Cela prolonge la **recherche** dans l'espace Ω , offrant une chance de trouver un **minimum global** ou du moins meilleur.

Tant que $T(t)$ reste grand, la composante stochastique est **significative**, et le SCN se "balade" sur le paysage d'énergie. Lorsque $T(t)$ diminue (phase de refroidissement), on fixe de plus en plus la structure, réduisant drastiquement les

grosses fluctuations. Au final, on converge vers une **configuration** stable, potentiellement exempte de blocage local (7.3.1.2).

Le bruit s'accorde aux mécaniques d'inhibition et de clipping (7.1.2.2) pour préserver la **cohérence** malgré les secousses aléatoires, et s'intègre également à un **flux** d'apprentissage continu (7.1.2.3) : par exemple, on peut “réchauffer” le réseau ponctuellement lorsqu'un lot de nouvelles entités apparaît, puis “refroidir” à nouveau.

Conclusion

En **résumé**, le recuit simulé dans un **SCN** s'écrit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t),$$

avec $\xi_{i,j}(t) \sim \mathcal{D}(0, \sigma^2(t))$ liée à la “température” $T(t)$. Ce simple **terme** $\xi_{i,j}$ modifie la dynamique DSL locale, permettant de **monter** localement l'énergie \mathcal{J} . À **température** élevée, on laisse le SCN “explorer” (porter un regard plus global). Quand la **température** chute, on “consolide” les clusters. Cette **combinaison** aboutit à un **mécanisme** de recuit qui, bien réglé, aide à **échapper** aux minima locaux et à réaliser un **clusterement** (ou partitionnement) final plus optimal.

7.3.2.2. $\xi_{i,j}(t) \sim$ Bruit Gaussien ou Uniforme, Amplitude Dictée par $T(t)$

Le recuit simulé (7.3.1) s'appuie sur l'ajout d'un **terme stochastique** $\xi_{i,j}(t)$ dans la mise à jour $\omega_{i,j}(t+1)$. Ce bruit, interprété comme un “bruit thermique”, dépend d'une **température** $T(t)$ qui gouverne l'**amplitude** des fluctuations. Dans le contexte du **Synergistic Connection Network** (SCN), il s'agit d'un mécanisme clé pour **échapper** aux minima locaux : tant que la température est **élevée**, de grands sauts aléatoires restent possibles ; lorsqu'elle diminue, le système se consolide autour d'un **minimum** plus stable. Ce passage de la phase “chaude” à la phase “froide” est au cœur de la logique du recuit. Cette section précise la nature du bruit $\xi_{i,j}(t)$, ses distributions possibles (gaussienne, uniforme...) et la façon dont sa variance ou son amplitude est **dictée** par $T(t)$.

Rappelons l'équation générale lorsque l'on introduit un **terme** stochastique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) \tau \omega_{i,j}(t)] + \xi_{i,j}(t).$$

Le $\xi_{i,j}(t)$ est alors le bruit, dit “**thermique**”, dont la distribution centrée ($E[\xi] = 0$) a une **variance** en rapport avec la température $T(t)$. De cette manière, en phase de **haute** température, la mise à jour $\omega_{i,j}(t)$ peut fluctuer substantiellement (faveur à l'exploration) ; en phase de **faible** température, le bruit s'amenuise et le SCN se fige (convergence locale).

On opte classiquement pour :

488. Une **loi gaussienne** : $\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma^2(t))$,

489. Une **loi uniforme** : $\xi_{i,j}(t) \sim \text{Unif}(-\Delta(t), +\Delta(t))$.

La **variance** (ou demi-largeur pour le cas uniforme) dépend de $T(t)$. Ainsi, on peut écrire :

$$\sigma(t) = \alpha \sqrt{T(t)}, \quad \text{ou} \quad \Delta(t) = \beta T(t).$$

L'essentiel est que l'amplitude du bruit suive la courbe de la **température** $T(t)$ (7.3.1.3).

On définit une **température** $T(t)$ décroissant au cours des itérations (ou du temps). Au début (phase chaude), T est élevé, on autorise de grands aléas ; plus tard (phase froide), T devient petit, le bruit se réduit, stabilisant la structure. Le recuit simulé requiert ainsi un **planning** de $T(t)$ (ex. $\alpha^t, \frac{1}{\ln t}$, etc.) :

$$T(t+1) = \alpha T(t) \quad (0 < \alpha < 1), \quad \text{ou} \quad T(t) = \frac{T_0}{\log(t+2)}, \quad \dots$$

Tant que $T(t)$ reste élevé, la **variance** du bruit est **grande** ($\sigma^2(t) \approx T(t)$), autorisant $\omega_{i,j}$ à faire des bonds aléatoires importants. Sur le plan **énergie**, même si $\Delta J > 0$, il y a une probabilité non négligeable que le réseau “monte” la barrière. Cela évite de se figer dans un attracteur local.

À mesure que $T(t)$ s’approche de zéro, on restreint drastiquement la **perturbation** $\xi_{i,j}(t)$. Les liens $\{\omega_{i,j}\}$ cessent de fluctuer significativement et se fixent. Sur le plan **DSL**, on retourne vers le cas quasi dénué de bruit, où la dynamique $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ détermine la consolidation finale des clusters.

Les **avantages** de cette approche résident principalement dans sa capacité à favoriser l'**évasion** des minima locaux. Tant que $T(t)$ reste suffisamment élevé, le bruit introduit peut permettre à $\omega_{i,j}$ de s’extraire d’un puits d’énergie et d’explorer des configurations alternatives. Un autre atout majeur réside dans le **contrôle précis du ratio exploration/exploitation**. Il est possible de moduler la durée de la phase chaude, dédiée à l’exploration, et celle de la phase froide, consacrée à l’exploitation et à la stabilisation des résultats obtenus.

En revanche, certaines **limites** sont à prendre en compte. Le **choix du planning** de décroissance de $T(t)$ représente un **paramètre sensible**, souvent difficile à ajuster de manière optimale. Une mauvaise calibration peut compromettre l’efficacité du processus. Par ailleurs, le **coût computationnel** est un élément non négligeable. Une phase chaude prolongée implique un grand nombre d’itérations au cours desquelles le réseau subit des modifications importantes, ce qui peut retarder significativement la convergence.

Des **variantes** permettent d’adapter la variance $\sigma^2(t)$ non seulement en fonction de $T(t)$, mais aussi selon la position de ω dans l’espace des solutions. Il est également possible d’opter pour un planning **multi-phase**, incluant quelques remontées de T en cas de suspicion de blocage. L’enjeu principal reste de maintenir un bruit proportionnel à la température afin de respecter le principe du recuit et d’assurer une exploration efficace de l’espace des solutions.

Conclusion

Le **terme** stochastique $\xi_{i,j}(t)$ dans la **règle** DSL, qu’on modélise comme un **bruit** gaussien ou uniforme à **amplitude** proportionnelle à $T(t)$, est la **mise en œuvre** concrète du recuit simulé :

490. **Phase chaude** : $T(t)$ élevé, **fortes** fluctuations $\xi_{i,j}$, exploration large,

491. **Phase froide** : $T(t)$ décroît, les **perturbations** se réduisent, on converge.

Cette **équation** unifiée $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots + \xi_{i,j}(t)$ décrit ainsi comment le **SCN** navigue au gré du bruit “thermique”. Le **succès** du recuit dépend certes du calibrage de $T(t)$, mais, bien mené, ce protocole de bruit piloté s’avère un **outil** puissant pour **échapper** aux minima locaux et viser un état final plus robuste.

7.3.2.3. Schéma Classique : $T(t+1) = \alpha \cdot T(t)$ ou $T(t) = \frac{T_0}{\log(t+1)}$, etc.

Dans un **recuit simulé**, la température $T(t)$ dirige l'**amplitude** du bruit $\xi_{i,j}(t)$ injecté dans la mise à jour $\omega_{i,j}(t)$ (cf. 7.3.2.1 et 7.3.2.2). Cette fonction $T: t \mapsto T(t)$ (le *planning de température* ou *schedule*) doit provoquer une **phase chaude** (haute température) favorisant l’exploration, puis une **phase froide** (faible température) permettant la stabilisation finale. Dans la pratique, on recourt le plus souvent à quelques **formes** simples de décroissance (généralement sur un intervalle $t = 0 \dots T_{\max}$), que l’on illustre ci-après.

A. **Schéma Géométrique** : $T(t+1) = \alpha T(t)$

On choisit une **température** initiale $T_0 > 0$. À chaque itération, on met à jour :

$$T(t+1) = \alpha T(t), \quad 0 < \alpha < 1.$$

En pratique, α se situe souvent entre 0.8 et 0.99, par exemple. L'idée est de réduire la température **exponentiellement** au cours des itérations :

$$T(t) = T_0 \alpha^t.$$

Après t itérations, on obtient $T(t) = T_0 \alpha^t$, qui peut descendre rapidement si α est notablement inférieur à 1 ($\alpha = 0.9$ par ex.). Plus α est proche de 1, plus la phase chaude s'étend dans le temps. On parle souvent de la "**demi-vie**" du recuit pour décrire l'échelle de temps où la température atteint la moitié (ou un petit pourcentage) de sa valeur initiale.

Cette méthode est très **simple** et fournit un contrôle direct via le paramètre α . Pourtant, elle peut :

- *Refroidir* trop vite (α trop bas), raccourcissant la phase chaude et limitant l'exploration,
- *Refroidir* trop lentement (α trop proche de 1), entraînant un grand nombre d'itérations avant la stabilisation.

B. Schéma Logarithmique : $T(t) = \frac{T_0}{\log(t+1)}$

Une autre approche consiste à définir :

$$T(t) = \frac{T_0}{\log(t+1)}, \quad t \geq 1,$$

(on peut légèrement modifier la formule pour le cas $t = 0$, ex. $\log(t+2)$) afin d'éviter toute division par zéro. La température est alors *infinie* en théorie pour $t \rightarrow 0$, puis diminue plus doucement qu'un schéma exponentiel.

Dans la **littérature** sur le recuit, le refroidissement logarithmique $T(t) \propto 1/\ln t$ jouit d'une justification **mathématique** : sous certaines conditions, il "garantit" (avec une certaine probabilité) l'accès à l'**optimum** global, car on n'arrête jamais vraiment de pouvoir franchir des barrières d'énergie. La baisse lente assure une exploration plus longue.

La décroissance est plus **lente** qu'avec $T(t) = \alpha^t$. Cela laisse plus de temps pour explorer, mais rend la convergence plus **lente** en fin de parcours. En **pratique**, on peut difficilement conserver une phase de très haute température trop longtemps pour de grands problèmes, car le nombre d'itérations grimpe vite.

C. Comparaisons et Ajustements Possibles

D'autres lois de décroissance existent :

- Polynomiale : $T(t) = T_0 / (1 + t)^\beta$ ($\beta > 0$),
- Adaptative : T réajustée selon la "qualité" des minima atteints, etc.

Le but commun est de prévoir un **début** (phase chaude) et une **fin** (phase froide), en modulant la durée plus ou moins agressive de la phase chaude.

D'un point de vue **théorique**, refroidir *lentement* (ex. $1/\ln t$) augmente la probabilité de dénicher la configuration globale minimale.

D'un point de vue **pratique**, un refroidissement trop lent peut être *coûteux* en temps, surtout dans de grands SCN.

Beaucoup d'implémentations choisissent une **décroissance** exponentielle (géométrique) simple. On paramètre α par quelques essais empiriques sur un échantillon ou un scénario type.

D. Regard sur la Convergence

Chaque itération se traduit par un **saut** dans l'espace Ω , dont la probabilité de franchir un mur d'énergie $\Delta J > 0$ dépend de $T(t)$.

- Si $T(t)$ **chute** trop tôt, on gèle le système ;

- Si $T(t)$ reste haut, on gaspille du temps en oscillations.

Le **nombre** d’itérations nécessaires pour atteindre une phase froide déterminée (ex. $T(t_{\text{cool}}) \approx \varepsilon$) dépend du schéma choisi. Avec un schéma exponentiel, on franchit ε en environ $\ln(\varepsilon/T_0)/\ln(\alpha)$. Avec un schéma logarithmique, la convergence mathématique peut être beaucoup plus lente, mais potentiellement plus “robuste” pour l’exploration globale.

Dans le DSL, on peut coupler ce planning à des **heuristiques** ponctuelles (ex. “mini-perturbations” si l’on détecte un blocage) ou à de l'**inhibition** qui agit comme un autre mécanisme d’ajustement (7.1.2.2). On peut même “re-montrer” la température ponctuellement si on suspecte un nouvel attracteur local.

Conclusion

En **recuit simulé**, le **planning de température** $T(t)$ constitue la **charnière** entre la phase chaude (fort bruit, haute probabilité de franchir des barrières) et la phase froide (faible bruit, stabilisation). Deux schémas classiques dominent :

492. **Schéma géométrique** :

$$T(t+1) = \alpha T(t), \quad (0 < \alpha < 1),$$

simple et **rapide** mais exigeant un choix de α pas trop proche de 1.

493. **Schéma logarithmique** :

$$T(t) = \frac{T_0}{\log(t+1)}$$

apportant **plus** de garanties théoriques de globalité, mais pouvant **ralentir** considérablement la fin de la convergence.

Le DSL se dote ainsi de ce “planning” pour **orchestrer** l’amplitude du bruit $\xi_{i,j}$ (7.3.2.2) :

- Haute température \Rightarrow grande **exploration**,
- Basse température \Rightarrow **cristallisation** du réseau.

Le **choix** du planning et le réglage de ses paramètres (valeur initiale, vitesse de décroissance) sont souvent **empiriques** et constituent l’un des aspects les plus délicats de la mise en œuvre du recuit simulé pour un **SCN** de taille réelle.

7.3.3. *Injection de Bruit Aléatoire*

Au-delà du **recuit simulé** (7.3.2) où la température $T(t)$ module l’amplitude d’un **bruit** $\xi_{i,j}(t)$, on peut envisager des **injections** de perturbations plus générales, sans nécessairement obéir à un planning de refroidissement strict. L’idée est de **secouer** la dynamique $\{\omega_{i,j}\}$ à des moments choisis pour élargir la recherche ou éviter la stagnation. Nous détaillons ici (7.3.3.1) les **formes** de bruit aléatoire possibles, avant (7.3.3.2) d’évoquer les “moments” d’injection et (7.3.3.3) l’impact sur la **convergence** et la **structure** de clusters.

7.3.3.1. **Formes du Bruit** $\mathcal{N}(0, \sigma^2)$, **Uniforme**, etc.

Lorsque l’on intègre un **terme stochastique** $\xi_{i,j}(t)$ dans la mise à jour d’un **Synergistic Connection Network (SCN)** (chap. 7.3.2), la **distribution** de ce bruit revêt une importance décisive. Plusieurs choix s’offrent à nous, chacun avec ses avantages et inconvénients : on peut opter pour un **bruit gaussien**, un **bruit uniforme**, voire un **bruit** discréteisé. L’essentiel est que son **amplitude** (variance ou demi-largeur) soit reliée à la “température” $T(t)$ (ou tout autre paramètre de recuit), de sorte que la **taille** des fluctuations se réduise ou s’intensifie en fonction de la phase de

refroidissement ou de chauffage. Cette section détaille les principales formes de bruit, leur mise en œuvre et leur impact sur la **dynamique** DSL.

A. Bruit Gaussien : $\mathcal{N}(0, \sigma^2)$

Le **bruit gaussien** (ou “normal”) est la forme la plus courante utilisée dans de nombreux contextes en raison de sa simplicité de génération et de ses propriétés analytiques intéressantes. Il est défini par la relation :

$$\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma^2(t)),$$

ce qui signifie que chaque $\xi_{i,j}(t)$ suit une loi normale centrée. L’écart-type $\sigma(t)$ est généralement dépendant de la **température** $T(t)$, avec une relation courante de la forme $\sigma(t) = c \cdot \sqrt{T(t)}$ ou une proportionnalité directe à $T(t)$.

L’un des principaux **avantages** de cette approche réside dans la facilité avec laquelle le bruit gaussien peut être généré, de nombreuses bibliothèques proposant des fonctions adaptées. Il offre également une grande **souplesse**, car un unique paramètre σ (ou $cT(t)$) permet de contrôler l’amplitude des variations. De plus, sa **long tail modérée** assure que les fluctuations restent centrées sur des valeurs moyennes, tout en autorisant occasionnellement des sauts plus importants, favorisant ainsi une exploration efficace de l’espace des solutions.

Toutefois, certaines **limites** doivent être prises en compte. Si σ est **trop grand**, les variations de $\omega_{i,j}$ peuvent être excessives dès les premières itérations, plongeant le réseau dans un état chaotique difficile à stabiliser. À l’inverse, si σ est **trop petit**, l’exploration devient insuffisante, limitant la capacité du système à échapper aux minima locaux.

Dans le cadre du **recuit** (7.3.2.3), l’évolution de $\sigma(t)$ suit une dynamique de réduction progressive. On passe ainsi d’une phase de haute variance, où l’exploration est intense, à une phase de faible variance, favorisant la stabilisation et la convergence du système.

B. Bruit Uniforme

Plutôt que d’utiliser un bruit gaussien, il est possible d’opter pour un **intervalle défini** $[-\delta(t), +\delta(t)]$ et de tirer $\xi_{i,j}(t)$ de manière uniforme. Cette approche se formalise ainsi :

$$\xi_{i,j}(t) \sim \text{Unif}(-\Delta(t), +\Delta(t)),$$

où $\Delta(t)$ est proportionnel à $T(t)$ dans le cadre d’un recuit. Cela signifie qu’à chaque itération, $\xi_{i,j}(t)$ peut prendre **n’importe quelle valeur dans l’intervalle** $[-\Delta(t), +\Delta(t)]$ avec une probabilité uniforme.

Cette approche présente plusieurs **avantages**. Tout d’abord, le bruit généré est **borné**, ce qui empêche l’apparition de valeurs extrêmes comme celles pouvant survenir dans le cas d’une distribution gaussienne aux longues queues. Par ailleurs, son **implémentation est triviale**, puisqu’il suffit de générer un nombre pseudo-aléatoire entre 0 et 1, puis d’appliquer un redimensionnement pour obtenir la valeur finale dans l’intervalle défini.

Cependant, certaines **limites** doivent être prises en compte. Contrairement au bruit gaussien, ce type de bruit n’a **pas de longues queues**, ce qui peut poser problème si l’on souhaite occasionnellement effectuer des “sauts” importants pour franchir de grandes barrières énergétiques. Par ailleurs, plusieurs analyses théoriques, notamment celles liées à l’algorithme de Métropolis, sont plus directement formulées en utilisant un bruit gaussien, rendant l’uniforme moins naturel dans certains contextes d’optimisation.

C. Bruit Discret ou “à Coups”

Plutôt que d’introduire un bruit continu, il est possible d’utiliser une **perturbation discrète**, où $\xi_{i,j}(t)$ ne prend que quelques valeurs spécifiques, comme $\{-\alpha, 0, +\alpha\}$ ou $\{-\Delta, +\Delta\}$, selon certaines probabilités. Cette approche conduit à une évolution discontinue de $\omega_{i,j}(t)$, ce qui peut rendre la dynamique plus “logique” en imposant une variation par **paliers fixes**. Ainsi, au lieu d’une perturbation graduelle, la mise à jour des poids consiste en une simple **incrémentation ou décrémentation** de α , ou une **stagnation** si aucun changement n’est appliquée.

Cette méthode présente plusieurs **avantages**. Son **implémentation est particulièrement simple**, puisqu’elle ne nécessite aucune fonction gaussienne ni génération complexe de bruit aléatoire. Elle peut également être **mieux**

adaptée à certaines architectures matérielles ou à des environnements où les valeurs manipulées doivent rester discrètes, comme dans des modèles symboliques ou quantifiés.

Cependant, cette approche comporte aussi des **limites**. L'absence de variations intermédiaires peut entraîner un **manque de finesse**, rendant impossible certains ajustements progressifs. Cela peut se traduire par des **oscillations plus abruptes**, où les transitions entre valeurs successives sont plus marquées qu'avec un bruit continu.

E. Bruit Corrélé ou Adaptatif

Dans la majorité des approches de recuit, chaque $\xi_{i,j}(t)$ est **indépendant**, ce qui signifie que chaque lien du réseau subit des perturbations aléatoires sans influence des autres. Toutefois, il est possible d'introduire un **bruit corrélé**, où certaines liaisons sont modifiées de manière coordonnée. Un exemple typique consiste à **pousser simultanément un ensemble de connexions**, par exemple lorsqu'un cluster entier doit être perturbé dans la même direction pour explorer de nouvelles configurations.

En complément, une autre variante repose sur l'**adaptation du bruit**. Ici, l'amplitude $\Delta_{i,j}(t)$ est ajustée en fonction de la situation actuelle du lien $\omega_{i,j}$. Si un lien est déjà bien établi, les perturbations sont réduites afin de ne pas **déstabiliser un cluster** trop facilement. En revanche, si un lien est faible ou instable, il bénéficie d'une **plus grande latitude** pour évoluer et se renforcer. Cette approche rend le recuit **plus intelligent**, en modulant dynamiquement les perturbations en fonction de l'état global du réseau.

F. La Règle DSL + Bruit

Quelle que soit la **distribution** choisie, qu'elle soit gaussienne, uniforme, discrète ou corrélée, on retrouve la formule générale :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t).$$

Dans cette relation, le terme $\xi_{i,j}(t)$ est modulé selon deux aspects. Tout d'abord, son **amplitude** est régulée par $T(t)$, influençant la force des perturbations introduites. Ensuite, sa **forme** dépend de la distribution choisie, pouvant être gaussienne, uniforme, discrète ou encore corrélée.

Au fil des itérations, l'application d'un **planning** spécifique permet de réduire progressivement la variance de $\xi_{i,j}(t)$. On passe ainsi d'une **phase chaude**, caractérisée par une forte exploration, à une **phase froide**, où les perturbations deviennent minimes, favorisant la stabilisation des valeurs de $\omega_{i,j}$.

L'évolution de la dynamique du système suit plusieurs principes clés.

Dans un premier temps, le vecteur $\Omega(t)$ subit simultanément l'effet de deux forces opposées. D'un côté, la **descente DSL** exprimée par $\eta[S - \tau\omega]$ tend à structurer le réseau. De l'autre, le **bruit** ξ introduit une perturbation aléatoire. L'ensemble produit un **processus stochastique** en dimension \mathbb{R}^{n^2} , dont la stabilité finale dépend directement du **planning de la variance** ainsi que de la structure des synergies $S(i,j)$.

Une conséquence immédiate de cette dynamique est la **formation de clusters**. Un bruit calibré de manière adéquate permet d'**explorer** différentes configurations de regroupement. À mesure que la phase finale s'installe, les clusters se **stabilisent**, limitant ainsi les fluctuations des liaisons $\omega_{i,j}$. D'un point de vue énergétique, les connexions entre noeuds ne subissent plus d'importantes perturbations et convergent vers des valeurs stables, qu'elles soient fortes ou faibles.

Toutefois, un bruit **excessif** peut perturber cette organisation. Lorsqu'il est trop intense, il devient possible de **briser des clusters déjà formés**, effaçant les synergies locales. Cela souligne l'importance d'un **recuit progressif**, où la réduction de $\sigma(t)$ ou $\delta(t)$ doit être réalisée en parfaite cohérence avec l'évolution de $T(t)$, afin d'assurer un équilibre entre exploration et stabilisation.

Conclusion

L'**injection** d'un bruit $\xi_{i,j}(t)$ dans la **règle** DSL peut s'opérer via différentes **distributions** :

494. **Gaussienne** $\mathcal{N}(0, \sigma^2)$: forme continue, symétrique, plus usuelle,

495. **Uniforme** $\text{Unif}(-\delta, +\delta)$: bornée, simple, contrôle direct de l'amplitude maximale,

496. **Discrète ou corrélée** : pour des contextes particuliers, ou pour imposer des “secousses collectives” sur un sous-ensemble de liaisons.

Toujours, l'**ampleur** du bruit s'adapte à la **température** $T(t)$ selon un schéma (géométrique, logarithmique, etc.), respectant l'idée du recuit : grande exploration initiale, convergence finale. Ainsi, le choix d'une **forme** de bruit (gauss, uniform, discrete...) et son **planning** de variance représentent deux **paramètres** de conception majeurs : ils fixent comment le SCN échappera aux minima locaux et comment il stabilisera ses **clusters** au terme du refroidissement.

7.3.3.2. Moments d'injection (chaque itération, ou par batch)

Pour qu'un **Synergistic Connection Network** (SCN) profite pleinement du **recuit simulé** (voir 7.3.2), on peut introduire un *terme stochastique* $\xi_{i,j}(t)$ à différentes **fréquences**. Cette composante bruitée est centrale à la logique de recuit : elle secoue le réseau, permettant de franchir localement des barrières d'énergie. Mais deux grandes approches se distinguent, selon **quand** on applique le bruit : (1) **à chaque itération**, ou (2) **périodiquement** (en “batchs” ou “cycles”), après un nombre d'itérations “pures” où l'on ne touche pas (ou peu) à la composante aléatoire.

A. Injection de Bruit à Chaque Itération

La mise à jour des pondérations suit la relation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ est tiré selon une distribution $\mathcal{D}(0, \sigma^2(t))$. Dans la version **totalement stochastique**, cette même règle est appliquée à **chaque** itération t . La gestion de la **température** $T(t)$, ou de l'écart-type $\sigma(t)$, permet de moduler **l'amplitude du bruit** tout au long du processus d'optimisation.

Dans une approche de **recuit continu**, les pondérations $\omega_{i,j}$ restent en **perpétuelle évolution**. À chaque pas t , elles peuvent augmenter ou diminuer localement sous l'effet du bruit, avec une phase initiale où ces variations sont **importantes** (phase chaude), puis de plus en plus **réduites** à mesure que le système atteint une **stabilisation** (phase froide). Contrairement à une approche où l'on introduirait des phases de stabilisation pure, cette méthode maintient une **perturbation constante** dans le réseau, ne laissant jamais le système évoluer uniquement selon la descente DSL.

Cette stratégie présente plusieurs **avantages**. Elle permet notamment d'**éviter un blocage dans un attracteur local**, puisque le bruit injecté garantit la possibilité de s'échapper d'un minimum piégé à **n'importe quelle itération**. De plus, elle favorise une **exploration homogène** de l'espace des solutions, en s'inspirant des processus stochastiques de type **Langevin**, tant que la température reste non nulle.

Toutefois, cette approche comporte aussi des **inconvénients**. La **perturbation permanente** peut ralentir la stabilisation du réseau et générer des **oscillations prolongées** si $\sigma(t)$ ou $T(t)$ ne sont pas soigneusement contrôlés. Un mauvais calibrage peut entraîner un système **trop agité**, incapable de converger efficacement. Pour éviter cet écueil, il est essentiel de **définir un planning de température rigoureux**, de manière à ce que le SCN ne soit pas inutilement perturbé lorsqu'il atteint une configuration stable.

B. Injection Périodique, par “Batch” ou “Cycles”

Dans cette approche, la dynamique DSL classique, basée sur la **descente locale sans bruit**, est laissée libre d'évoluer pendant K itérations consécutives. Puis, à intervalles réguliers, une **perturbation contrôlée** est appliquée en injectant un “**shoot**” de bruit $\xi_{i,j}(t)$. Mathématiquement, cette alternance est définie par la relation suivante :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)], & \text{si } t \bmod K \neq 0, \\ \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t), & \text{si } t \bmod K = 0. \end{cases}$$

Cette méthode établit une distinction claire entre les **phases de descente pure** et les **phases d'exploration**. Pendant $(K-1)$ itérations consécutives, le bruit est nul ($\xi = 0$), permettant ainsi au SCN de progresser vers un attracteur local

de manière naturelle. Puis, à la ***K*-ième itération**, un bruit est injecté, potentiellement **dépendant d'un planning de température**, afin de perturber l'état actuel et d'éviter un verrouillage dans un minimum local sous-optimal.

L'un des **avantages** majeurs de cette approche réside dans la présence de **phases calmes**, où le SCN peut se focaliser sur l'optimisation locale et éventuellement converger vers une solution pertinente. En parallèle, les **phases de recuit** assurent une opportunité ponctuelle **d'échapper aux pièges des minima locaux**, offrant ainsi un compromis entre stabilité et exploration. De plus, la paramétrisation de la fréquence K et de la décroissance de $\sigma(t)$, par exemple via une loi de refroidissement exponentielle de type $\sigma_{n_{\text{batch}}} = \alpha^{n_{\text{batch}}}$, permet d'organiser l'apprentissage en **cycles successifs**, facilitant l'ajustement global du processus.

Cependant, cette approche présente également quelques **inconvénients**. Il devient nécessaire de régler simultanément plusieurs paramètres : la **période K** , l'**amplitude du bruit**, et la **loi de refroidissement**, introduisant ainsi une **double combinatoire** dans l'ajustement des hyperparamètres. Si la valeur de K est **trop grande**, le SCN risque de rester bloqué dans un attracteur pendant un long moment avant qu'une perturbation ne le déloge. À l'inverse, si K est **trop petit**, la dynamique du réseau se rapproche de l'approche **totalement stochastique**, supprimant presque toute phase de stabilisation entre deux perturbations successives.

C. Comparaison et Choix

L'approche **full stochastique** repose sur une **exploration continue**, où le bruit est injecté à chaque itération sans interruption. Cette méthode permet une couverture homogène de l'espace des solutions, mais peut rendre la **stabilisation plus difficile**, en particulier si la variance σ ne décroît pas assez rapidement. Elle est largement utilisée dans des **schémas inspirés du processus de Langevin**, où le bruit joue un rôle central dans l'exploration dynamique des configurations possibles.

L'approche par **batchs ou cycles** introduit une alternance entre des phases de **descente locale pure** et des phases d'**exploration ponctuelle**, où le bruit est injecté périodiquement. Cette structuration permet une séparation plus nette entre la **phase d'exploitation**, où le SCN se stabilise autour d'un attracteur, et la **phase de montée**, où un recuit stochastique peut être appliqué si un minimum local semble inadéquat. Toutefois, cette méthode demande un **paramétrage plus complexe**, notamment dans le choix de la période K et de l'intensité du bruit à chaque cycle.

Au final, il n'existe pas de règle universelle pour choisir l'une ou l'autre de ces approches. Le choix dépend essentiellement du **besoin en exploration continue ou intermittente**. Les algorithmes DSL à grande échelle, ou ceux utilisés en **apprentissage continu**, peuvent privilégier l'approche par batchs lorsqu'il est nécessaire de conserver **des phases d'actualisation locale prolongées**, entrecoupées de perturbations ciblées lorsque le système détecte une stagnation (voir 7.3.3.3).

Conclusion

Le **moment** où l'on injecte le bruit $(\xi_{i,j}(t))$ fait l'objet de deux approches principales :

497.**À chaque** itération (recuit "full stochastique"),

498.**Périodiquement**, par cycles ou batchs, où le bruit n'est actif qu'après un certain nombre d'itérations pures.

Chacune présente un **compromis** : le recuit permanent assure une exploration ininterrompue, mais peut gêner la stabilisation locale ; le recuit par batch est plus segmenté, plus maîtrisé, mais exige de définir *quand* précisément lancer la "secousse" pour ne pas trop retarder la dynamique. Selon la taille du **SCN**, les objectifs de convergence et le **planning de température** (7.3.2.3), on optera pour l'une ou l'autre (voire un hybride).

7.3.3.3. Impact sur la Convergence et la Structure de Clusters

L'ajout d'un **terme de bruit** $\xi_{i,j}(t)$ (voir 7.3.2) dans la mise à jour $\omega_{i,j}(t)$ d'un **Synergistic Connection Network (SCN)** ne se borne pas à de simples fluctuations locales : il remodelle la **dynamique** du réseau et influence nettement la **convergence** ainsi que la **formation** des clusters. L'injection de bruit, modulée par la température $T(t)$, permet des mouvements "contre le gradient" susceptibles de **franchir** des barrières d'énergie et de **rompre** la stagnation dans des

minima locaux. Cette section décrit l'impact concret du bruit sur la stabilisation du SCN et la structure de clusters qui émerge.

A. Effet sur la Convergence

L'évolution des pondérations dans la règle DSL incluant du bruit suit la relation :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ suit une distribution $\text{Distrib}(0, \sigma^2(t))$, dépendant de la température $T(t)$. Ce processus peut être interprété comme un **système stochastique évoluant dans l'espace des pondérations** $\{\omega_{i,j}\}$, avec un double impact sur la convergence.

D'une part, le bruit permet de **franchir les barrières d'énergie**, en autorisant des fluctuations temporaires qui augmentent l'énergie J . Cette propriété est essentielle lorsqu'il s'agit de **quitter un attracteur local**, c'est-à-dire une structure stabilisée qui pourrait être sous-optimale. En perturbant momentanément les connexions, le SCN peut explorer de nouvelles configurations susceptibles de mener à un état plus performant.

D'autre part, l'amplitude du bruit influence directement la **vitesse de stabilisation** du réseau. Une **température** $T(t)$ **trop élevée** maintient une agitation prolongée, retardant ainsi la fixation des pondérations dans un état stable. Cette situation se traduit par des oscillations persistantes, empêchant une convergence ferme. À l'inverse, un **refroidissement trop rapide** risque d'interrompre prématurément la phase d'exploration, piégeant potentiellement le réseau dans un minimum local sans bénéficier des avantages de la dynamique stochastique.

Dans un schéma de **recuit simulé**, la phase initiale est caractérisée par une **température élevée** favorisant l'exploration globale de l'espace des solutions Ω . Progressivement, une phase de **refroidissement** diminue $\sigma(t)$, ce qui **stabilise** les pondérations et permet au SCN de se fixer dans un état optimisé. C'est généralement au terme de cette transition que l'on observe la **stabilisation la plus marquée**, lorsque le réseau s'installe définitivement dans un **minimum** découvert au cours des explorations successives.

B. Impact sur la Structure des Clusters

Lorsqu'un lien $\omega_{i,j}$ reste dans une **zone intermédiaire** (par exemple entre 0.4 et 0.6), sans basculer naturellement vers un état stable proche de 1 ou de 0, la dynamique DSL classique peut être insuffisante pour trancher. L'introduction d'un **bruit stochastique** $\xi_{i,j}(t)$ peut alors jouer un rôle déterminant dans l'**évolution de la structure des clusters**.

Si la synergie $S(i,j)$ entre les deux entités est favorable, le bruit peut **tirer la valeur de $\omega_{i,j}$ au-delà d'un seuil critique** (par exemple 0.7 ou 0.8), renforçant ainsi la liaison et favorisant son intégration définitive au sein d'un cluster. À l'inverse, si la synergie est faible, une **perturbation aléatoire peut réduire la pondération en dessous d'un seuil bas** (par exemple sous 0.2), entraînant la suppression du lien. Ce **basculement stochastique** permet de **clarifier la structure du réseau**, évitant une prolifération de liaisons ambivalentes qui compromettaient la netteté des regroupements formés.

Dans les premières phases d'évolution du SCN, le bruit favorise une **exploration dynamique**, où les clusters émergent et se réarrangent rapidement, explorant différentes configurations de sous-réseaux possibles. À mesure que la **température** $T(t)$ **diminue**, le réseau entre dans une **phase de raffinement** : les liaisons les plus solides se **renforcent**, tandis que les connexions plus faibles sont **supprimées** sous l'effet des perturbations. Cette transition progressive permet d'atteindre une organisation plus stable et plus cohérente.

Si le bruit ne disparaît jamais complètement, le SCN peut atteindre un **équilibre statistique**, caractérisé par des **oscillations autour de structures dominantes** plutôt qu'une convergence stricte. Ce phénomène est comparable à un **système à température non nulle** en physique, où les interactions entre particules fluctuent sans jamais se figer totalement. Dans certains cas, cette dynamique peut être décrite par une **distribution de Boltzmann**, où les pondérations ω continuent de varier autour de valeurs moyennes bien définies, maintenant ainsi une structure **semi-stable mais adaptable**.

C. Analyses et Indicateurs Pratiques

L'évaluation de l'impact du bruit sur la convergence et la structuration des clusters peut être réalisée à l'aide de plusieurs indicateurs. L'un des plus courants est le **taux de stabilisation**, qui peut être estimé en suivant l'évolution de la **variance globale** des pondérations, $\text{Var}(\{\omega_{i,j}\})$, ou encore la norme des variations $\|\Delta\omega\|$ entre itérations successives. Lorsque le bruit est important, cette variance reste **élevée plus longtemps**, traduisant un état dynamique instable. Elle commence ensuite à décroître progressivement lorsque $\sigma(t) \rightarrow 0$, signalant une stabilisation des connexions. Plus la **phase chaude** dure, plus la **stabilisation** est retardée.

Une autre mesure pertinente repose sur la distinction entre **cohésion intra-cluster et liens inter-cluster**. En suivant la somme ou la moyenne des pondérations $\omega_{i,j}$ au sein d'un **même cluster** \mathcal{C} et celles reliant des **clusters distincts**, il est souvent observé que le **bruit renforce ces contrastes**. Certaines connexions internes se **solidifient**, tandis que d'autres, plus faibles, finissent par disparaître sous l'effet des perturbations. Le résultat final est généralement une **meilleure séparation des clusters**, rendant les structures plus nettes et plus marquées.

Le bénéfice du bruit dans un **SCN** peut être évalué de manière **quantitative** en comparant les performances obtenues avec et sans recuit sur un ensemble de benchmarks. Plusieurs métriques, telles que la **modularité**, le **silhouette score** ou d'autres indices de qualité de partitionnement, indiquent souvent que l'introduction d'un bruit aléatoire **améliore la qualité finale** du découpage du réseau. Contrairement à une **descente locale purement déterministe**, qui peut enfermer le SCN dans un minimum sous-optimal, l'ajout d'une perturbation contrôlée permet au système de s'**extraire de ces pièges** et d'explorer des configurations plus optimales.

Conclusion

Lorsque le **bruit** $\xi_{i,j}(t)$ (piloté par un planning de température $T(t)$) est injecté dans la dynamique DSL :

499. **Convergence** : Le recuit **allonge** la phase d'exploration et rend la trajectoire $\{\omega(t)\}$ stochastique. Cela ralentit parfois la stabilisation, mais permet de **franchir** des barrières d'énergie et d'éviter des attracteurs locaux trop précoce.
500. **Clusters** : Les fluctuations font basculer plus rapidement les liaisons ambiguës vers des valeurs extrêmes (fortes ou faibles). Cela **clarifie** la formation des clusters, évitant une prolifération de liens moyens.
501. **Phases** : On distingue la **phase chaude** (bruit élevé, forte exploration) de la **phase froide** (bruit faible, stabilisation). On peut, en outre, maintenir un bruit résiduel non nul pour demeurer adaptatif, ce qui aboutit à un "équilibre stochastique" où les clusters oscillent autour d'un état moyen.

En **synthèse**, le recuit via $\xi_{i,j}(t)$ "bouscule" la convergence et la structure de clusters : le **SCN** ne se fige plus aussi vite, exploite la **stochasticité** pour se défaire des minima locaux, et aboutit à un **arrangement** de liaisons généralement plus stable (en phase froide) et plus net (grâce à une polarisation renforcée des pondérations).

7.3.4. Exemples d'Implémentation

Le **recuit simulé**, lorsqu'il est combiné à la dynamique DSL (Deep Synergy Learning), peut s'implémenter de manière relativement concise si l'on respecte quelques principes : (a) définir un **cycle** d'itérations où l'on met à jour $\omega_{i,j}$ via la règle DSL, (b) injecter un **bruit** dont l'**amplitude** dépend d'une température $T(t)$ décrivant la phase de recuit, (c) faire progressivement **décroître** la température pour passer d'une étape "exploratoire" (bruit élevé) à une étape "d'exploitation" (bruit faible).

7.3.4.1. Pseudo-Code d'un Cycle DSL + Recuit

L'implémentation concrète d'un **Synergistic Connection Network** (SCN) combiné au **recuit simulé** peut se décrire par une procédure itérative qui, à chaque étape, applique la **mise à jour DSL** habituelle puis ajoute un **terme stochastique** proportionnel à une **température** $T(t)$. Le schéma qui suit illustre cette logique : on part d'initialisations

$\omega_{i,j}^{(0)}$ et d'un plan de température, on effectue la mise à jour standard $\Delta_{\text{DSL}}\omega_{i,j}$, on injecte le bruit, puis on refroidit la température. Le tout se poursuit sur un nombre d'itérations donné.

```

Algorithm: DSLRecuitSimule( $\{\omega_{i,j}^{(0)}\}, S(\cdot, \cdot), \eta, \tau, T_0, \alpha, N_{\max}\}$ )
Initialisation:  $\omega_{i,j}(0) \leftarrow \omega_{i,j}^{(0)}$ ,  $T(0) \leftarrow T_0$ ,  $t \leftarrow 0$ .
Tant que  $t < N_{\max}$  :
  1)  $\Delta_{\text{DSL}}\omega_{i,j}(t) = \eta [S(i, j) - \tau \omega_{i,j}(t)]$ .
  2) Générer un bruit  $\xi_{i,j}(t) \sim \mathcal{N}(0, 1)$  (par exemple), puis poser  $\xi_{i,j}^{\text{(eff)}}(t) = T(t) \xi_{i,j}(t)$ .
  3)  $\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}\omega_{i,j}(t) + \xi_{i,j}^{\text{(eff)}}(t)$ .
  4) Appliquer si nécessaire un clipping ou une inhibition :
      $\omega_{i,j}(t + 1) \leftarrow \max\{0, \omega_{i,j}(t + 1)\}$  (ou tout autre réglage).
  5)  $T(t + 1) = \alpha \times T(t)$  (ou toute autre loi  $T(t) = T_0/\ln(t + 2)$ , etc.).
  6)  $t \leftarrow t + 1$ .
Fin de la boucle. Retourner  $\{\omega_{i,j}(N_{\max})\}$ .
```

La procédure débute en initialisant les **pondérations** $\omega_{i,j}(0)$ ainsi que la **température** $T(0) = T_0$. À chaque itération, on calcule d'abord la **variation DSL** qui correspond à la descente locale $\eta [S(i, j) - \tau \omega_{i,j}(t)]$. Ensuite, on génère un **bruit** $\xi_{i,j}(t)$ d'une certaine distribution (gaussienne, uniforme, etc.), dont l'**amplitude** est déterminée par $T(t)$. On met alors à jour la pondération $\omega_{i,j}(t + 1)$ en additionnant cette variation DSL et la **perturbation** stochastique. Parfois, on ajoute un **mécanisme de clipping** ou un terme d'**inhibition** pour contrôler la compétition (voir le chapitre 7.1.2.2). Finalement, on **refroidit** la température selon un plan de décroissance, par exemple $T(t + 1) = \alpha T(t)$. Cette boucle se répète jusqu'à ce que t atteigne la limite N_{\max} . Le **recuit** permet de visiter un espace plus vaste de configurations $\{\omega_{i,j}\}$, favorisant la sortie de minima locaux et la découverte d'une solution plus optimale.

7.3.4.2. Étude de Mini-Cas : 10 Entités, 2 Minima Globaux

Il est souvent plus aisés de saisir l'impact du **recuit simulé** (ou d'une perturbation stochastique) en pratiquant sur un **mini-cas** contrôlé. Dans cette section, on se limite à un réseau de **10 entités** $\mathcal{E}_1, \dots, \mathcal{E}_{10}$. On va construire la fonction de **synergie** $S(\cdot, \cdot)$ de sorte qu'il existe deux configurations distinctes — deux façons de partitionner ces entités en clusters — qui aboutissent à des **minima** (quasi) équivalents de l'énergie du SCN. On étudiera alors le comportement comparé d'une descente **DSL** purement locale, incapable de "sauter" d'un minimum à l'autre, et d'un **DSL + recuit** qui rend possible la transition entre configurations concurrentes.

A. Description du Mini-Cas

Il est supposé que nous disposons d'une règle d'énergie de la forme

$$\mathcal{J}(\boldsymbol{\Omega}) = - \sum_{i < j} \omega_{i,j} S(i, j) + \frac{\tau}{2} \sum_{i < j} \omega_{i,j}^2,$$

où $\boldsymbol{\Omega}$ agrège tous les $\omega_{i,j}$, le terme $\sum \omega_{i,j} S(i, j)$ capturant la "qualité" des clusters et $\tau/2 \sum \omega^2$ la régularisation évitant la prolifération de liens.

On suppose que, sur les 10 entités, deux regroupements Ω^A et Ω^B sont possibles :

- **Regroupement A :**

$\{\mathcal{E}_1, \dots, \mathcal{E}_5\}$ vs. $\{\mathcal{E}_6, \dots, \mathcal{E}_{10}\}$.

- **Regroupement B :**

$\{\mathcal{E}_1, \dots, \mathcal{E}_4, \mathcal{E}_{10}\}$ vs. $\{\mathcal{E}_5, \dots, \mathcal{E}_9\}$.

On définit la **table** $S(i, j)$ de sorte qu’au sein de chaque bloc d’A, la synergie S soit élevée (ex. +1), et qu’entre blocs A elle soit négative (ex. -0.2). Puis on fait de même pour B : au sein des blocs B, on maintient $S \approx +1$, et entre blocs B, on prend $S \approx -0.2$. En veillant à soigneusement calibrer les valeurs, on fait en sorte que la configuration Ω^A (où les liens internes à A sont renforcés, ceux externes sont faibles) présente une énergie $J(\Omega^A)$ égale ou presque à $J(\Omega^B)$. Ainsi, on obtient deux **minima** de même niveau ou quasi équivalent.

Comme A et B diffèrent dans la façon de distribuer les entités, ces deux solutions conduisent à des matrices ω nettement différentes :

- Dans Ω^A , on aura $\omega_{i,j}$ élevé si i et j appartiennent au **même** bloc A_1 ou A_2 .
- Dans Ω^B , les liens internes sont élevés pour la répartition spécifique de B.

Sans perturbation, la descente locale DSL ne pourra pas basculer d’une configuration où $\omega \approx \Omega^A$ vers Ω^B , car on se trouve dans une “vallée” où chaque pas local renforce l’état acquis plutôt que d’en sortir.

B. Comportement Sans Recuit : Descente Locale

Lorsque la descente **DSL** agit seule (sans bruit), on itère :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

On initialise $\omega_{i,j}(0) \approx 0$. Les petites différences aléatoires (ex. dans la synergie ou l’ordre de mise à jour) suffisent à “faire pencher” la configuration vers Ω^A ou Ω^B . Une fois un **bloc** de liens internes prend l’ascendant, le réseau se stabilise vite, ne disposant d’aucun mécanisme pour franchir la barrière qui sépare Ω^A de Ω^B .

Si l’on examine plusieurs *runs* :

- On constate qu’**environ** la moitié des exécutions aboutit à A, l’autre moitié à B (ou selon l’initialisation).
- On ne peut *jamais* faire la transition A → B en cours de route, car localement A est déjà un minimum stable, **aucune** petite variation ne reconfigure de façon drastique la partition.

C. Recuit Simulé : Franchissement de Barrière

Si l’on ajoute un **bruit** $\xi_{i,j}(t) \sim \mathcal{N}(0, \sigma^2(t))$, la mise à jour devient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t).$$

Au début ($t \approx 0$), on prend $\sigma(t) \approx \sigma_0$ relativement grand (phase chaude). Les liaisons $\omega_{i,j}$ subissent alors des variations aléatoires capables de **rompre** l’état Ω^A (si on s’oriente d’abord vers A) et d’initier la montée vers Ω^B . Cette aptitude à “monter” $\Delta J > 0$ localement, puis “redescendre” vers un autre attracteur, est l’essence du **franchissement de barrière**.

Lorsque l’on **refroidit** $\sigma(t)$ petit à petit ($\sigma(t+1) = \alpha \sigma(t)$ par ex.), ces oscillations se raréfient et le système finit par se poser dans un *unique* minimum — éventuellement celui qu’il n’aurait jamais atteint sans bruit. Si Ω^B est globalement un peu meilleur, le réseau a maintenant une **chance** de le découvrir, plutôt que de se figer trop tôt dans A.

D. Conclusion et Enseignements

Ce mini-cas de **10 entités** et **2 minima** ex aequo (ou quasi ex aequo) illustre la différence cruciale entre :

- La descente **DSL** pure, qui se **fixe** rapidement dans une configuration sans avoir la possibilité d’en sortir.
- Le **DSL** enrichi de *recuit*, où un **bruit** contrôlé ($\sigma(t)$) permet d’effectuer des **sauts** aléatoires hors de l’état local et, en phase chaude, d’examiner d’autres configurations.

Dans des configurations de plus grande ampleur (réseaux à centaines ou milliers d'entités, etc.), la probabilité d'**attracteurs multiples** se retrouve encore plus marquée. Le recuit (ou d'autres heuristiques stochastiques) s'avère donc déterminant pour :

- **Renforcer** l'exploration,
- **Maximiser** les chances de trouver un arrangement plus satisfaisant (moins d'erreur, cluster plus cohérent),
- **Éviter** la cristallisation prématurée dans un **minimum** local.

Le résultat, dans la pratique, est une dynamique DSL plus “globale”, où les minima d'énergie peuvent être dépassés par des “franchissements” rendus possibles par les phases de bruit élevé, puis consolidés lorsque le refroidissement $\sigma(t) \rightarrow 0$ se met en place. Cette étude simplifiée sert d'exemple **pédagogique** pour comprendre qu'en l'absence de recuit, un SCN peut rester figé dans l'**un** de ses minima locaux, alors qu'avec recuit, il lui devient **possible** de “switcher” durant la phase chaude et de se stabiliser dans un état potentiellement plus proche de l'optimum global.

7.3.4.3. Observations Pratiques : Stabilisation et Temps de Refroidissement

L'intégration d'un **recuit simulé** à la dynamique d'un **Synergistic Connection Network** (SCN) présente une série de phénomènes empiriques qu'il convient d'observer afin de calibrer au mieux le protocole de recuit. La formule de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ est le bruit stochastique d'amplitude dictée par la “température” $T(t)$, induit plusieurs **phases** distinctes dans le déroulement de la simulation. Les paragraphes qui suivent décrivent les conséquences pratiques de ces phases, depuis la “chauffe” initiale (bruit important) jusqu'à la “consolidation” finale (bruit quasi nul).

Au début de l'algorithme, la température $T(0)$ est fixée à une valeur relativement élevée, par exemple $T(0) = T_0$. L'amplitude du bruit $\|\xi_{i,j}(t)\| \approx T(t)$ se révèle alors importante, entraînant des variations relativement grandes des pondérations $\omega_{i,j}$. Cette agitation initiale pousse le SCN à “secouer” sa configuration : même si, localement, la descente DSL tend à renforcer certains liens et à en affaiblir d'autres, la fluctuation stochastique peut briser ces tendances à courte échéance. Cela favorise l'exploration de plusieurs configurations, en autorisant des “sauts” dans l'espace $\{\omega_{i,j}\}$. Si le réseau se situait dans un puits d'énergie sous-optimal, ces secousses ont de bonnes chances de l'en extraire.

Au cours de cette phase chaude, on assiste empiriquement à une grande volatilité de la matrice $\{\omega_{i,j}\}$. Il arrive parfois que des liaisons franchissent des valeurs élevées, puis retombent, de manière aléatoire. Les premiers itérations peuvent ainsi montrer une succession de formations éphémères de micro-clusters. Cette agitation n'est pas un défaut : elle empêche la cristallisation trop hâtive d'un attracteur. On note toutefois que si la température reste trop importante durant trop longtemps, le réseau peine à repérer et consolider une structure durable. À mesure que les itérations s'écoulent, le protocole de recuit va diminuer la température $T(t)$. Selon le planning choisi, on peut réduire T par une formule géométrique $T(t+1) = \alpha T(t)$ ou logarithmique $T(t) = T_0 / \log(t+2)$. Dans tous les cas, la force du bruit $\xi_{i,j}(t)$ se réduit progressivement. Les fluctuations deviennent moins violentes : un lien $\omega_{i,j}$ ne subit plus de brusques changements, mais de petites perturbations. Cette période est cruciale pour effectuer le “tri” entre des liaisons qui ont conservé une forte synergie intrinsèque, et d'autres liaisons qui n'étaient maintenues que par des hasards aléatoires. En pratique, on perçoit la naissance de clusters plus stables : le réseau commence à se structurer de façon plus nette et cohérente.

Le réglage de la vitesse de refroidissement influe considérablement sur la qualité de la solution trouvée et sur le temps de calcul. S'il s'avère trop rapide, le réseau n'a pas le loisir d'explorer suffisamment et risque de rester coincé dans un puits local. À l'inverse, s'il est trop lent, on prolonge énormément la phase chaude, ce qui maintient un haut degré d'oscillation et retardé la stabilisation, augmentant le coût en itérations. La pratique montre que l'on ajuste empiriquement α (dans un schéma exponentiel) ou la constante du planning logarithmique afin de parvenir à un compromis satisfaisant.

Une fois la température tombée en dessous d'un seuil, le bruit $\|\xi_{i,j}(t)\|$ devient négligeable devant la descente DSL. Le réseau, alors, se comporte presque comme une descente locale traditionnelle, qui va consolider les liaisons cohérentes avec la synergie $S(i,j)$ et affaiblir les liaisons injustifiées. Le SCN se "fige" dans un attracteur final, typiquement un regroupement (un ensemble de clusters) plus robuste qu'on n'aurait obtenu par simple descente sans recuit.

Si la phase chaude a été assez longue pour franchir les barrières d'énergie séparant les minima locaux, on espère aboutir à un minimum plus global, ou du moins un attracteur de meilleure qualité (énergie plus basse, clusters plus nets). Dans la pratique, on mesure souvent la modularité ou la somme $-\sum \omega_{i,j} S(i,j) + \tau/2 \sum \omega_{i,j}^2$ pour vérifier que le résultat est supérieur (ou inférieur, selon la convention) à celui obtenu par une simple descente locale.

Les expérimentations montrent que le choix des paramètres de recuit est toujours délicat : la température initiale T_0 , le facteur de décroissance α (ou l'échelle du planning logarithmique), la taille du SCN, la complexité du paysage d'énergie et le nombre d'itérations allouées sont autant de facteurs qui influent sur la probabilité d'échapper à un minimum local et sur le temps qu'il faut pour atteindre une configuration stationnaire.

De manière générale, on observe typiquement les comportements suivants :

- Une **phase chaude** à haute température où le SCN demeure très agité, forme puis détruit de nombreux micro-clusters.
- Un **refroidissement** graduel où le réseau commence à se stabiliser, les clusters vraiment pertinents (forts $S(i,j)$) se consolident, les liaisons fluctuantes étant poussées à se renforcer ou se réduire sous l'impulsion de fluctuations plus faibles.
- Une **phase finale** de faible bruit, où plus aucune transition majeure de la structure n'est observée, et l'on reconnaît un ensemble de clusters bien dessinés, indices d'une convergence vers un attracteur stable et vraisemblablement meilleur que celui obtenu sans recuit.

Au total, le recuit simulé se présente donc comme un compromis entre la prolongation d'une exploration aléatoire et la stabilisation locale, que l'on contrôle grâce à un plan de température. L'étude empirique révèle que ce protocole, tant que la phase chaude est assez longue et le refroidissement pas trop hâtif, permet d'accroître la qualité de la solution et d'améliorer la netteté des clusters finaux.

7.4. Inhibition Avancée et Contrôle de la Compétition

En plus des mécanismes de **recuit** (chap. 7.3) et d'autres approches stochastiques, le **DSL** (Deep Synergy Learning) s'appuie aussi sur des **règles de compétition** pour éviter une prolifération indiscriminée de liens $\omega_{i,j}$. Parmi celles-ci, on compte l'**inhibition dynamique**, qui vise à modérer les connexions excessivement nombreuses ou moyennes émanant d'une même entité, pour favoriser une **sélection** plus tranchée (des liens vraiment forts ou quasi nuls).

7.4.1. Inhibition Dynamique

L'inhibition, déjà évoquée en **chap. 4.2 et 4.4**, intervient comme un **terme** supplémentaire dans la mise à jour $\omega_{i,j}(t+1)$ du DSL. Elle introduit une **compétition** entre les liaisons sortantes d'une même entité i , incitant cette entité à "choisir" les liens les plus synergiques et à **affaiblir** ou **abandonner** les autres.

7.4.1.1. Rappel (Chap. 4.2, 4.4) sur l'Inhibition Latérale / Compétitive

L'**inhibition latérale** constitue un mécanisme de **compétition** appliquée aux liaisons sortantes d'une même entité dans un **Synergistic Connection Network**. Il s'agit d'imposer, pour chaque entité \mathcal{E}_i , une forme de **couplage** entre ses liens $\{\omega_{i,j}\}$: si l'ensemble des liaisons de \mathcal{E}_i se met à croître, un *terme* d'inhibition ajoute une pénalité commune qui freine ou inverse cette croissance. Ce principe fut abordé dans les sections dédiées (chap. 4.2 et 4.4) comme une manière de **sélectionner** les liaisons fortes et d'éviter la prolifération de nombreux liens intermédiaires. Il convient ici d'en rappeler la formule et la signification mathématique, en vue de développer des versions avancées (cf. §7.4.2).

Pour une entité \mathcal{E}_i , on appelle « latéral » ou « compétitif » le fait que la somme des liaisons $\sum_{k \neq j} \omega_{i,k}(t)$ exerce une **influence** sur la mise à jour de chaque liaison $\omega_{i,j}(t)$. En d'autres termes, l'entité i ne saurait avoir simultanément un grand nombre de liaisons toutes modérément fortes, car l'inhibition imposera une « taxe » proportionnelle à la somme de ces liaisons. Ainsi, si l'on souhaite maintenir un $\omega_{i,j}$ élevé, il faut le "mériter" par une synergie $S(i,j)$ suffisamment bonne pour compenser la pénalisation.

On schématisé cette idée dans une **formule** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau \omega_{i,j}(t) - \gamma \sum_{k \neq j} \omega_{i,k}(t) \right],$$

où $\gamma > 0$ représente le **coefficent** d'inhibition. La compétition s'exprime par le terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ qui affecte la croissance de $\omega_{i,j}$.

On rappelle que la **règle** DSL standard (sans inhibition) prend la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

L'ajout de l'inhibition y injecte une **interaction** supplémentaire entre les liaisons sortantes de la même entité i . Mathématiquement, le terme $\gamma \sum_{k \neq j} \omega_{i,k}(t)$ induit un couplage non linéaire qui "répartit" la capacité de \mathcal{E}_i à se connecter parmi ses différents partenaires. Si $\sum_k \omega_{i,k}$ devient trop grande, il devient plus **difficile** de poursuivre la croissance d'un lien $\omega_{i,j}$.

Par analogie biologique, on parle d'"inhibition latérale" dans un cerveau où un neurone fortement connecté pénalise la croissance de connexions concurrentes pour limiter la saturation.

Une conséquence directe de cette **compétition** est la réduction notable des **liens moyens**. Autrement dit, une entité i se retrouve à privilégier **quelques** liaisons $\omega_{i,j}$ qu'elle maintient à un niveau élevé, tandis qu'elle abandonne (tend à 0) la plupart de ses liaisons "moyennes" ou "faibles". Les **clusters** dans le réseau deviennent alors plus **tranchés** : chaque entité \mathcal{E}_i se concentre sur ses partenaires les plus synergiques et délaisse les autres. Sur le plan visuel, si l'on

représente le SCN sous forme d'un graphe, l'inhibition latérale tend à clarifier la structure en dessinant des communautés mieux séparées.

Exemple.

Pour une entité \mathcal{E}_i ayant quatre liaisons $\omega_{i,1}, \dots, \omega_{i,4}$, la compétition se formalise par un terme $-\gamma \sum_{k=1}^4 \omega_{i,k}(t)$ dans la mise à jour de $\omega_{i,j}$. Les liaisons se font "concurrence" : pour garder un lien $\omega_{i,j}$ élevé, il faut que le produit $\eta [S(i,j) - \tau \omega_{i,j}(t)]$ surmonte la pénalisation $\gamma \sum_{k \neq j} \omega_{i,k}(t)$. Dans la dynamique itérative, on verra qu'après plusieurs pas, \mathcal{E}_i ne conserve qu'un ou deux liens significatifs, tandis que les autres s'affaiblissent vers zéro. Ainsi se crée une structure plus économique en liaisons, où chaque entité "sélectionne" soigneusement ses connexions.

Conclusion

L'**inhibition latérale** ou **compétitive** permet de résoudre le problème de "connections moyennes multiples", en forçant chaque entité à "choisir" les liens qu'elle veut effectivement conserver. Cette contrainte s'ajoute à la règle DSL basique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Le résultat est un **réseau** (SCN) plus **sparse** et plus **isible**, avec des **clusters** aux contours mieux définis. Les sections ultérieures (7.4.1.2, 7.4.1.3) détailleront des extensions plus avancées, comme l'adaptation de γ au cours de l'apprentissage, ou la combinaison avec d'autres formes de régularisation, afin de mieux contrôler la **dynamique** de compétition et de stabilisation du réseau.

7.4.1.2. Approche Évoluée :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau \omega_{i,j}(t) - \gamma \sum_{k \neq j} \omega_{i,k}(t) \right].$$

L'équation ci-dessus enrichit la mise à jour habituelle de la dynamique DSL par un terme d'**inhibition latérale**, formalisé par $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$. Elle préserve les ingrédients du cadre classique : un **taux** d'apprentissage η , un **facteur** de décroissance τ , et la **fonction** de synergie $S(i,j)$ déterminant la compatibilité entre entités \mathcal{E}_i et \mathcal{E}_j . La nouveauté réside dans la présence d'un coefficient $\gamma > 0$ qui couple les liaisons sortantes de \mathcal{E}_i dans une forme de **compétition** ou de **sélection** plus marquée.

Dans la version traditionnelle du DSL, la pondération $\omega_{i,j}$ évolue surtout en fonction de l'écart $S(i,j) - \tau \omega_{i,j}(t)$. Cette relation favorise l'augmentation de $\omega_{i,j}$ lorsque la synergie $S(i,j)$ est positive et notable, tout en imposant un amortissement linéaire $\tau \omega_{i,j}$. Cependant, sans une forme de régulation complémentaire, une entité \mathcal{E}_i peut se trouver enclin à maintenir un trop grand nombre de liaisons de force moyenne ou modérément élevée : elle « s'étale » sur plusieurs partenaires, ce qui tend à engendrer des structures de clusters moins nettes et moins discriminantes.

Le terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ introduit une **compétition avancée**. Il signifie qu'à chaque fois que $\omega_{i,j}$ désire croître, elle doit non seulement surmonter la décroissance linéaire $\tau \omega_{i,j}$, mais aussi compenser l'**inhibition** proportionnelle à la somme des autres liaisons $\omega_{i,k}(t)$ sortantes de \mathcal{E}_i . Plus la somme $\sum_{k \neq j} \omega_{i,k}(t)$ est grande, plus la liaison $\omega_{i,j}$ fait face à une pénalisation. Le paramètre γ détermine l'intensité de cet effet : un γ plus élevé rend la compétition entre liaisons plus rude, forçant une entité à se focaliser quasi exclusivement sur une poignée de connexions dont la synergie est jugée supérieure, plutôt que de répartir ses ressources sur de multiples liaisons d'importance modérée.

Cette logique se rapproche de l'**inhibition latérale** rencontrée dans certaines architectures neuronales biologiques, où un neurone très actif inhibe ses voisins, ou dans certaines distributions de ressources où la somme des allocations impose un effet d'auto-limitation. Sur le plan mathématique, l'introduction du produit $\omega_{i,j} \sum_{k \neq j} \omega_{i,k}$ dans l'énergie implicite modifie l'analyse de l'équilibre. À l'état stationnaire, la valeur $\omega_{i,j}^*$ découle alors d'un compromis entre la

synergie $S(i,j)$, la décroissance linéaire $\tau \omega_{i,j}$ et la concurrence $\gamma \sum_{k \neq j} \omega_{i,k}$. La conséquence est une sélection plus franche : l'entité \mathcal{E}_i ne retient et ne développe vraiment que les liens soutenus par un score $S(i,j)$ assez élevé pour compenser l'inhibition. À défaut d'un tel score, la liaison décroît et s'éteint, menant à un réseau plus clairsemé en liaisons moyennes.

Au niveau des **clusters**, on observe ainsi une spécialisation accrue : les entités se concentrent sur un plus faible nombre de partenaires, ce qui fait émerger des communautés plus tranchées, avec des connexions internes plus franches et des connexions externes très faibles ou nulles. Cette compétition permet également d'éviter les « zones grises » où plusieurs liaisons modérées à la fois feraient flotter les entités entre différents clusters sans véritable cohérence. Le choix de γ peut être ajusté selon l'objectif recherché : pour détecter peu de gros clusters, on pourra privilégier une valeur γ moyenne, tandis qu'une valeur élevée de γ accentue l'élimination de liaisons et fait émerger un plus grand nombre de petits clusters très denses.

L'approche évoluée proposée dans cette équation répond donc à la question du « trop-plein » de liaisons d'intensité intermédiaire. Elle enrichit la mise à jour DSL en ajoutant de la **sélectivité** interne, un mécanisme qui couple les liaisons sortantes d'une même entité afin de prévenir la saturation et de favoriser une structure de clusters plus nette et plus robuste.

7.4.1.3. Ajuster γ en cours de route pour moduler la compétition

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma(t) \sum_{k \neq j} \omega_{i,k}(t).$$

Lorsque la formule d'**inhibition avancée** inclut le coefficient $\gamma > 0$ (cf. §7.4.1.2) devant la somme $\sum_{k \neq j} \omega_{i,k}(t)$, il est souvent pertinent d'envisager un γ **dynamique**, c'est-à-dire **variable** au fil des itérations, noté $\gamma(t)$. Cet ajustement au cours de l'apprentissage permet de **moduler** finement la **compétition** que s'imposent les liaisons sortantes d'une même entité \mathcal{E}_i . Dans la pratique, il est rare qu'une unique valeur de γ convienne aussi bien en début qu'en fin de formation des clusters : on souhaite généralement une **pénalisation** plus douce (ou quasi inexistante) aux premières itérations, pour laisser libre cours à l'exploration, puis une **pénalisation** plus forte lorsque le réseau amorce sa stabilisation, afin de clarifier les liaisons et d'éviter la stagnation de multiples liens moyennement forts.

A. Pourquoi un γ variable ?

Lorsque le SCN (Synergistic Connection Network) débute son apprentissage, on préfère garder γ plus **faible** de sorte que l'entité \mathcal{E}_i ne soit pas contrainte de sélectionner trop tôt des liens peu justifiés. Cette phase initiale, proche d'une exploration large, favorise la création de liens provisoires, lesquels se renforcent ou non selon la synergie $S(i,j)$. Si γ est trop grand dès le départ, une entité se retrouve cantonnée à un très petit nombre de connexions, ce qui peut nuire à la découverte de clusters potentiellement plus cohérents.

Par la suite, lorsque le réseau a eu le temps d'explorer divers rapprochements entre entités, on souhaite clarifier et préciser la structure émergente. On augmente alors $\gamma(t)$ pour renforcer la **compétition** : chaque entité, si elle entretient plusieurs liaisons moyennes, verra leur somme $\sum_{k \neq j} \omega_{i,k}(t)$ devenir un frein important. Seules les liaisons supportées par une synergie $S(i,j)$ suffisamment élevée surmonteront cette pénalisation, tandis que les autres s'étioleront jusqu'à disparaître. Ce mécanisme affine la clusterisation et évite des configurations où une entité se répartirait indéfiniment entre trop de partenaires modérés.

Au-delà de la simple transition “faible $\gamma \rightarrow$ fort γ ”, un schéma d'adaptation continue peut repérer, par exemple, que la densité moyenne du réseau (Densite(ω), le nombre de liens au-dessus d'un certain seuil, ou l'entropie) reste trop élevée. On peut alors accroître γ pour contraindre le nombre de liaisons. À l'inverse, si l'on constate que trop peu de liaisons sont actives, et que le réseau se morcelle, on peut réduire γ pour relâcher la compétition et redonner une chance à certaines liaisons de renaître.

B. Schémas de Variation pour $\gamma(t)$

Une méthode directe est de faire croître $\gamma(t)$ de manière linéaire à chaque itération :

$$\gamma(t+1) = \gamma(t) + \Delta_\gamma \quad (\text{ou } \gamma(t) = \gamma_0 + \alpha t).$$

On part d'une valeur initiale γ_0 modeste puis on l'accroît régulièrement. Cette montée graduelle favorise le basculement d'une **phase** d'exploration (où $\gamma \approx \gamma_0$) vers une **phase** de consolidation (où γ plus grand impose une stricte compétition).

On peut tout autant adopter une augmentation plus lente, par exemple

$$\gamma(t) = \gamma_0 + \beta \ln(t+1),$$

créant une progression de l'inhibition plus proche d'une croissance logarithmique. De même, un mode **pallier** consiste à maintenir γ fixe sur les itérations $0 \leq t < T_1$, puis à le doubler ou l'augmenter au-delà d'un certain nombre d'itérations. La dynamique du réseau évolue alors par sauts successifs : la première phase reste "tolérante" aux multiples liaisons, puis chaque nouveau pallier de γ accroît la concurrence.

Plutôt que de programmer $\gamma(t)$ à l'avance, on peut laisser l'algorithme réagir à des **indicateurs** de surcharge de liens, de stagnation, ou de densité excessive. Par exemple, si un certain indicateur $I(\omega)$ dépasse un seuil I_{\max} , on augmente γ d'un quantum δ_γ , puis on réévalue la structure. Ce schéma offre un contrôle **réactif**, plus intelligent qu'une simple loi temporelle, assurant que la compétition ne devienne ni trop punitive ni insuffisante.

C. Conséquences sur la Dynamique du Réseau

Lorsque γ était initialement faible, chaque entité \mathcal{E}_i a pu "essayer" plusieurs liaisons $\omega_{i,j}$. En rehaussant γ , on impose une décroissance significative aux liens dont la synergie $S(i,j)$ ne compensait pas l'augmentation de la somme $\sum_{k \neq j} \omega_{i,k}$. Au bilan, cela crée un *switch* plus ou moins progressif : du mode exploratoire au mode sélectif. Les clusters deviennent plus tranchés, la densité de liens "moyens" chute.

Sur le plan **mathématique**, la montée de $\gamma(t)$ peut se rapprocher d'un cycle "phase initiale de croissance des liens" → "phase de tri" → "phase d'ancrage des liens les plus pertinents." Les entités s'**ancrent** dans un cluster où leur synergie locale reste capable d'équilibrer le terme inhibiteur, formant in fine de solides blocs communautaires.

Une valeur γ_{\max} trop élevée peut conduire chaque entité à ne retenir qu'un unique lien majoritaire, voire aucun si la compétition l'emporte. Une valeur modérée conserve un équilibre où deux ou trois liaisons fortes par entité subsistent. Cette **marge** de choix est laissée au concepteur, selon les objectifs de parsimonie ou de communauté désirés.

Conclusion

Ajuster γ en cours d'apprentissage dans la formule inhibitrice

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma(t) \sum_{k \neq j} \omega_{i,k}(t)$$

se révèle un **moyen** efficace d'allier découverte et sélectivité. Dans une **première phase**, un faible γ donne de la liberté à l'entité \mathcal{E}_i pour essayer des connexions multiples. Puis, l'augmentation progressive de γ force une **compétition** renforcée, coupant les liens jugés insuffisamment justifiés et soulignant les connexions au $S(i,j)$ assez élevé.

D'un point de vue **opérationnel**, on peut procéder par **lois** de croissance (linéaire, logarithmique, paliers) ou via un **feedback** sur la densité ou l'entropie du réseau, rendant la compétition **adaptative**. Le résultat final est un SCN plus clair, où chaque entité \mathcal{E}_i a concentré ses liaisons sur un nombre restreint de partenaires clefs, permettant des **clusters** nettement dessinés et une structure globale plus robuste.

7.4.2. Ajustement Automatique de γ

Dans de nombreux scénarios, l'**inhibition** (cf. §7.4.1) est introduite pour limiter la prolifération de liens moyens ou trop nombreux. Cependant, choisir un unique γ fixe peut s'avérer **sous-optimal** : le comportement optimal d'inhibition

peut évoluer au fil des itérations ou selon l'état global du **SCN**. La solution consiste à définir un **mécanisme d'ajustement automatique** de γ , rendant l'inhibition **adaptative** en fonction d'indicateurs de la structure du réseau.

7.4.2.1. Stratégie Auto-Adaptative : si le Réseau Devient Trop Dense, on Accroît γ

Dans le **Deep Synergy Learning (DSL)**, les mécanismes d'**inhibition** visent à réguler la croissance simultanée de plusieurs liaisons fortes autour d'une même entité. On retrouve ainsi une mise à jour du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

dans laquelle le **coefficent** $\gamma > 0$ dicte le degré de **pénalisation** infligée à une entité \mathcal{E}_i possédant de multiples liens en parallèle. Plus la somme $\sum_{k \neq j} \omega_{i,k}(t)$ est élevée, plus la liaison $\omega_{i,j}$ est contrainte à diminuer. Cette **inhibition compétitive** ou **latérale** évite qu'une même entité sature le réseau par un trop grand nombre de connexions fortes.

Il arrive toutefois que le **réseau** apparaisse trop *relâché* (beaucoup de liaisons élevées) ou, au contraire, trop *bloqué* (inhibition excessive). Une **stratégie** consiste alors à rendre γ **auto-adaptatif**, si la **densité** ou la **somme** globale des $\omega_{i,j}$ devient trop importante, on accroît γ . Si, à l'inverse, le réseau peine à former des **clusters** et manque de liens forts, on réduit γ . Ainsi, la **compétition** s'ajuste dynamiquement en fonction d'un **indicateur** global, garantissant un **équilibre** entre trop de connexions et trop peu de connexions.

A. Idée Mathématique de l'Auto-Régulation

On introduit une variable $\gamma(t)$ qui évolue dans le temps au gré de la structure émergente. La mise à jour prend alors la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [\mathbf{S}(i,j) - \tau \omega_{i,j}(t)] - \gamma(t) \sum_{k \neq j} \omega_{i,k}(t),$$

où $\gamma(t)$ n'est plus constant, mais lui-même **modifié** à chaque itération. Pour piloter cette évolution de $\gamma(t)$, on se dote d'un **indicateur** $\Delta(t)$ traduisant la **quantité** ou la **densité** de liaisons. Plusieurs choix sont possibles :

502.Une **densité globale** $D(t)$ décrite par la proportion de liens au-dessus d'un certain seuil θ . Par exemple,

$$D(t) = \frac{1}{n(n-1)} \sum_{i \neq j} \mathbf{1}_{\{\omega_{i,j}(t) > \theta\}},$$

avec **1** la fonction indicatrice.

503.Une **moyenne des pondérations**,

$$\Delta(t) = \frac{1}{n(n-1)} \sum_{i \neq j} \omega_{i,j}(t).$$

504.Un **indice d'entropie** ou de **concentration**, apte à détecter si trop de liaisons se concentrent sur un petit nombre d'entités.

Pour contrôler γ , on fixe une valeur cible δ_0 : si $\Delta(t)$ dépasse δ_0 , c'est que le réseau s'**emballe** et on accroît γ . Si $\Delta(t)$ est trop bas, on le diminue pour relâcher l'inhibition. Une forme simple de cette **adaptation** s'écrit

$$\gamma(t+1) = \gamma(t) + \alpha [\Delta(t) - \delta_0],$$

où α est un **pas** modéré, assurant une correction **progressive** plutôt que brutale. On peut également borner $\gamma(t)$ entre deux valeurs γ_{\min} et γ_{\max} , afin d'éviter des inhibitions négatives ou infinies.

B. Signification et Avantages

Cette **auto-régulation** favorise le maintien d'une *densité* raisonnable de liaisons. Si le réseau évolue vers un état où nombre de liens sont très forts, alors $\Delta(t)$ augmentera au-dessus de δ_0 , ce qui incitera $\gamma(t)$ à croître et donc à **dissuader** un même nœud \mathcal{E}_i d'entretenir des poids élevés avec beaucoup de voisins \mathcal{E}_k . L'augmentation de $\gamma(t)$ accroît le terme

$$-\gamma(t) \sum_{k \neq j} \omega_{i,k}(t),$$

créant un **phénomène** de compétition renforcée et réduisant la prolifération de liens. À l'inverse, si le réseau se montre trop **frileux** et peine à consolider les connexions essentielles, $\Delta(t)$ descendra en dessous de δ_0 , ce qui fera **diminuer** $\gamma(t)$. Les entités auront alors plus de latitude pour renforcer plusieurs liaisons à la fois, facilitant la **constitution** de **clusters**.

Cette démarche s'inscrit dans une **vision** plus large de la *plasticité adaptative*, où non seulement les liens $\omega_{i,j}$ évoluent localement sous l'effet de la **synergie** $\mathbf{S}(i,j)$, mais où le *paramètre d'inhibition* γ se reconfigure également au fil du temps. Le **réseau** reste ainsi dans une zone de **densité** modérée, évitant d'être trop chargé ou trop vide. Les **clusters** émergent plus clairement, sans excès de dispersion ni "verrouillage" précoce.

Conclusion

Le fait de rendre **auto-adaptative** la **compétition latérale** améliore grandement la **flexibilité** du **Synergistic Connection Network** (SCN). En ajustant γ selon un indicateur global $\Delta(t)$, on obtient un **mécanisme d'auto-régulation** : lorsqu'une croissance excessive de liens est détectée, on **resserre** l'inhibition ; quand la dynamique est trop faible, on **relâche** la compétition. D'un point de vue **mathématique**, cette extension consiste à intégrer une variable $\gamma(t)$ qui, elle aussi, obéit à une **mise à jour** discrète. Le **Deep Synergy Learning** profite alors d'une convergence plus robuste, avec une **formation** de **clusters** à la fois plus sélective et plus stable, tout en évitant l'explosion de la **densité** ou la disparition de liaisons potentiellement utiles. Cette **stratégie** illustre la capacité du **DSL** à incorporer des **règles** dynamiques supplémentaires pour optimiser son **auto-organisation** de manière *autonome* et *équilibrée*.

7.4.2.2. Calcul d'un Indice de Densité Globale ou d'Entropie pour Piloter γ

Pour **adapter** dynamiquement le coefficient d'inhibition γ dans la règle de mise à jour du **DSL**, il est possible de s'appuyer sur une **métrique** globale : soit la **densité** moyenne des liens, soit une **entropie** capturant leur répartition. Cette information, collectée à chaque itération (ou à intervalles réguliers), guide alors l'évolution de γ . L'**idée** est la suivante : si la structure devient **trop dense** (nombreux liens moyens ou forts), on **accroît** γ afin de *renforcer* l'inhibition compétitive. S'il n'y a au contraire **pas assez** de connexions consolidées, on **réduit** γ pour donner plus de latitude à la croissance des poids.

A. Indice de Densité Globale

Pour avoir un aperçu du **degré** de connectivité du réseau, on peut définir l'indice :

$$I(t) = \frac{1}{\binom{n}{2}} \sum_{1 \leq i < j \leq n} \omega_{i,j}(t),$$

où n est le **nombre d'entités** du Synergistic Connection Network (SCN) et $\binom{n}{2} = n(n - 1)/2$ le **nombre total** de paires. On effectue ici la **moyenne** de l'ensemble des liaisons $\omega_{i,j}$. Par construction, plus $I(t)$ est élevé, plus les pondérations sont en général **fortes**, signe d'un réseau relativement dense. Inversement, $I(t) \approx 0$ reflète une quasi-absence de liaisons non nulles.

L'approche courante consiste à fixer une valeur-cible I_0 (densité souhaitée) et à régler $\gamma(t)$ selon

$$\gamma(t + 1) = \gamma(t) + \alpha [I(t) - I_0],$$

où $\alpha > 0$ est un **paramètre d'ajustement**.

- Si $I(t) > I_0$ (*réseau trop dense*), on **augmente** γ , accentuant l'inhibition et freinant la croissance simultanée de plusieurs liaisons.
- Si $I(t) < I_0$ (*réseau trop pauvre en liens*), on **baisse** γ , autorisant plus aisément la co-existence de multiples connexions fortes.

Comme exposé en 7.4.2.1, la **mise à jour** des poids

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma(t) \sum_{k \neq j} \omega_{i,k}(t)$$

devient alors **sensible** à $\gamma(t)$, elle-même régie par la **densité**. Mathématiquement, on ferme la boucle entre la **distribution** des liens (densité globale) et le **paramètre** d'inhibition, évitant tant la **surchauffe** (trop de liens) que l'**asphyxie** (pas assez de liens).

B. Indice d'Entropie pour la Distribution des Liens

Plutôt que de se focaliser sur la **moyenne** des poids, on peut vouloir contrôler la **répartition** des $\omega_{i,j}$. L'entropie de Shannon représente un candidat naturel. On normalise d'abord chaque liaison :

$$p_{i,j}(t) = \frac{\omega_{i,j}(t)}{\sum_{p,q} \omega_{p,q}(t)},$$

puis on définit l'**entropie** :

$$H(t) = - \sum_{i,j} p_{i,j}(t) \ln(p_{i,j}(t)).$$

Si $H(t)$ est **élevée**, alors la “masse” de connexions est *dispersée* sur beaucoup de paires (i,j) ; le réseau est “uniformisé” et peu sélectif. Un **faible** $H(t)$ témoigne d'une distribution **inégale**, marquée par quelques poids dominants et un grand nombre de liaisons proches de zéro.

Comme pour la densité, on peut fixer une cible H_0 (niveau souhaité de concentration ou de sélectivité) :

$$\gamma(t+1) = \gamma(t) + \beta [H(t) - H_0],$$

avec $\beta > 0$.

- **Réseau trop “plat”** ($H(t) \gg H_0$) : on **augmente** γ pour pousser les entités à se concurrencer davantage, réduisant le nombre de liaisons moyennes.
- **Réseau trop “concentré”** ($H(t) \ll H_0$) : on **baisse** γ , permettant à plus de connexions de survivre et évitant une monopolisation par quelques liens.

C. Implantation Mathématique dans la Dynamique

On reprend la **mise à jour** inhibitrice présentée en 7.4.2.1 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma(t) \sum_{k \neq j} \omega_{i,k}(t),$$

et on y associe un **feedback** :

$$\gamma(t+1) = \gamma(t) + f(\text{MesureGlobale}(t)),$$

où $\text{MesureGlobale}(t)$ est soit le score de **densité** $I(t)$ soit l'**entropie** $H(t)$. La fonction f peut être linéaire (comme dans les formules ci-dessus) ou plus sophistiquée, et l'on met éventuellement des **bornes** $\gamma_{\min} \leq \gamma(t) \leq \gamma_{\max}$ pour éviter des valeurs extrêmes.

Le **choix** du pas α ou β est crucial. Un pas trop grand peut entraîner des **oscillations** : dès que la densité dépasse δ_0 , on accroît γ trop brutalement, forçant aussitôt la densité à chuter sous δ_0 , etc. Généralement, on prend

$$\alpha, \beta \ll \eta,$$

assurant une **évolution** plus lente de $\gamma(t)$ que des $\omega_{i,j}(t)$, ce qui favorise la stabilisation progressive. En pratique, on peut mettre à jour $\gamma(t)$ toutes les L itérations pour lisser la réponse.

D. Discussion et Bénéfices

Adapter γ en fonction de la **densité** ou de l'**entropie** du réseau constitue un **mécanisme d'auto-régulation**. La compétition n'est plus figée, mais réagit au **niveau** global de connexions. Cela empêche un "emballement" où toutes les entités se connectent fortement, ou au contraire un "blocage" où l'inhibition demeure trop sévère.

Cette méthode se combine aisément à d'autres stratégies du DSL (seuil de suppression, recuit simulé, etc.). Lorsqu'on introduit de nouvelles entités ou qu'un **changement** de distribution survient, la **métrique** globale (densité ou entropie) en rend compte, ajustant automatiquement γ .

Grâce à l'**entropie**, on agit non seulement sur la *quantité* totale de liens, mais aussi sur leur *répartition*. Un certain **degré** de sélectivité peut être visé : trop de liens "tièdes" (entropie haute) peut être indésirable, de même qu'une concentration excessive (entropie trop faible).

Conclusion

Le **calcul** d'un **indice** global — qu'il s'agisse de la *densité* $I(t)$ ou de l'*entropie* $H(t)$ — offre une **façon élégante** de **réguler** γ (voir aussi §7.4.2.1). En adaptant la **compétition latérale** aux évolutions du **réseau**, on évite qu'il ne devienne trop riche en liaisons "moyennes" (densité trop forte ou entropie trop élevée) ou au contraire trop appauvri en connexions utiles. **Mathématiquement**, cela se traduit par une **dynamique couplée** :

$$\begin{cases} \omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma(t) \sum_{k \neq j} \omega_{i,k}(t), \\ \gamma(t+1) = \gamma(t) + f(\text{MesureGlobale}(t)), \end{cases}$$

qui améliore la **stabilité** et la **sélectivité** du réseau tout au long de l'**auto-organisation**. Cette logique d'**adaptation globale** peut s'appliquer dans divers contextes d'apprentissage non supervisé, renforçant la **robustesse** et l'**efficacité** du **Deep Synergy Learning**.

7.4.2.3. Éviter l'Excès d'Inhibition qui Bloquerait la Formation de Clusters Cohérents

Dans le cadre d'une **inhibition dynamique** (voir §7.4.2) visant à **réguler** la compétition entre liens sortants depuis une même entité \mathcal{E}_i , on utilise une équation du type

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

où $\gamma > 0$ constitue le **paramètre d'inhibition**. L'objectif, rappelé en §7.4.2.1 et §7.4.2.2, est de **contraindre** chaque entité \mathcal{E}_i à sélectionner un *petit* nombre de liaisons vraiment *pertinentes*, plutôt que d'entretenir simultanément de multiples liens moyens.

Un **problème** apparaît toutefois quand γ devient trop *élevé* : les termes d'inhibition peuvent alors **annuler** le renforcement attendu, même lorsque la synergie $S(i,j)$ est raisonnable, empêchant la **croissance** nécessaire des poids

$\omega_{i,j}$. Cette section explique **pourquoi** cet “excès d’inhibition” peut nuire à la **formation** de clusters cohérents et **comment** on peut y remédier.

A. Mécanisme du Blocage par Excès d’Inhibition

Le **terme** $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ dans la mise à jour agit comme une **pénalité** de compétition, fondée sur la masse totale de liens sortants depuis \mathcal{E}_i . Une valeur trop forte de γ crée une situation où la somme

$$\gamma \sum_{k \neq j} \omega_{i,k}(t)$$

risque de **dépasser** le gain $\eta [S(i,j) - \tau \omega_{i,j}(t)]$ alloué à la liaison $\omega_{i,j}$. Même lorsque la synergie $S(i,j)$ n’est pas négligeable, la contribution négative liée à la compétition peut se révéler plus importante que le renforcement positif, d’où une **variation** $\Delta \omega_{i,j}(t)$ négative ou presque nulle.

Cet **effet** provoque un **blocage** : la plupart des liaisons $\omega_{i,j}$ ne parviennent pas à franchir un **seuil** minimum pour se consolider. Les entités \mathcal{E}_i s’orientent alors vers une configuration *quasi stérile* où les poids restent faibles. Si la synergie n’atteint pas un niveau *extrêmement* haut (capable de compenser la pénalisation concurrente), on n’observe pas de véritable clusterisation. Le réseau s’appauvrit et peine à distinguer des groupes d’entités pourtant compatibles.

B. Ajustement de γ pour Éviter le Blocage

La solution réside dans un **contrôle** plus fin de γ . Les sections précédentes ([§7.4.2.1](#) et [§7.4.2.2](#)) décrivent des stratégies d'**auto-adaptation** du paramètre d’inhibition, qui tiennent compte d’une métrique globale : densité, entropie ou somme moyenne des poids. Lorsque des **indicateurs** montrent que la compétition est devenue *excessive*, il convient de **baisser** γ pour laisser s’exprimer davantage de liaisons.

Un **exemple** consiste à surveiller la *densité sortante* de chaque entité \mathcal{E}_i ,

$$d_i(t) = \sum_j \omega_{i,j}(t),$$

qui doit rester au-dessus d’un certain seuil (par exemple, κ). Si la plupart des $d_i(t)$ stagnent en dessous de κ , on décrémente γ . Sur un plan mathématique, cela revient à écrire

$$\gamma(t+1) = \gamma(t) - \alpha [\kappa - \bar{d}(t)],$$

dès lors que $\bar{d}(t) < \kappa$. On parle alors de **rétroaction** négative : plus le réseau se montre *sous-connecté*, plus la pénalisation γ diminue, libérant progressivement la croissance des poids $\omega_{i,j}$.

D’autres variantes consistent à utiliser l'**entropie** globale ou l'**indice de densité** moyen $I(t)$ abordés en [§7.4.2.2](#), pour s’assurer que le réseau ne dérive pas vers une situation où trop *peu* de liens se forment.

C. Maintien d’un Juste Milieu

Le **fondement** du DSL repose sur la dialectique entre le **renforcement** (lié à la synergie S) et la **décroissance** (terme $\tau \omega_{i,j}$), auxquels s’ajoute la **compétition latérale** $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$. Trouver un “juste milieu” implique :

- 505. Un γ suffisamment positif pour **restreindre** la croissance simultanée de multiples liaisons “moyennes” et **forcer** un choix sélectif,
- 506. Un γ pas trop élevé afin de **permettre** la co-existence d’au moins deux ou trois liaisons fortes chez \mathcal{E}_i , condition souvent nécessaire à la **formation** de clusters un tant soit peu riches.

Lorsque γ franchit un **cap critique**, le réseau “sèche” et ne génère plus que des *micro-connexions* isolées, sans véritable regroupement de synergies. L'**auto-organisation** se retrouve alors *bridée*, ne reflétant pas le potentiel d’interaction entre entités.

Conclusion

L'**excès d'inhibition** (γ trop grand) crée un **blocage** dans la formation de **clusters** cohérents : la compétition étouffe les liens même pour des synergies $\{S(i, j)\}$ somme toute honnêtes, poussant les pondérations $\omega_{i,j}$ vers un régime *sous-critique*. Le **SCN** devient ainsi trop **épars** et perd la capacité de structurer les entités en groupes significatifs. Pour y remédier, on **règule** γ (éventuellement de façon *automatique*) en s'appuyant sur des mesures comme la **densité**, l'**entropie** ou la **cohésion** locale. Ce **pilotage** évite le phénomène de “tuer-lien” généralisé et préserve la possibilité de voir émerger des **clusters** plus complexes. En ce sens, la **compétition latérale** n'est pas un simple frein, mais une **force** modulable, devant être *dosée* pour stimuler une sélectivité qui reste *productive* pour l'**auto-organisation** du Deep Synergy Learning.

7.4.3. Méthodes de Seuil Adaptatif

Dans l'optique d'**alléger** la structure du SCN (Synergistic Connection Network) ou de **contrôler** la compétition entre liens (cf. 7.4.1, 7.4.2), il est souvent judicieux d'introduire un **seuil** θ : dès que la pondération $\omega_{i,j}$ descend **sous** ce seuil, on la **remet** à zéro (ou on la coupe). L'idée est de **forcer** la parcimonie, en éliminant les liens “faiblement pertinents”.

7.4.3.1. Imposer $\omega_{i,j} = 0$ si $\omega_{i,j} < \theta(t)$

Dans le **Deep Synergy Learning (DSL)**, il est fréquent de vouloir **supprimer** (ou rendre **nulle**) toute liaison jugée insuffisamment élevée, afin de **clarifier** la structure du **Synergistic Connection Network (SCN)** et de maintenir la **parcimonie** des liens. Cette idée se concrétise via une **règle de seuil**, généralement appliquée en **post-traitement** après la mise à jour “classique” des poids $\omega_{i,j}$.

A. Règle de Seuil (« Hard Threshold »)

Après avoir calculé $\omega_{i,j}(t + 1)$ selon la règle du DSL — décrite, par exemple, en 7.4.1 ou 7.4.2 — on impose :

$$\omega_{i,j}(t + 1) = \begin{cases} \omega_{i,j}(t + 1), & \text{si } \omega_{i,j}(t + 1) \geq \theta(t), \\ 0, & \text{sinon.} \end{cases}$$

où $\theta(t)$ est un **seuil** (constant ou variable) dictant la “limite” en deçà de laquelle on **annule** la liaison. D'un point de vue purement **algorithmique**, on :

507. **Met à jour** la matrice $\{\omega_{i,j}(t)\}$ à partir du **DSL** (chap. 7.4.1 et 7.4.2).

508. **Applique** le “coup de ciseau” : tout $\omega_{i,j}(t + 1)$ dont la valeur retombe sous $\theta(t)$ est **fixé à zéro**.

Interprétation : on effectue un “**hard thresholding**”. Une liaison trop faible est jugée *superflue* et retirée purement et simplement. Le **réseau** conserve alors uniquement des **pondérations** supérieures ou égales à $\theta(t)$, ce qui **réduit** le nombre de liens à manipuler et **accentue** la sélectivité.

B. Rôle dans la Parcimonie

Le recours à une **règle de seuil** est un levier de **parcimonie** : on élimine les liaisons dont la contribution au réseau est jugée négligeable, évitant un **graphe trop dense**. D'autres mécanismes (ex. *inhibition* ou *saturation*) luttent déjà contre la prolifération des liens de moyenne amplitude, mais le “hard threshold” agit comme un **filtre** final, assurant une **topologie** bien plus épars.

- **Avantage** : Lisibilité et calcul facilité. On ne s'occupe plus des liens $< \theta$, ce qui diminue la complexité de mise à jour (si on choisit de *ne plus* recalculer $\omega_{i,j}$ une fois mis à zéro).

- **Inconvénient** : Danger de *couper* un lien légèrement en dessous de θ qui aurait pu “refleurir” par la suite si la synergie augmentait. On peut y remédier via une **politique** de réactivation ou un **seuil** adaptatif (voir §7.4.3.2 et §7.4.3.3).

C. Justification Mathématique

Une manière de **motiver** ce hard thresholding repose sur une **analyse** du “coût vs. utilité” de chaque liaison $\omega_{i,j}$. Dans les **approches** d’optimisation avec régularisation (type ℓ_1), on sait que :

509.Les liens de très faible amplitude **n’apportent** pas un gain notable dans la fonction d’**énergie** ou d’**affinité** globale.

510.Ils **alourdissent** néanmoins la structure et la complexité du réseau (coûts de stockage, calcul, possible bruit).

En mettre la **valeur** à zéro se rapproche d’une **projection** sur un ensemble **épars**, où les plus petits coefficients sont *purement annulés*.

Matriciellement, on peut voir le “coup de ciseau” comme une **projection** \mathcal{P}_θ :

$$\tilde{\omega}_{i,j}(t+1) = \max\{\omega_{i,j}(t+1), 0\} \mathbf{1}(\omega_{i,j}(t+1) \geq \theta(t)).$$

(ou une forme voisine). Dès que $\omega_{i,j}(t+1)$ s’avère en dessous du seuil $\theta(t)$, on force la valeur à 0. Ceci renvoie à des concepts d’**opérateurs proximaux** régulièrement utilisés en **optimisation parcimonieuse**.

D. Choix de θ

Comme l’indique la notation $\theta(t)$, le **seuil** peut lui aussi *évoluer* au fil du temps. Deux grandes approches :

511.**Seuil fixe** $\theta > 0$: on choisit dès le départ une valeur (petite ou moyenne) qui convient à la dimension $\|\omega\|$.

512.**Seuil adaptatif** $\theta(t)$: on fait **monter** (ou **descendre**) le seuil à mesure que l’**apprentissage** progresse, poussant le réseau à se “consolider” en un petit nombre de liaisons fortes.

Un θ trop **grand** coupe *trop* de liens, risquant de **fragmenter** exagérément le SCN et d’empêcher la formation de clusters de taille correcte.

Un θ trop **faible** laisse survivre de multiples liens moyens ou faibles, nuisant à la sélectivité et augmentant la **complexité** algorithmique.

En pratique, l’utilisateur (ou l’algorithme d’**auto-adaptation**, §7.4.3.2–§7.4.3.3) règle θ en cherchant un compromis entre parcimonie et cohérence du réseau (clusters suffisamment soudés).

Conclusion

Imposer $\omega_{i,j} = 0$ si $\omega_{i,j} < \theta(t)$ offre une **méthode** directe et **mathématiquement** justifiée pour forcer la **parsimonie** du SCN. Les étapes typiques sont :

513.**Calcul** : on met à jour $\omega_{i,j}(t+1)$ selon la règle de **plasticité** (ex. inhibition, saturation).

514.**Seuil** : on applique le test $\omega_{i,j}(t+1) \geq \theta(t)$; sinon, $\omega_{i,j}(t+1) \leftarrow 0$.

Ce *post-traitement* de **hard thresholding** est à la fois **simple** (pas de paramètre d’optimisation supplémentaire, hormis θ) et **efficace** (réduit immédiatement la masse de liens). Il constitue un **complément** à l’inhibition compétitive (chap. 7.4.1) ou à la saturation (chap. 7.4.2), améliorant la **lisibilité** et la **sélectivité** du réseau. Les sections suivantes, 7.4.3.2 et 7.4.3.3, détailleront des moyens d’**adapter** dynamiquement le seuil θ et illustreront son impact sur la **formation** de clusters.

7.4.3.2. θ Peut Varier au Fil du Temps pour Encourager la Parcimonie Progressive

Dans la **règle de seuil** exposée en 7.4.3.1, fixer θ de façon *constante* peut conduire soit à un élagage *trop précoce*, soit à un maintien prolongé de nombreux liens “moyens”. Il est alors souvent plus judicieux de **faire évoluer** θ au cours des itérations, pour réaliser une **transition** d’un réseau initialement tolérant (préservant la possibilité d’une large exploration) vers un réseau final plus **parsimonieux** (ne gardant que les connexions robustes).

A. Formalisation : $\theta(t)$ comme Fonction Croissante ou Mixte

Une manière simple de faire **monter** le seuil θ avec l’itération t est de définir

$$\theta(t) = \theta_0 + \beta t \quad (\text{ou } \beta \sqrt{t}),$$

où $\theta_0 > 0$ est la valeur initiale (modeste) et β un petit coefficient de croissance. Tant que t reste faible, $\theta(t) \approx \theta_0$ demeure peu strict, autorisant la **survie** de nombreuses liaisons moyennes. À mesure que t grandit, $\theta(t)$ devient plus exigeant et **écarte** graduellement les liens trop faibles.

D’autres lois s’avèrent utiles selon la *vitesse* de resserrement souhaitée :

515. Logarithmique

$$\theta(t) = \theta_0 (1 + \alpha \ln(1 + t)),$$

assurant un **resserrement** lent (la fonction \ln croît modérément).

516. Exponentielle

$$\theta(t) = \theta_0 \exp(\alpha t),$$

favorisant un **resserrement** très rapide.

Dans tous les cas, on aboutit à l’idée qu’une liaison $\omega_{i,j}(t)$ doit désormais être **supérieure** à $\theta(t)$, laquelle *augmente* avec le temps. Les liens qui ne suivent pas cette exigence finissent **coupés** (remis à zéro).

On peut résumer l’approche comme suit :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}^*(t+1), & \text{si } \omega_{i,j}^*(t+1) \geq \theta(t+1), \\ 0, & \text{sinon,} \end{cases}$$

où $\omega_{i,j}^*(t+1)$ désigne la **mise à jour** DSL (inhibition, saturation, etc.) avant **seuillage**, et $\theta(t+1) \geq \theta(t)$. En imposant une barre de plus en plus haute, on **forcer** progressivement la **sélectivité** dans le réseau.

B. Impact sur la Dynamique du DSL

Lorsque $\theta(t)$ reste **faible** (début d’apprentissage), on maintient activement un nombre plus important de liaisons $\omega_{i,j}$. Cela autorise :

517. Exploration des synergies potentielles.

518. Évitement d’un verrouillage trop tôt dans des solutions “sous-optimales” : certains liens modestes en début de parcours pourraient s’avérer plus utiles plus tard, au fur et à mesure que la synergie $S(i,j)$ se renforce.

À mesure que la variable $\theta(t)$ **s’accoûte**, de nombreux liens autrefois au-dessus du seuil se retrouvent “plafonnés” et sont coupés :

- La **dynamique** DSL conserve alors seulement les plus fortes $\omega_{i,j}$, traduisant des synergies fermement établies.
- Le **réseau** se *clarifie* et laisse apparaître des **clusters** plus nets, sans la contamination de liens “moyens”.

Concrètement, on peut représenter la **règle** sur deux niveaux :

519. **Mise à jour :**

$$\omega_{i,j}^*(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] \pm \dots \quad (\text{inhibition, etc.}).$$

520. **Post-Traitement :**

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}^*(t+1), & \text{si } \omega_{i,j}^*(t+1) \geq \theta(t+1), \\ 0, & \text{sinon.} \end{cases}$$

Avec $\theta(t)$ croissant, on resserre la condition de conservation sur $\omega_{i,j}$.

C. Avantages et Limites

L'un des principaux bénéfices de cette approche est son **équilibre entre exploration et stabilisation**. En début de processus, un certain **degré de permissivité** est maintenu, ce qui permet au SCN d'explorer diverses configurations sans restriction excessive. Progressivement, à mesure que l'apprentissage avance, une **sélection plus stricte** des connexions s'opère, favorisant la consolidation des liens les plus robustes. Cette dynamique permet d'obtenir une structure **éparse et lisible**, où les sous-groupes significatifs apparaissent avec plus de clarté.

Le **paramétrage de la loi de croissance** ($\theta(t)$) constitue un défi important. Différentes courbes peuvent être envisagées, qu'elles soient **linéaires, logarithmiques ou exponentielles**, mais leur choix est délicat. Une suppression **trop précoce** des liens risque d'entraver la formation de clusters cohérents, tandis qu'une suppression **trop tardive** peut maintenir des connexions non pertinentes trop longtemps. Un autre écueil potentiel apparaît en **phase finale**, lorsque ($\theta(t)$) devient trop élevé : il peut alors supprimer des liens qui commençaient pourtant à s'affirmer. Pour remédier à ce problème, il peut être judicieux d'introduire une **stratégie de réactivation** ou une **fenêtre de tolérance**, permettant d'éviter l'élimination prématurée de connexions pertinentes.

Cette approche peut être combinée efficacement avec d'autres stratégies d'optimisation du SCN :

- **L'inhibition compétitive** ([Section 7.4.1](#)) : cette méthode limite naturellement la prolifération des liens pendant l'apprentissage, avant que le filtrage final ne soit appliqué via ($\theta(t)$).
- **Le recuit stochastique** ([Section 7.3](#)) : en maintenant du **bruit** en début d'apprentissage, le réseau peut explorer différentes solutions. Progressivement, à mesure que le recuit progresse, ($\theta(t)$) monte, ce qui **stabilise et cristallise** les structures finales du réseau.

Ces combinaisons permettent d'affiner la stratégie d'apprentissage du SCN, en équilibrant les phases d'exploration et de stabilisation pour garantir une convergence plus efficace et une segmentation des clusters plus précise.

Conclusion

Le **seuil dynamique** $\theta(t)$ constitue un **outil puissant** pour gérer la **parsimonie** de manière **progressive**. En choisissant une croissance de θ bien dosée (linéaire, logarithmique, ou autre), on permet d'abord une **exploration** non bridée du **Synergistic Connection Network**, puis on procède à un **élagage** de plus en plus drastique. Mathématiquement, cette montée du **seuil** durcit le **test** :

$$\omega_{i,j}(t+1) \geq \theta(t+1) \Rightarrow (\text{conservation}),$$

ce qui affine la **sélection** des liens et **simplifie** la topologie au cours de l'auto-organisation. Au final, le **réseau** s'aboutit sur un **ensemble** de liaisons suffisamment fortes pour former des **clusters** stables et expressifs, tout en évitant la **surcharge** de connexions faibles.

7.4.3.3. Exemples : $\theta(t) = \theta_0 \cdot (1 + \beta t)$ ou Forme Logarithmique

Pour **encourager** la **parsimonie progressive** dans la mise à jour du Synergistic Connection Network (SCN), on peut faire **varier** le seuil θ au cours du temps, au lieu de le fixer définitivement (voir [7.4.3.2](#)). Plusieurs **lois** d'évolution sont possibles, allant d'une **croissance** linéaire à une **croissance** logarithmique ou exponentielle. L'objectif est de

commencer l'apprentissage avec un seuil **peu exigeant** (préservant la possibilité de diverses liaisons “moyennes”) pour, **progressivement**, le rendre plus **élevé** et ainsi sélectionner un nombre restreint de liens vraiment forts.

A. Forme Linéaire : $\theta(t) = \theta_0 (1 + \beta t)$

On pose

$$\theta(t) = \theta_0 (1 + \beta t),$$

où $\theta_0 > 0$ est la valeur **initiale** et $\beta \geq 0$ un **taux** de croissance. Lorsque t (le *temps* ou l'*itération*) augmente, $\theta(t)$ s’élève de façon **linéaire**, obligeant peu à peu chaque liaison $\omega_{i,j}(t)$ à dépasser un **seuil** de plus en plus haut pour rester active.

Au **démarrage** ($t \approx 0$), $\theta(t) \approx \theta_0$: les liaisons de force $\omega_{i,j} \approx \theta_0$ ou un peu plus restent **autorisées**.

Au **fur et à mesure** (t grandit), $\theta(t)$ grimpe, excluant nombre de liens “moyens”. Seules subsistent les $\omega_{i,j}$ nettement **au-dessus** du seuil.

Dans un **SCN** déjà influencé par d’autres mécanismes tels que **l’inhibition** ou **la saturation**, l’augmentation progressive de $\theta(t)$ joue un rôle structurant à deux niveaux.

L’augmentation graduelle de $\theta(t)$ permet de **préserver une phase d’exploration initiale** où le réseau maintient une **densité plus importante** de connexions. Durant cette période, les liens peuvent évoluer librement, ce qui favorise la formation de structures intermédiaires et l’exploration de différentes organisations possibles avant toute restriction trop stricte.

À mesure que $\theta(t)$ continue d’augmenter, un **tri sélectif** s’opère sur l’ensemble des connexions. En fin de processus, seules les liaisons qui ont **suffisamment grandi et résisté à la concurrence** sont conservées, tandis que les autres sont progressivement éliminées. Ce filtrage assure une structure finale optimisée, où les connexions maintenues sont les plus pertinentes et les plus stables face aux contraintes du SCN.

On ajoute à la fin de chaque itération la règle de *post-traitement* :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } \omega_{i,j}(t+1) \geq \theta(t+1), \\ 0, & \text{sinon.} \end{cases}$$

avec $\theta(t+1) = \theta_0 + \beta(t+1)$. Un β trop grand peut cependant *couper* rapidement de nombreux liens, tandis qu’un β trop petit laisse survivre longtemps beaucoup de connexions intermédiaires.

B. Forme Logarithmique : $\theta(t) = \theta_0 \ln(1 + \gamma t)$ ou $\theta_0 \ln(t + 2)$

Une **fonction logarithmique** ou quasi-logarithmique offre une **montée** moins brutale que la forme linéaire :

$$\theta(t) = \theta_0 \ln(1 + \gamma t) \quad \text{ou} \quad \theta_0 \ln(t + 2),$$

avec $\theta_0 > 0$ et $\gamma > 0$.

- Au tout début ($t \approx 0$), $\ln(1 + \gamma t) \approx \gamma t$, l’augmentation demeure modeste.
- Pour des valeurs de t plus élevées, $\theta(t)$ grandit toujours, mais **moins vite** qu’une fonction linéaire.

L’augmentation progressive de ($\theta(t)$) évite les **effets “ciseaux”**, empêchant le seuil de devenir **trop élevé trop vite** et laissant ainsi plus de temps aux liaisons pour se renforcer lorsque la synergie ($S(i,j)$) le justifie.

En phase finale, bien que le seuil atteigne une valeur conséquente, la **réduction des liens s’opère de manière progressive** plutôt que brutale, ce qui simplifie la structure tout en maintenant une transition plus fluide.

Exemples Numériques

- $\theta(t) = \theta_0 \ln(t + 2)$:

- À $t = 0$, $\theta(0) = \theta_0 \ln(2) \approx 0.693 \theta_0$.
- À $t = 100$, $\theta(100) = \theta_0 \ln(102) \approx 4.625 \theta_0$.
- À $t = 1000$, $\theta(1000) = \theta_0 \ln(1002) \approx 6.91 \theta_0$.
- $\theta(t) = \theta_0 \ln(1 + \gamma t)$: on peut régler γ pour accélérer ou freiner la *vitesse* de croissance du seuil.

C. Impact sur la Dynamique et Formation des Clusters

Quel que soit le *choix* (linéaire, log, exponentiel), la philosophe reste :

521. **Phase initiale** : $\theta(t)$ faible, \Rightarrow beaucoup de liaisons au-dessus du seuil, *exploration large* du réseau.
522. **Phase intermédiaire** : $\theta(t)$ commence à s'élever, \Rightarrow élimination des connexions qui ne se renforcent pas assez.
523. **Phase finale** : $\theta(t)$ est devenu nettement plus strict, \Rightarrow le réseau **s'épure** pour ne garder que les **clusters** solides.

Le **compromis** :

- Un $\theta(t)$ croissant **trop rapidement** risque de “décapiter” prématurément certains liens moyennement prometteurs (qui avaient besoin de plus de temps pour se renforcer).
- Un $\theta(t)$ croissant **trop lentement** laisse une densité importante de liens “moyens” fort longtemps, ce qui peut polluer la *structure* et *ralentir* la différenciation en clusters.

Dans la pratique, cette évolution adaptative du **seuil** complète bien d’autres mécanismes (inhibition locale, recuit simulé, etc.). En particulier, l’**inhibition** incite déjà chaque entité \mathcal{E}_i à privilégier quelques liaisons fortes ; le **seuil** dynamique “officialise” définitivement la coupure des liens faibles ou moyens, *après* un temps d’observation suffisant.

Conclusion

Des **lois** telles que :

$$\theta(t) = \theta_0 (1 + \beta t) \quad (\text{linéaire}) \quad \text{ou} \quad \theta_0 \ln(1 + \gamma t) \quad (\text{logarithmique}),$$

illustrent la façon dont on peut **programmer la croissance** d’un seuil θ au fil des itérations, pour aboutir à une **parsimonie progressive**.

524. **Linéaire** : la *tension* monte régulièrement, précipitant un tri plus *brusque* à mesure que t s’élève.

525. **Logarithmique** : la croissance est plus mesurée, aidant à ne pas couper trop tôt des liens en devenir.

Cette **flexibilité** du *seuil adaptatif* s'accorde bien avec la logique d'**auto-organisation** en Deep Synergy Learning : on permet d'abord aux entités de “tester” plusieurs connexions, puis on **resserre** progressivement la sélection, donnant naissance à des **clusters** plus stables et plus lisibles.

7.5. Méthodes Hybrides et Heuristiques

Au-delà du **recuit simulé** (7.3) et des mécanismes de **compétition avancée** (inhibition, saturation), il existe divers **schémas hybrides** ou **heuristiques** qui peuvent renforcer l'efficacité de la dynamique DSL et en améliorer la **convergence**. L'idée générale consiste à "greffer" des routines complémentaires (sparsification, algorithmes évolutionnaires, multi-run, etc.) sur le **cœur** de la mise à jour $\omega_{i,j}(t+1)$. Ces approches fournissent :

- 526. **Un contrôle** plus fin de la structure du SCN (par exemple, en limitant explicitement le nombre de liens),
- 527. **Des solutions** plus globales (en combinant la descente DSL avec des méthodes d'exploration plus large, comme les algorithmes génétiques),
- 528. **Une robustesse accrue** (réaliser plusieurs runs, fusionner, comparer, etc.).

7.5.1. Sparsification Contrôlée

La **sparsification** vise à **réduire** la densité de la matrice ω . Sans ce type de mécanisme, le réseau DSL pourrait rapidement compter un grand nombre de liens $\omega_{i,j}$ "moyennement forts", ce qui complique la **lecture** de la structure et accroît le **coût** de mise à jour. Une **sparsification** maintient les liaisons jugées essentielles et supprime (ou évite la formation de) celles dont la synergie est faible ou redondante.

7.5.1.1. k-NN Local : Chaque Entité ne Garde que k Liens Maximum

Au sein d'un **Synergistic Connection Network** (SCN), il est parfois **nécessaire** de contraindre la **densité** des connexions pour conserver une structure *lisible* et maîtriser la complexité algorithmique. Une approche standard consiste à imposer, pour chaque entité \mathcal{E}_i , un **maximum** de k liaisons sortantes. C'est ce qu'on appelle la **règle k-NN local** : localement, \mathcal{E}_i ne garde que ses k plus forts liens, et coupe les autres.

A. Principe k-NN Local

On considère un réseau comportant n entités. Après chaque étape de mise à jour des poids $\{\omega_{i,j}\}$ (via la règle DSL, éventuellement avec inhibition, voir 7.4), on applique un **filtrage** :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } j \in \text{TopK}(i, t+1), \\ 0, & \text{sinon.} \end{cases}$$

Ici, $\text{TopK}(i, t+1)$ est l'ensemble des k *indices* j pour lesquels $\omega_{i,j}(t+1)$ est la plus grande (ou parmi les plus grandes) après la mise à jour. Autrement dit, on **trie** tous les liens $\omega_{i,j}$ sortant de \mathcal{E}_i par ordre décroissant et ne **conserve** que les k meilleurs, mettant les autres à 0.

L'approche permet de **réduire la complexité** en bornant à k le degré sortant de chaque entité, limitant ainsi la densité totale du réseau à $O(nk)$ au lieu de $O(n^2)$. Cette contrainte favorise une **sélectivité accrue**, où chaque nœud \mathcal{E}_i ne conserve que les liaisons **les plus synergiques**, évitant l'accumulation de connexions de force moyenne. Sur le plan **visuel et algorithmique**, le SCN devient plus **épars**, facilitant ainsi son analyse et sa stabilisation.

Cependant, cette méthode comporte certains **écueils potentiels**. Un lien $\omega_{i,j}$ encore en phase de consolidation peut être **supprimé prématurément** s'il est jugé trop faible à un instant t , alors qu'il aurait pu se renforcer avec le temps. Le choix du **paramètre k** est également critique : une valeur trop faible limite la formation de clusters, tandis qu'une valeur trop élevée **ne réduit pas suffisamment la densité**, maintenant un SCN surchargé.

B. Motivations Mathématiques

Sans mécanisme de coupe, un DSL peut générer un grand nombre de liens moyens. Or, traiter $O(n^2)$ liaisons devient coûteux pour le calcul de la mise à jour et de l'inhibition. Le k-NN local agit donc comme un **réducteur** de complexité : chaque entité \mathcal{E}_i maintient un degré sortant de $\leq k$. Le graphe global a ainsi au plus $O(nk)$ liaisons non nulles.

D'un point de vue **théorique**, imposer que chaque noeud "choisisse" seulement k liens se rapproche d'une **construction** k-NN usuelle dans les graphes de similarité. On élimine les valeurs **moyennes** ou **faibles** parce qu'elles n'apportent pas un gain notable à la structure, tout en alourdisant la cohérence du réseau. La **parsimonie** ainsi acquise simplifie la mise en évidence de **clusters** : seuls quelques contacts "préférentiels" par entité suffisent.

La **règle** de k-NN local n'est pas une descente de gradient "pure" : elle introduit un **post-traitement** ou "projection" après le calcul de $\omega_{i,j}(t+1)$. Cela peut être vu comme un **algorithme** de type "proximal" en optimisation, où l'on force la *solution* à être *k-sparse* pour chaque ligne. En pratique, cette étape **améliore** la stabilité et la lisibilité, **accélérant** souvent la convergence vers une structure de clusters *plus distincts*.

C. Implication pour la Formation de Clusters

En imposant qu'un noeud \mathcal{E}_i ne garde que k liens, on **accentue** la "sélection" de ses partenaires préférentiels. Les entités tendent à former de **petits groupes** bien reliés, plutôt que de disperser leurs poids sur de nombreux voisins. Ce phénomène renforce la **cohésion** de clusters émergents.

Si $k \approx 1$ ou 2, le réseau devient un ensemble de liaisons quasi *arborescentes*, peut-être trop pauvre pour détecter des *clusters* riches.

Si $k \approx n$, on ne gagne plus grand-chose : il n'y a quasiment pas de coupe.

Typiquement, on **dimensionne** k comme $O(\log(n))$ ou $O(\sqrt{n})$ selon la taille du réseau et le degré de sélectivité souhaité.

Avec un **réseau** dont le nombre d'arêtes est $O(nk)$, la **mise à jour** $\omega_{i,j}(t+1)$ devient plus **rapide** car on ne traite *réellement* que les liens non nuls. L'inhibition latérale ($\sum_{k \neq j} \omega_{i,k}$) est aussi plus concise. On obtient un **gain** substantiel pour des systèmes de grande dimension.

Exemple Numérique Minimal

Supposons un SCN de taille $n = 10$ et choisissons $k = 3$. Après chaque cycle de mise à jour (inhibition, etc.), pour chaque entité \mathcal{E}_i , on :

529. **Trie** $\{\omega_{i,j}\}_{j=1..10}$ par ordre décroissant.

530. **Conserve** les 3 plus grands liens.

531. **Met à zéro** les 6 autres.

Le **réseau** obtenu affiche alors au plus $10 \times 3 = 30$ liaisons, voire moins si certaines $\omega_{i,j}$ sont déjà nulles. En termes de **clusters**, chaque \mathcal{E}_i est "connecté" aux 3 voisins pour qui la synergie s'est montrée la plus forte. On répète ce filtrage à chaque itération, favorisant au fur et à mesure l'émergence de groupes soudés.

Conclusion

La **règle k-NN local** impose que, pour chaque entité \mathcal{E}_i , seules k liaisons soient **actives** (celles de poids le plus élevé) et les autres soient **coupées** (mises à 0). Bien que cela sorte du cadre d'une descente d'énergie purement "continuelle", c'est une **approche heuristique** très efficace, car :

532. Elle **limite** la densité, passant de $O(n^2)$ à $O(nk)$.

533. Elle **clarifie** le SCN, renforçant la **sélectivité** et accélérant la formation de **clusters**.

534. Elle **facilite** la mise à jour, notamment si on combine cette coupure de liens à des mécanismes d'inhibition (moins de sommes à calculer).

Le **choix** de k dépend du **niveau** de parcimonie recherché, de la taille n et du degré de connectivité souhaité dans les clusters. Dans la suite (7.5.2, 7.5.3), on examinera d'autres heuristiques d'élagage visant des effets similaires ou complémentaires pour l'auto-organisation du **Deep Synergy Learning**.

7.5.1.2. Impact sur la Formation de Clusters et la Rapidité de Mise à Jour

L'adoption d'une **règle k-NN local** (voir § 7.5.1.1) exerce une influence notable sur la **formation de clusters** et sur la **vitesse** de la mise à jour dans le cadre du **Deep Synergy Learning** (DSL). L'idée de ne conserver, pour chaque entité \mathcal{E}_i , qu'un nombre **maximal** k de liaisons induit à la fois une simplification de la **topologie** du Synergistic Connection Network (SCN) et une **réduction** du nombre de poids $\omega_{i,j}$ réellement manipulés. Les effets de cette coupure sélective se mesurent tout particulièrement dans la dynamique d'**auto-organisation** des clusters et dans la **rapidité** avec laquelle le réseau converge.

A. Clarification de la Formation de Clusters

La **règle** consistant à imposer, après chaque mise à jour, que seules les k liaisons $\omega_{i,j}$ les plus grandes soient conservées (et les autres ramenées à zéro) aboutit à une **structuration** plus nette des groupes. Dans un **SCN**, les entités cherchant à renforcer leurs liens avec les voisins les plus "synergiques" (voir § 2.2 sur la notion de synergie $S(i,j)$), le fait de fixer un **seuil** sur le **nombre** de connexions autorisées accroît la **sélectivité** : un nœud \mathcal{E}_i ne peut s'attacher solidement qu'à un ensemble restreint de cibles, ce qui favorise la création de **sous-réseaux** cohérents.

Sur le plan **mathématique**, la conservation de seulement k liaisons revient à imposer, à l'issue de la mise à jour $\omega_{i,j}(t+1)$, la projection

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } j \in \text{TopK}(i, t+1), \\ 0, & \text{sinon.} \end{cases}$$

On **incite** donc chaque nœud \mathcal{E}_i à privilégier un petit nombre de liens **forts**, plutôt que de diffuser ses ressources sur de multiples connexions moyennes. Les entités ayant des affinités fortes tendent à se regrouper en **clusters** plus "lisibles". Les liaisons **intra-cluster** se renforcent, alors que les liaisons **inter-cluster** trop faibles sont coupées lors du tri, ce qui accentue la **compartimentation**.

Il s'ensuit que la lecture des **composantes** dans le graphe final devient plus aisée : on repère rapidement des **sous-groupes** de nœuds fortement interconnectés, sans que le graphe ne s'encombre de nombreuses arêtes de faible amplitude. La **qualité** de la partition en clusters peut ainsi s'en trouver améliorée, surtout si la valeur de k est adaptée aux **dimensions** et à la **diversité** des entités.

B. Rapidité de Mise à Jour Accrue

Réduire le **degré** sortant de chaque nœud à k liaisons non nulles allège considérablement le travail de mise à jour, car le réseau ne comporte plus $O(n^2)$ arêtes mais plutôt $O(nk)$. Lorsque le **Deep Synergy Learning** s'exécute à chaque itération, la pondération $\omega_{i,j}(t+1)$ est ajustée selon une **règle** de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] \pm \dots$$

ou encore intégrant un terme d'**inhibition** (cf. § 7.4). En pratique, si un grand nombre de $\omega_{i,j}$ sont fixées à zéro, on gagne un **facteur** notable sur le coût de calcul : la mise à jour locale ne concerne plus que les liens effectivement **actifs** (voir également § 2.2.3 pour les arguments relatifs à la parcimonie).

Ce **gain** est d'autant plus crucial dans des réseaux de grande taille, où la matrice $\{\omega_{i,j}\}$ pourrait autrement contenir plusieurs dizaines de millions de poids. Avec la règle k-NN local, chaque entité \mathcal{E}_i ne manipule que k liens sortants, en réduisant proportionnellement le **temps** de parcours et d'inversion éventuelle pour calculer des sommes comme $\sum_{k \neq j} \omega_{i,k}(t)$.

La plus grande **rapidité** de la mise à jour stimule également la **convergence** du SCN : en limitant la “pollution” par des poids de petite amplitude, on raccourcit le temps nécessaire pour que les liens forts s’établissent et que les liens faibles soient éliminés. Les entités **s’orientent** plus vite vers les voisins les plus adaptés, stabilisant la **structure** de clusters. Bien qu’il y ait un risque de “couper” trop tôt certaines liaisons naissantes, ce biais est souvent compensé par la facilité de détection des **clusters** dominants.

Conclusion

La **règle k-NN local**, évoquée en § 7.5.1.1, ne se contente pas de **réduire** la densité du réseau : elle influence directement la **formation** des clusters et la **vitesse** d’itération du Deep Synergy Learning. En restreignant chaque noeud à ses k meilleurs partenaires, on **améliore** la lisibilité de la partition en sous-groupes, tout en **accélérant** la mise à jour des poids, puisqu’on ne traite effectivement qu’un nombre limité de liaisons à chaque cycle. Cette conjonction entre **sélectivité** et **efficacité** rend le k-NN local particulièrement attractif dans des environnements de grande dimension, pour lesquels la **complexité** $O(n^2)$ se révèle souvent trop lourde. Le paramètre k doit être choisi de sorte à maintenir assez de **liberté** pour l’émergence des clusters, sans abaisser trop fortement la connectivité essentielle au sein du SCN.

7.5.2. Approches de Type Genetic Algorithm

Dans la recherche de **méthodes heuristiques** pour échapper aux minima locaux ou explorer l’espace $\{\omega_{i,j}\}$ de manière plus **globale** (voir §§7.2.2.3, 7.3.1, 7.4.2), les **algorithmes génétiques** (Genetic Algorithms, GA) fournissent un cadre relativement simple et puissant. Ils consistent, rappelons-le, à faire **évoluer** une population de *solutions* (ici, des structures de pondérations ω) en leur appliquant des **opérateurs** de mutation, de crossover, etc., tout en sélectionnant les plus “adaptés” (ceux qui minimisent le mieux la fonction d’énergie \mathcal{J} ou maximisent un critère de cohésion du SCN).

7.5.2.1. Codage d’une “solution” = structure de ω

Dans une **approche génétique** (voir la section 7.5.2 dans son ensemble), il est nécessaire de définir la façon dont un **individu** (ou *solution candidate*) encode la **structure** du réseau, c’est-à-dire l’ensemble des **pondérations** $\omega_{i,j}$ dans un **Synergistic Connection Network** (SCN). Ce **codage** doit être suffisamment général pour représenter n’importe quelle configuration $\{\omega_{i,j}\}$, mais aussi suffisamment **concise** pour permettre la mise en œuvre pratique des opérations de **croisement** et de **mutation** (voir § 7.5.2.2).

A. Principe du Codage Génétique

Dans un **algorithme génétique** (AG) standard, on manipule une **population** \mathcal{P} d’**individus**, chaque individu représentant une *configuration possible* du SCN. Cette configuration correspond aux **valeurs** des liaisons $\omega_{i,j}$. Un individu peut donc se voir comme un **vecteur** \mathbf{W} résumant toutes les composantes $\omega_{i,j}$ pour $1 \leq i < j \leq n$.

Dans le cas où le réseau est **orienté**, on considère plutôt la totalité des paires (i, j) avec $i \neq j$. Mathématiquement, cela donne une **dimension** $\dim \approx n(n - 1)$, qui peut atteindre $O(n^2)$. L’individu peut alors être un **tableau** (ou une **chaîne**) de longueur $O(n^2)$.

Lorsque les pondérations $\omega_{i,j}$ sont de nature **réelle**, il est possible d’utiliser un **encodage** en valeurs flottantes. Si l’on reste dans un **cadre** d’algorithmes génétiques plus “classiques”, on peut vouloir discréteriser ou quantifier chaque $\omega_{i,j}$ sur un certain nombre de **bits**, créant ainsi un individu en **codage binaire**. On retrouve alors une structure du type

$$\mathbf{W} = \begin{pmatrix} b_{1,2}^1, \dots, b_{1,2}^q, \dots, b_{i,j}^1, \dots, b_{i,j}^q, \dots \\ \text{bits de } \omega_{1,2} \quad \quad \quad \text{bits de } \omega_{i,j} \end{pmatrix},$$

où q représente la **précision** choisie pour chaque pondération.

B. Représentation Binaire, Réelle ou Hybride

En **algorithmes génétiques**, la **représentation binaire** demeure très commune, car elle facilite l'implémentation d'opérateurs de **croisement** et de **mutation** au niveau des bits (voir § 7.5.2.2). Toutefois, dans le cadre d'un **Deep Synergy Learning** (DSL) où $\omega_{i,j}$ est typiquement un **réel positif** (ou nul), il est tout à fait possible d'employer un **AG à valeurs réelles** : dans ce cas, chaque individu manipule directement les $\omega_{i,j} \in \mathbb{R}$.

Sur le plan **mathématique**, on se situe alors dans \mathbb{R}^{dim} avec $\text{dim} \approx n^2$. L'opérateur de mutation correspond à une **perturbation gaussienne** ou uniforme appliquée à chaque $\omega_{i,j}$, tandis que le croisement combine par exemple de façon **arithmétique** ($\alpha \omega_{i,j}^{(\text{père})} + (1 - \alpha) \omega_{i,j}^{(\text{mère})}$) les valeurs parentales.

C. Fonction d'Évaluation : Énergie ou Cohésion de Clusters

Pour **évaluer** la **qualité** d'une solution \mathbf{W} , on définit une **fonction de coût** $J(\mathbf{W})$ reflétant l'objectif du DSL : par exemple, on peut juger du **niveau d'auto-organisation**, de la **cohérence** en clusters ou de l'**énergie** associée à la synergie (voir le chapitre 2.2 pour les formules d'énergie usuelles).

Une approche classique consiste à considérer la **fitness** d'un individu sous la forme :

$$\text{Fitness}(\mathbf{W}) = \frac{1}{1 + J(\mathbf{W})} \quad \text{ou} \quad -J(\mathbf{W}),$$

de sorte que **minimiser** $J(\mathbf{W})$ équivaut à **maximiser** la fitness.

Les **individus** de l'algorithme génétique se voient ainsi **comparés** selon la valeur de $J(\mathbf{W})$, et les "meilleurs" (ceux affichant un plus bas coût) sont privilégiés lors de la **sélection** (voir § 7.5.2.2).

D. Exemple Numérique Minimal

Considérons un **SCN** où $n = 4$. On suppose le réseau *non orienté*, de sorte qu'il suffit de coder $\binom{4}{2} = 6$ pondérations $\omega_{1,2}, \omega_{1,3}, \omega_{1,4}, \omega_{2,3}, \omega_{2,4}, \omega_{3,4}$. Un **individu** \mathbf{W} dans ce GA se présentera sous la forme :

$$\mathbf{W} = (\omega_{1,2}, \omega_{1,3}, \omega_{1,4}, \omega_{2,3}, \omega_{2,4}, \omega_{3,4}).$$

Si l'on choisit un **codage binaire** (8 bits par composante), cela donne une **chaîne** de $6 \times 8 = 48$ bits. Chaque manipulation génétique (crossover, mutation, etc.) portera sur cette chaîne, modifiant des **blocs** (ou des bits) correspondant à telle ou telle liaison $\omega_{i,j}$.

E. Intégration avec la Dynamique DSL

Il est possible d'utiliser cette **codification** de ω dans un *Algorithme Génétique pur*, où la mise à jour n'est faite que par les opérateurs GA (sélection, croisement, mutation). Néanmoins, on peut aussi **combiner** l'AG et la **descente locale** (ou la **règle** DSL), de sorte que chaque individu "s'améliore" localement via un mini-cycle DSL avant d'être réincorporé dans la population. Ceci est un **schéma hybride** :

535. On part d'un individu \mathbf{W} .

536. On applique quelques pas de la règle **DSL** ($\omega_{i,j}(t+1) = \dots$) pour obtenir un **minimum local** \mathbf{W}^* .

537. On code \mathbf{W}^* en binaire ou en réel pour le **soumettre** à la population du GA.

De cette façon, le GA gère la **diversité** globale et la recombinaison, tandis que la **descente DSL** agit comme un **opérateur** de "raffinement" local, ce qui peut accélérer la **convergence** vers une structure optimisée.

Conclusion

Dans le cadre d'un **algorithme génétique** appliqué au **Deep Synergy Learning**, la **configuration** des pondérations $\{\omega_{i,j}\}$ doit être **encodée** sous une forme adaptée : un **vecteur** (ou une **chaîne**) contenant les valeurs des liaisons. Ce

codage peut être **binaire** ou **réel**, selon qu'on souhaite exploiter les opérateurs GA classiques sur bits ou des versions continues. Chaque individu se voit alors **évaluer** via une **fonction** d'énergie \mathcal{J} (ou un indice de clusterisation), permettant de **sélectionner** les meilleurs, de **crosser** (mélanger) les gènes et de **muter** (modifications aléatoires). Ce mécanisme offre un **moyen** complémentaire de sortir des minima locaux, en **recombinant** efficacement des solutions issues de la dynamique DSL. L'**intégration** entre la mise à jour DSL et le GA (voir § 7.5.2.2 pour les détails sur le crossover/mutation) peut ainsi procurer des performances **robustes**, alliant l'exploration globale d'un GA à la **plasticité** locale d'un SCN.

7.5.2.2. Opérateurs de Mutation (Ajout/Rajeunissement de Liens) et Crossover (Mélange de Deux Structures)

Les algorithmes génétiques appliqués au **Deep Synergy Learning** (DSL) consistent à faire évoluer un **ensemble** de configurations $\{\omega_{i,j}\}$ (voir § 7.5.2.1) en leur appliquant des “métaphores” inspirées de la **sélection naturelle**. Chaque **individu** est une représentation d'un SCN (Synergistic Connection Network) complet, c'est-à-dire une matrice ou un vecteur de pondérations. Pour **faire progresser** la population d'itération en itération, on définit deux **grands** opérateurs : la **mutation**, qui modifie localement les liaisons, et le **crossover**, qui combine deux réseaux parents en un nouvel enfant.

A. Rappel : Codage d'un “individu”

Un *individu* est une **structure** $\omega = \{\omega_{i,j}\}$, soit sous forme d'une **matrice** $n \times n$ (cas orienté) ou d'un **vecteur** de dimension $n(n - 1)/2$ (cas non orienté). Chaque individu possède une **fitness**, par exemple $\text{Fitness}(\mathbf{v}) = -\mathcal{J}(\mathbf{v})$, où \mathcal{J} désigne la **fonction d'énergie** mesurant la qualité des clusters ou le degré de synergie (voir § 2.2 sur la définition de l'énergie).

D'un point de vue **algorithmique**, on maintient une **population** de taille M : $\mathcal{P} = \{\Omega^{(1)}, \dots, \Omega^{(M)}\}$. À chaque **génération**, on **évalue** la fitness de chaque $\Omega^{(p)}$, on **sélectionne** (selon leur score) quels individus vont se reproduire, puis on leur applique des **opérateurs** de mutation et de crossover. Au terme de ces modifications, on obtient une **nouvelle** population, qu'on évalue à nouveau, et ainsi de suite jusqu'à convergence ou épuisement des ressources de calcul.

La **mutation** et le **crossover** définissent l'essence même d'un **algorithme génétique** : ils permettent d'explorer l'espace $\{\omega_{i,j}\}$ de façon plus large qu'une simple descente locale (cf. § 7.5.2.1).

B. Opérateurs de Mutation

La **mutation** a pour but d'introduire des **modifications** locales et **aléatoires** dans la structure $\{\omega_{i,j}\}$. À chaque génération, on sélectionne certains individus (ou certaines liaisons) pour leur infliger des changements stochastiques, de manière à maintenir la **diversité** dans la population et éviter un blocage dans un unique minimum local.

Dans un **SCN**, la mutation peut prendre la forme d'une **altération** directe de la pondération $\omega_{i,j}$. On peut :

- 538. **Ajouter** (ou **amplifier**) un lien qui était proche de zéro, en lui assignant une valeur positive, afin de “tester” de nouvelles connexions entre deux entités (i, j) .
- 539. **Supprimer** (ou **dégrader**) un lien existant, en ramenant $\omega_{i,j}$ à zéro ou près de zéro. Cela simule la disparition d'une connexion précédemment considérée.
- 540. **Rajeunir** un lien, c'est-à-dire perturber sa valeur $\omega_{i,j}$ de manière aléatoire (par exemple en l'augmentant ou en la réduisant de façon stochastique). Si $\omega_{i,j}$ était jusqu'alors moyen, cette opération peut le rendre plus fort ou plus faible et influer sur la cohésion du cluster.

On peut décrire la **mutation** d'une liaison $\omega_{i,j}$ par une équation du type

$$\omega_{i,j}^{(\text{new})} = \begin{cases} 0, & \text{avec une probabilité } p_{\text{cut}}, \\ \omega_{i,j}^{(\text{old})} + \delta, & \text{avec une probabilité } p_{\text{mut}}, \\ \omega_{i,j}^{(\text{old})}, & \text{sinon.} \end{cases}$$

où δ est un échantillon tiré d'une distribution gaussienne (ou autre) centrée en zéro, et où p_{cut} et p_{mut} sont des probabilités de "couper" ou de "rajeunir" la liaison. Les valeurs δ peuvent être proportionnelles à $\omega_{i,j}^{(\text{old})}$ ou indépendantes, selon la conception choisie.

Un **taux de mutation** trop élevé peut causer une **instabilité** importante : les solutions bougent sans cesse et peinent à converger. Un taux trop faible peut laisser la population se **spécialiser** trop vite autour d'un minimum local. Il existe donc un **compromis** : on veut **maintenir** une source de perturbation constante, tout en laissant aux bons schémas de liaisons $\omega_{i,j}$ la possibilité de se stabiliser.

C. Opérateur de Crossover (Mélange de Deux Structures)

Le **crossover** mélange deux "parents" (deux configurations $\Omega^{(A)}$ et $\Omega^{(B)}$) pour générer un "enfant" $\Omega^{(C)}$. Il s'agit de transférer des "morceaux" de la structure de A et B au sein de C, dans l'espoir de **combiner** les points forts des deux parents.

Si $\omega_{i,j}^{(A)}$ et $\omega_{i,j}^{(B)}$ sont les poids sur la liaison (i,j) dans les deux parents, on peut définir

$$\omega_{i,j}^{(C)} = \begin{cases} \omega_{i,j}^{(A)}, & \text{avec probabilité 0.5,} \\ \omega_{i,j}^{(B)}, & \text{avec probabilité 0.5,} \end{cases}$$

afin de tirer chaque lien de l'un ou l'autre parent. Dans un **crossover** plus évolué (dit arithmétique, par exemple), on prend un **mélange** linéaire :

$$\omega_{i,j}^{(C)} = \alpha \omega_{i,j}^{(A)} + (1 - \alpha) \omega_{i,j}^{(B)},$$

pour un certain $\alpha \in [0,1]$. Cela génère une interpolation entre les valeurs parentales. De la sorte, des clusters solides présents chez le premier parent peuvent coexister avec d'autres clusters dominants présents chez le second parent, si la répartition de liaisons ne se contredit pas trop.

Dans la pratique, on peut aligner **toutes** les composantes $\omega_{i,j}$ dans un **vecteur** de dimension $O(n^2)$, puis réaliser un "**one-point crossover**" classique : on choisit un index κ et on prend les composantes en-dessous de κ chez le parent A, et celles au-delà de κ chez le parent B. Ou l'on peut effectuer un **deux-points** (coupures multiples dans la liste). Parfois, un **post-traitement** est nécessaire pour éviter des incohérences (liens négatifs, saturations, etc.).

Le **crossover** se veut un opérateur **massif** : alors que la mutation agit sur un *petit* nombre de liaisons, le croisement élabore un **nouveau** réseau en prenant *large* des schémas de liaisons entiers, si possible de bonne qualité. En combinant ainsi deux solutions, l'**algorithme génétique** peut explorer plus loin que la somme de deux descentes locales, espérant recombiner des *clusters* cohérents provenant de chaque parent.

D. Rôle dans la Recherche Globale et Combinaison avec le DSL

Dans un **algorithme génétique** complet, ces opérateurs mutation et crossover s'appliquent à **chaque génération**, sur la population en cours, afin de **générer** une nouvelle population. Les liens $\omega_{i,j}$ se voient modifiés de deux manières : localement (mutation aléatoire) et globalement (mélange entre individus). Ceci permet d'éviter un enfermement dans un **minimum local** (voir chap. 7.2 sur les risques de convergence prématurée), et de découvrir des structures singulières où deux clusters forts issus de parents différents se retrouvent fusionnés chez l'enfant.

Dans certains schémas **hybrides**, on peut imbriquer la **règle DSL** (mise à jour auto-organisée) et l'**AG** :

541. Chaque individu subit quelques itérations de la descente DSL, pour parfaire localement sa matrice ω .

542. Les meilleurs individus sont ensuite pris pour opérer un crossover, une mutation.

543. On reforme la population et l'on réapplique la descente DSL.

Cette alternance rend la **recherche globale** plus efficace, fusionnant le “tamis” local du DSL et la large exploration combinatoire de l'**AG**.

Conclusion

Au sein d'un **Synergistic Connection Network**, le **couple** mutation-crossover constitue le **moteur** de la démarche génétique :

- La **mutation** agit comme un **ajustement** local (ajout, suppression, revalorisation des liens), prévenant la stagnation et entretenant la **diversité**.
- Le **crossover** permet un **mélange** de deux configurations parents, favorisant une exploration plus large de l'espace $\{\omega_{i,j}\}$ et la **création** de nouvelles combinaisons de clusters.

Cette **double** dynamique (perturbation fine, mixage global) augmente la probabilité de dénicher des **solutions** globalement optimales, moins susceptibles de rester bloquées dans des minima locaux. La **logique** génétique s'ajoute donc aux règles d'auto-organisation DSL en offrant un cadre d'**évolution** populationnelle, où la compétition entre individus se substitue à la seule descente locale, et où l'on peut découvrir des configurations originales par **recombinaison** de structures partielles déjà performantes.

7.5.2.3. Coexistence avec la dynamique DSL ?

Même lorsqu'on adopte un **algorithme génétique** (ou une autre heuristique globale) pour guider la structure de $\{\omega_{i,j}\}$, on ne souhaite pas pour autant **abandonner** la logique d'auto-organisation propre au **DSL**. Il s'agit plutôt d'une **cohabitation** : la dynamique DSL continue de procéder à ses mises à jour locales, tandis que l'algorithme heuristique assure périodiquement (ou en parallèle) une **exploration** plus large dans l'espace des pondérations.

A. Principe de la Coexistence

En **mode alterné**, on peut définir un cycle d'itérations DSL (disons T étapes) durant lesquelles $\omega_{i,j}$ évoluent selon la règle locale $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$. Puis, à la fin de ces T itérations, un **module génétique** vient “relire” l'état Ω et produire quelques “mutations” ou “crossovers” afin de créer de nouvelles configurations.

En **mode parallèle**, on peut imaginer un thread ou un processus secondaire qui, à intervalles réguliers, évalue la “qualité” de la solution courante et opère des modifications globales, pendant que la dynamique DSL se poursuit sur un autre thread. L'enjeu est de **synchroniser** les écritures dans Ω (accès concurrent aux liens $\omega_{i,j}$).

Le **DSL** prend en charge la **descente** ou la stabilisation locale (7.2.2.1), permettant de raffiner la structure d'un cluster ou de renforcer des liens pertinents.

Les **heuristiques globales** (dont l'algorithme génétique) servent à **s'échapper** des configurations trop figées (7.2.2.2) en “brassant” plus largement l'espace $\{\omega_{i,j}\}$. Par exemple, en clonant plusieurs solutions (états de Ω), en les “croisant” et en “mutant” des parties significatives du réseau.

B. Mécanisme pratique d'une itération “mixte”

À un instant t , on évalue la fonction d'énergie $J(\Omega(t))$ ou un critère dérivé (ex. modularité, cohésion, etc.). Si la solution est satisfaisante, on la marque comme “individu” de bonne qualité dans la **population** gérée par l'algorithme génétique.

Mathématiquement, on peut enregistrer la matrice $\Omega(t)$ entière comme un **chromosome C**.

L'**AG** (algorithme génétique) sélectionne deux “individus” (matrices $\Omega^{(p)}, \Omega^{(q)}$) et opère un “crossover” :

$$\Omega_{i,j}^{(\text{child})} = \begin{cases} \Omega_{i,j}^{(p)}, & \text{avec probabilité 0.5,} \\ \Omega_{i,j}^{(q)}, & \text{avec probabilité 0.5,} \end{cases}$$

ou un schéma plus sophistiqué (moyenne, bloc...). Puis on **mute** certains liens $\Omega_{i,j} \leftarrow \Omega_{i,j} + \delta$ aléatoirement.

On peut *remplacer* la configuration courante du SCN par $\Omega^{(\text{child})}$ ou **mixer** les deux (ex. partiellement). Ensuite, la **dynamique DSL** reprend son cours local, consolidant les liens cohérents et affaiblissant ceux incohérents.

But : si la solution enfant se révélait “mauvaise”, la descente DSL la rectifiera rapidement ; si elle recèle un potentiel global plus élevé (clusters inédits), DSL renforcera ce potentiel.

C. Avantages et Considérations Mathématiques

Au lieu de se borner à la descente DSL, l'AG injecte une **diversité** : la “progéniture” d'individus différents peut **explorer** des régions de l'espace $\{\omega_{i,j}\}$ que la descente locale n'aurait jamais visitées. Cela **relance** la recherche à un **niveau** macro.

Sur le plan **mathématique**, si l'on *trop souvent* remplace Ω par des mutations radicales, la dynamique locale DSL perd son effet de raffinement. On doit régler la **périodicité** du “crossover/mutation” et l'intensité des modifications pour ne pas se retrouver dans un état de chaos permanent.

Le **DSL** agit comme un **raffineur** local (sorte de “gradient descent”), tandis que l'**algorithme génétique** tient lieu de “métaheuristique” globale. Ensemble, ils forment une **stratégie** de type “memetic algorithm” (dans la littérature en optimisation) où la “phase locale” (DSL) perfectionne chaque individu, et la “phase globale” (AG) génère de nouvelles configurations à tester.

Exemples Numériques

Mini-réseau : sur un jeu de 15 entités, la descente DSL seule peut aboutir à 2 clusters, mais parfois coincés dans une répartition déséquilibrée. L'AG, en “croisant” 2 solutions stables distinctes, obtient un enfant potentiellement plus adapté, que DSL localement consolide, découvrant un troisième cluster plus fin.

Validation : on compare la fonction $J(\Omega)$ ou la “modularité” de la solution avant et après l'injection de l'hybride. Souvent, on observe un **gain** par rapport à l'usage unique d'une descente locale.

Conclusion

La **coexistence** entre un **algorithme génétique** (ou autre heuristique globale) et la **dynamique DSL** combine :

544.**Descente locale** : le SCN, via sa mise à jour $\omega(t+1) = \dots$, affine la configuration,

545.**Exploration globale** : l'AG (ou la colonie de fourmis, ou tout autre) brasse l'espace $\{\omega\}$ de façon plus large,

546.**Phase d'interaction** : régulièrement (ou en parallèle), l'AG propose une “mutation” ou un “crossover” qui **remanie** la matrice Ω , puis la descente DSL localement stabilise ou rejette la nouveauté.

Ce **mélange** d'un algorithme local (DSL) et d'un algorithme global (AG) vise à obtenir le **meilleur** des deux mondes : la **réactivité** et la **robustesse** face aux minima locaux. Au final, on dispose d'une **méthode** plus puissante pour parvenir à des solutions auto-organisées de meilleure qualité, tout en préservant la **simplicité** de la dynamique DSL au sein de chaque étape locale.

7.5.3. Recherche Multi-Début ou Multi-Run

Dans de nombreux problèmes d'optimisation complexes (y compris ceux régis par la mise à jour DSL), la dynamique de descente locale (ou pseudo-descente) peut dépendre fortement des **conditions initiales**. Un moyen de **limiter** le risque d'enfermement dans un minimum local défavorable consiste à recourir à des **initialisations multiples** (multi-début) ou des **exécutions multiples** (multi-run). L'idée est de lancer la même dynamique DSL un certain nombre de

fois, depuis des points $\{\omega_{i,j}(0)\}$ différents (ou avec des perturbations initiales distinctes), puis de **comparer** ou **combiner** les solutions obtenues pour accroître les chances de trouver (ou d'approcher) un minimum plus global de la fonction d'énergie \mathcal{J} .

7.5.3.1. Exécuter la Dynamique DSL Plusieurs Fois avec des Initialisations Différentes

Dans le **Deep Synergy Learning** (DSL), la fonction d'énergie $\mathcal{J}(\Omega)$ (voir chap. 7.2.1) peut présenter des **multiples** attracteurs ou minima locaux (cf. § 7.2.2). Il est alors naturel de se prémunir contre un **verrouillage** précoce en lançant plusieurs **exécutions** (ou “runs”) de la dynamique DSL, chacune depuis une **initialisation** distincte des poids $\omega_{i,j}$. Cette approche, dite de *multi-run* ou *multi-start*, vise à explorer différentes régions de l'espace $\{\omega\}$, afin d'augmenter les chances de **découvrir** un minimum local plus profond, voire global, qu'un unique run n'atteindrait pas.

A. Principe du Multi-Run

Le **principe** consiste à exécuter plusieurs fois l'algorithme DSL, qui met à jour la **matrice** $\{\omega_{i,j}(t)\}$ en fonction des règles habituelles (voir § 2.2.2 ou § 7.2) :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] \pm \dots$$

ou une variante intégrant l'**inhibition** (chap. 7.4) ou des mécanismes spécifiques (chap. 7.2.3). À la fin de chaque run k , on obtient une configuration finale $\Omega_{\text{final}}^{(k)}$. On répète ainsi :

$$\text{Run } k \rightarrow \{\omega_{i,j}(t)\}_{t=0}^{T_k} \text{ avec initialisation } \omega_{i,j}^{(k)}(0).$$

Les **initialisations** $\omega_{i,j}^{(k)}(0)$ diffèrent d'un run à l'autre, ce qui permet de **couvrir** plusieurs bassins d'attraction. L'évolution DSL étant sensible aux conditions initiales (espace potentiellement non convexe), on espère ainsi **échapper** à un minimum local trop restreint en choisissant de “nouvelles” portes d'entrée dans l'espace $\{\omega\}$.

B. Initialisation Aléatoire ou Stratégique

Le **choix** des conditions initiales $\omega_{i,j}^{(k)}(0)$ peut se faire de diverses manières :

Aléatoire (uniforme ou gaussien).

On peut tirer chaque liaison $\omega_{i,j}^{(k)}(0)$ de façon **uniforme** dans un intervalle $[-\delta, \delta]$ ou $[0, \delta]$ si l'on souhaite rester dans des pondérations positives. Cette méthode maximise la **diversité** des départs, autorisant des trajectoires d'évolution très variées du SCN.

Biaisé par la géométrie ou la similarité.

Si l'on connaît une **structure** latente (par exemple un embedding $\mathbf{x}_i \in \mathbb{R}^d$ pour chaque entité \mathcal{E}_i), on peut initialiser $\omega_{i,j}^{(k)}(0)$ en fonction d'une distance $\|\mathbf{x}_i - \mathbf{x}_j\|$. Les runs diffèrent alors par des **perturbations** stochastiques sur ces valeurs. Cela donne un point de départ déjà “cohérent” avec la topologie du problème, tout en conservant la chance de s'en éloigner un peu.

Réutiliser d'anciennes solutions + “Shake”.

À la fin d'un run, on obtient $\Omega_{\text{final}}^{(k)}$. On peut s'en servir comme **initialisation** d'un nouveau run $k+1$, après y avoir injecté un **bruit** modéré (un “shake” aléatoire). C'est un schéma de *random restart* où l'on repart d'une solution existante en la “déstabilisant” légèrement, cherchant un nouveau bassin d'attraction à proximité.

C. Analyse de Convergence et Sélection Finale

Chaque **run** se prolonge jusqu'à la **stabilisation** (ou un nombre d'itérations T_k imposé). On obtient ainsi un ensemble de **configurations finales** $\{\Omega_{\text{final}}^{(1)}, \dots, \Omega_{\text{final}}^{(R)}\}$. On **compare** alors leurs valeurs de la fonction $\mathcal{J}(\Omega)$:

$$\Omega_{\text{best}} = \arg \min_{k \in \{1, \dots, R\}} \mathcal{J}(\Omega_{\text{final}}^{(k)}),$$

ou toute autre mesure de qualité (par exemple la **cohésion** en clusters). Cette **comparaison** permet de **choisir** la configuration la plus avantageuse, c'est-à-dire celle offrant le minimum d'énergie ou le maximum de modularité (voir § 2.2.3, § 7.2.1 pour les notions associées).

Dans certains cas, les **runs** se révèlent converger à des topologies très similaires. Dans d'autres cas, on obtient des configurations sensiblement différentes (clusters structurés différemment), signe de la présence de plusieurs minima locaux stables.

D. Coût Computationnel et Bénéfices

Lancer R exécutions du DSL multiplie évidemment par R la **charge de calcul** (sauf si l'on dispose d'une **parallélisation**). Il s'agit donc d'un **compromis** : plus on tente de runs, plus on augmente la probabilité de **découvrir** un minimum local plus profond ou au moins distinct des précédents. Dans des problèmes hautement non convexes, cette approche multi-run est un **classique** de l'optimisation, simple à mettre en œuvre et relativement efficace pour obtenir une "stratégie de globalité" sans recourir à des mécanismes internes plus complexes (par ex. recuit simulé § 7.3, algorithmes génétiques § 7.5.2).

En outre, on peut marier le **multi-run** avec diverses **techniques** d'exploration : injecter un bruit plus fort en phase initiale, ajouter un "shake" stochastique de temps en temps (cf. § 7.2.2.3). Chaque run peut bénéficier d'**aménagements** particuliers, assurant une diversité accrue des chemins de descente.

E. Pistes de Variation et Schémas Avancés

L'approche du **planning adaptatif** permet d'associer un **recuit** (cf. § 7.3) à chaque run et d'ajuster le **profil de température** pour les runs suivants, notamment si plusieurs runs se retrouvent coincés dans un même attracteur.

Dans un **schéma d'apprentissage continu**, il est possible de **réutiliser** la meilleure solution Ω_{best} obtenue lors d'une session précédente comme point de départ pour la suivante, tout en lançant simultanément quelques runs avec des **initialisations totalement aléatoires** afin de conserver une diversité dans l'exploration. —

*Une autre approche consiste en la **fusion de solutions**, où plusieurs matrices finales $\boldsymbol{\Omega}^{(k)}$ peuvent être combinées, par **assemblage ou mélange** (voir § 7.5.3.2), afin d'obtenir un **consensus** ou une **moyenne optimisée**, si le problème s'y prête.*

Conclusion

La **recherche multi-run** (ou *multi-démarrage*) propose d'**exécuter** la **dynamique** DSL plusieurs fois, avec des **initialisations** distinctes. Chaque exécution opère comme une **descendance locale** menant à un attracteur éventuel. Au final, on **compare** les solutions obtenues et on retient la **meilleure** (ou on les combine, cf. § 7.5.3.2).

En **optimisation non convexe**, ce principe de *rand restart* est un **moyen simple** d'éviter le piégeage dans un seul minimum local. Appliqué au DSL, il donne une méthode "externe" (hors dynamique interne) pour accroître la **qualité** et la **robustesse** des solutions, tout en restant aisément **parallelisable** et combinable avec d'autres stratégies d'exploration (recuit, algorithmes génétiques, etc.).

7.5.3.2. Fusionner les solutions ou choisir la meilleure par un critère global

Lorsque l'on exécute la **dynamique DSL** (ou tout autre algorithme d'optimisation) en mode **multi-run** — c'est-à-dire en lançant plusieurs "essais" indépendants, avec des initialisations distinctes ou des paramètres légèrement différents — on obtient un **ensemble** de configurations finales $\{\Omega^{(r)}\}_{r=1}^R$. Chaque exécution r produit donc un **SCN** (Synergistic Connection Network) stabilisé, potentiellement un **minimum local** dans l'espace $\{\omega_{i,j}\}$. La question se pose alors : **comment** exploiter ces R solutions ?

En pratique, deux stratégies peuvent être mises en œuvre. La première consiste à **choisir la meilleure solution** selon un **critère global**, tel que l'énergie $J(\Omega)$, la modularité ou encore la cohésion du réseau. La seconde approche repose

sur la **fusion de plusieurs solutions**, permettant de construire un **réseau combiné** ou un **consensus**, intégrant les points forts de chaque exécution afin d'optimiser la structure finale.

A. Choisir la Meilleure Solution par un Critère Global

L'évaluation repose sur la définition d'un **critère** permettant de comparer les différentes solutions obtenues. L'un des plus courants consiste à mesurer l'**énergie** $J(\Omega^{(r)})$ (cf. chap. 7.2.1) ou une **mesure de qualité**, telle que la modularité ou la densité intra-cluster. La solution retenue, notée $\Omega^{(r^*)}$, est celle qui **minimise** J ou **maximise** la modularité. Si l'on définit un **score** $Q(\Omega)$, comme un indicateur de qualité du clustering, la sélection s'effectue selon :

$$\Omega^{(r^*)} \text{ tel que } r^* = \arg \max_{r \in \{1, \dots, R\}} Q(\Omega^{(r)}).$$

ou selon argmin si l'objectif est de minimiser un coût.

L'un des **avantages** majeurs de cette approche est sa **simplicité**, puisqu'elle permet d'obtenir une solution $\Omega^{(r^*)}$ directement exploitable pour la suite du traitement, en conservant les clusters et les liaisons les plus robustes. Elle apporte également une **clarté structurelle**, en évitant les mélanges d'informations ambiguës et en garantissant la conservation d'une seule structure **cohérente** des poids $\omega_{i,j}$.

Cependant, certaines **limites** doivent être prises en compte. Il arrive que plusieurs solutions $\Omega^{(r)}$ soient **quasi équivalentes** en termes de coût ou très proches en qualité, ce qui peut signifier que plusieurs partitions valables existent. En ne conservant qu'une seule solution, il y a un risque de **perte d'information**, notamment si plusieurs minima locaux de qualité similaire existent. Dans ce cas, rejeter certaines solutions peut priver le modèle d'**alternatives pertinentes** et de perspectives de segmentation différentes du réseau.

B. Fusionner les Solutions : Approche "Consensus" ou "Ensemble"

Chaque exécution r produit un **réseau** $\Omega^{(r)}$. Si ces réseaux présentent des **similarités structurelles** (par exemple, la plupart possèdent un même cluster de base) mais aussi certaines **disparités locales** liées aux minima atteints, il peut être pertinent de **fusionner** ces informations pour créer un **réseau consensuel** plus robuste. Cette approche est similaire aux **méthodes d'ensemble** utilisées en apprentissage machine, telles que le bagging ou le voting, qui visent à améliorer la stabilité et à limiter le sur-apprentissage.

La fusion peut s'opérer en **moyennant** ou en prenant la **médiane** des pondérations $\omega_{i,j}$ sur l'ensemble des exécutions :

$$\omega_{i,j}^{(\text{consensus})} = \frac{1}{R} \sum_{r=1}^R \omega_{i,j}^{(r)} \quad \text{ou} \quad \text{med}\{\omega_{i,j}^{(1)}, \dots, \omega_{i,j}^{(R)}\}.$$

Ainsi, le **SCN final** reflète la **cohérence des solutions** obtenues. Une autre approche consiste à fixer un **seuil** : une liaison (i, j) est conservée si elle apparaît dans plus de **50 % des runs** avec une valeur $\omega_{i,j} > \theta$. Cette méthode produit un réseau plus **discret**, mettant en évidence uniquement les connexions les plus consensuelles.

L'un des principaux **avantages** de cette fusion est sa **robustesse**, en lissant les divergences purement locales et en réduisant les effets des perturbations stochastiques. Elle offre également une **vision plus globale** de l'espace des solutions : les liens **fortement présents** dans plusieurs runs sont renforcés, tandis que les liaisons trop incertaines sont atténuerées ou supprimées.

Toutefois, cette approche comporte certaines **limites**. Elle peut **diluer la netteté** du réseau, en produisant des liens modérés plutôt que des connexions clairement marquées ou nulles. La **complexité computationnelle** peut aussi devenir un problème si le nombre d'exécutions est très élevé, nécessitant un traitement sur un ensemble de $O(n^2)$ **liens**, ce qui peut s'avérer coûteux en ressources.

C. Choix d'un Critère Global ou d'une Fusion

Dans la **pratique**, le DSL peut adopter l'une ou l'autre stratégie en fonction du **contexte**.

Si l'objectif est d'obtenir un **SCN unique et opérationnel**, comme dans des calculs en temps réel ou des applications robotiques, il est préférable de **choisir** la solution $\Omega^{(r^*)}$ qui minimise \mathcal{J} ou maximise la modularité. Cette approche garantit un **déploiement simple et direct**, optimisé pour l'exploitation immédiate du réseau.

En revanche, si l'enjeu est d'assurer une **robustesse accrue** ou de générer un **diagnostic plus riche**, une **approche par fusion** est plus adaptée. Ce procédé permet d'extraire un **réseau moyen ou majoritaire**, capable de mieux résister aux variations dues aux minima locaux, tout en mettant en évidence les connexions **consensuelles vs incertaines**.

Quel que soit le choix adopté, il est nécessaire de définir un **critère d'évaluation**. Dans le premier cas, un **score** $\text{score}(\Omega)$ basé sur l'énergie, la modularité ou le ratio interne/externe est utilisé pour **sélectionner** la meilleure solution. Dans le second cas, un opérateur Consensus (moyenne, médiane, vote) permet de réaliser une **fusion cohérente** des différentes solutions obtenues.

Conclusion

Dans une procédure **multi-run** ou **multi-débuts** du DSL (où l'on lance plusieurs exécutions indépendantes, chacune pouvant converger vers un attracteur différent), il est crucial de **traiter** ces solutions multiples :

- **Choisir la meilleure** par un **critère** global (ex. $\min \mathcal{J}$ ou maxmodularité).
- **Fusionner** tout ou partie des solutions via un **réseau** consensuel (moyenne, médiane, vote).
- **Choisir la meilleure** offre un **résultat unique** et lisible, d'application directe.
- **Fusionner** exploite la **diversité** des minima locaux, pour un SCN potentiellement plus **robuste**, reflétant la variété des chemins d'optimisation parcourus.

Ainsi, selon la finalité (besoin d'un unique "cluster final" vs. besoin de voir la distribution des solutions), on privilégiera l'une ou l'autre méthode. Dans tous les cas, la démarche multi-run élargit la **recherche** de solutions par rapport à un run unique, ce qui peut **améliorer** la performance globale du DSL et **réduire** le risque de se bloquer dans un attracteur isolé.

7.5.3.3. Pistes pour se rapprocher d'un minimum plus global

Dans la mesure où les **heuristiques** (génétiques, colonies de fourmis, etc.) et les **approches multi-run** (7.5.3.2) visent déjà à élargir la recherche dans l'espace des liaisons $\{\omega_{i,j}\}$, on peut se demander quelles **stratégies** supplémentaires permettent d'**approcher** un minimum plus global, c'est-à-dire de surmonter la simple convergence locale de la descente DSL. Les pistes ci-dessous complètent (ou prolongent) les méthodes hybrides pour **maximiser** les chances de trouver une configuration Ω^* de plus faible énergie \mathcal{J} .

A. Combinaison des Heuristiques et du Recuit Simulé

L'association du **recuit simulé** et de l'**approche génétique** permet d'intégrer une phase d'exploration locale à l'intérieur de chaque individu d'une population évolutive. Concrètement, pour un individu donné, représenté par une matrice $\Omega^{(k)}$, un **mini-cycle de recuit** est appliqué afin d'injecter du bruit stochastique et de modifier **localement** la structure du réseau.

Le processus suit plusieurs étapes. D'abord, un individu $\Omega^{(k)}$ est généré. Ensuite, un recuit local est effectué pendant L itérations selon la mise à jour :

$$\omega_{i,j} \leftarrow \omega_{i,j} + \Delta_{\text{DSL}}(i,j) + \sigma(t) \xi_{i,j}(t),$$

où $\Delta_{\text{DSL}}(i,j)$ représente la mise à jour du SCN et $\xi_{i,j}(t)$ est un bruit dont l'amplitude $\sigma(t)$ décroît avec le temps. Une fois cette phase terminée, l'individu est évalué en fonction de son énergie $\mathcal{J}(\Omega^{(k)})$, puis passe à l'étape de sélection génétique impliquant **crossover et mutation globale**. Cette approche combine ainsi une **perturbation stochastique fine** par recuit et une **exploration plus large** via les algorithmes génétiques.

Pour éviter une **explosion des coûts computationnels**, la gestion du **planning de température** peut être optimisée en limitant le nombre d'itérations de recuit à l'intérieur de chaque individu. Une stratégie consiste à **n'appliquer un**

recuit court (10–20 itérations) qu’aux individus survivants après chaque sélection. Cela permet de ne pas s’appuyer uniquement sur la phase de mutation et crossover, mais d’offrir à chaque individu la possibilité d'**affiner sa descente locale**, tout en laissant le processus génétique s’occuper du franchissement des barrières énergétiques globales.

B. Hybridation Inhibition / Heuristiques Multi-run

L'**inhibition avancée** peut être utilisée pour filtrer efficacement les solutions dans un cadre heuristique global. Un des défis majeurs des algorithmes évolutionnaires est la **multiplication excessive des états candidats**, notamment lorsque de nombreux “enfants” sont générés. Afin de **réduire cet espace de recherche**, une forme d'**inhibition ou de parcimonie** peut être imposée avant l’évaluation de la fonction \mathcal{J} . Mathématiquement, cela revient à **normaliser ou contraindre** la matrice ω après une mutation, par exemple en limitant la somme maximale des pondérations sortantes par entité, selon la contrainte $\sum_j \omega_{i,j} \leq \kappa$. Ce filtrage permet de focaliser l’optimisation sur des solutions plus **parcimonieuses et pertinentes**.

L’approche **multi-run avec perturbation** offre un autre levier d’optimisation. Comme mentionné en (7.5.3.2), plusieurs exécutions de la dynamique DSL peuvent être lancées depuis différentes **conditions initiales**. Pour renforcer cette exploration, une **procédure de réajustement** peut être appliquée après chaque run local :

Analyser la structure Ω atteinte.

Appliquer une **inhibition drastique**, en coupant les liens les plus faibles sous un seuil adaptatif.

Réintroduire un **bruit stochastique modéré**.

Relancer la dynamique DSL pour quelques itérations.

Cette stratégie, basée sur le principe "**shake + inhibition + re-run**", vise à **déstabiliser les attracteurs locaux** et à favoriser la recherche d’une configuration plus optimale en maintenant une **diversité contrôlée** au sein des solutions générées.

C. Mécanismes de Sélection / Contrôle

L'**acceptation de solutions moins bonnes** peut être nécessaire dans un **paysage énergétique non convexe**, où atteindre un minimum global implique parfois une **augmentation transitoire de \mathcal{J}** . Cette démarche repose sur des critères tels que la **règle de Metropolis** (recuit simulé) ou un **ratio d’acceptation** dans un algorithme évolutionnaire, permettant de ne pas systématiquement éliminer un individu moins performant si cela favorise l’évasion d’un puits local. Mathématiquement, cette transition est régulée par un facteur $\Delta\mathcal{J}$, avec une probabilité d’acceptation définie par :

$$\exp(-\Delta\mathcal{J}/T)$$

lorsque $\Delta\mathcal{J} > 0$, introduisant ainsi une **composante exploratoire** contrôlée.

En parallèle, un "**shake**" **périodique** peut être instauré indépendamment du recuit, appliqué toutes les K itérations DSL. Cette perturbation est définie par :

$$\omega_{i,j} \leftarrow \omega_{i,j} + \epsilon_{i,j},$$

où $\epsilon_{i,j}$ est un **petit bruit symétrique** introduisant une variation contrôlée. L’objectif est de **perturber légèrement la structure** pour permettre au SCN de s’extraire d’un attracteur local, tout en s’assurant que la perturbation soit **assez faible** pour ne pas altérer la cohérence globale du réseau, mais **suffisamment forte** pour briser les barrières de minima trop étroits.

D. Synthèse : Vers un Minimum Plus Global

L’ensemble de ces **pistes** — recuit intégré aux heuristiques, inhibition sélective, multi-run, acceptation probabiliste de solutions moins bonnes, “shake” périodique — forment un **cercle** de méthodes visant à **libérer** la dynamique DSL de ses limitations purement locales. Les **principes clés** sont :

Maintenir la descente locale comme base (car elle stabilise et oriente la formation de clusters),

Autoriser des perturbations stochastiques ou globales capables de franchir les barrières d'énergie,

Combiner éventuellement l'inhibition avancée et la parcimonie pour limiter l'explosion combinatoire,

Multiplier les chemins d'exploration (ex. runs indépendants), puis fusionner ou comparer les solutions atteintes.

D'un point de vue **mathématique**, on peut voir ces méthodes comme des **routines** d'optimisation "globales" appliquées à la pseudo-fonction \mathcal{J} du SCN, évinçant la contrainte de la simple "descente déterministe". Elles ne **garantissent pas** la découverte de l'optimum global, mais **améliorent** nettement la probabilité d'échapper aux minima locaux, et donc de **rapprocher** le système d'un arrangement Ω^* plus satisfaisant dans la pratique.

7.6. Adaptation Incrémentale et Apprentissage Continu

Un **SCN** (Synergistic Connection Network) peut fort bien fonctionner dans un mode **batch**, où l'on dispose d'un jeu d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ final, et où l'on exécute la dynamique DSL (mise à jour $\omega_{i,j}(t)$) jusqu'à convergence. Toutefois, nombre d'applications exigent que le **réseau** continue de s'**adapter** dès lors que de **nouvelles entités** apparaissent ou que d'**anciennes** entités disparaissent. C'est tout l'enjeu de l'**apprentissage continu** (ou **incrémental**), parfois en **flux** (streaming), assurant qu'un **SCN** demeure **réactif** à l'évolution du contexte (capteurs, utilisateurs, données...).

7.6.1. Mise à Jour en Ligne

Lorsque les entités $\{\mathcal{E}_i\}$ arrivent en **continu**, on ne peut pas se contenter de recalculer totalement la **matrice** $\{\omega_{i,j}\}$ à chaque insertion, un tel $O(n^2)$ reprocessing, répété à chaque nouvelle entité, deviendrait prohibitif. On privilégie des mécanismes **en ligne** (online) où l'on met à jour *localement* la structure du SCN.

7.6.1.1. Scénario où des Entités \mathcal{E}_i ou Flux Sensoriels Arrivent en Continu

Dans le **Deep Synergy Learning** (DSL), un **Synergistic Connection Network** (SCN) peut évoluer au fil du temps lorsque de **nouvelles** entités (ou de nouveaux flux sensoriels) se présentent. Le but est de **mettre à jour** la structure existante sans avoir à relancer un **traitement complet** de toutes les pondérations $\omega_{i,j}$. Cette approche **en continu** (ou *online*) se révèle cruciale dans des contextes d'apprentissage incrémental, de systèmes de recommandation temps réel, ou de traitement de flux de données où des **objets** inédits font leur apparition à chaque instant.

A. Arrivée d'une Nouvelle Entité \mathcal{E}_{n+1}

On suppose qu'au temps t , un **nouvel objet** \mathcal{E}_{n+1} (par exemple un nouveau document, utilisateur, ou capteur) doit être **intégré** au SCN. Mathématiquement, si le réseau contenait déjà n entités, la **matrice** ω de dimension $n \times n$ se voit étendue à $(n + 1) \times (n + 1)$. On introduit donc de **nouvelles** liaisons $\omega_{(n+1),j}$ et $\omega_{j,(n+1)}$ (selon que le SCN est orienté ou non), initialisées à zéro ou à une petite valeur ω_{init} .

On souhaite alors **connecter** \mathcal{E}_{n+1} à ses pairs $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ d'une façon cohérente avec la logique du DSL, c'est-à-dire tenir compte de la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$. Dans un mode strictement *batch*, il faudrait réévaluer **toutes** les paires (i, j) . Ici, l'idée est de procéder de façon *partielle* ou *locale*, évitant une refonte complète du réseau.

B. Mise à Jour DSL Partielle

Après avoir inséré la ligne et la colonne correspondant à \mathcal{E}_{n+1} , on applique la **règle** DSL à un *voisinage* limité. Formulée pour chaque nouvel indice $(n + 1), j$, la mise à jour typique s'écrit :

$$\omega_{(n+1),j}(t + 1) = \omega_{(n+1),j}(t) + \eta [S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau \omega_{(n+1),j}(t)].$$

Dans une **approche** vraiment *en ligne*, on ne veut pas calculer la synergie pour tous les $j = 1, \dots, n$. On se limite souvent à un sous-ensemble $N((n + 1), k)$, par exemple les k plus proches entités selon des heuristiques (distance, indice de similarité rapide, etc.). Cela réduit le coût de traitement, tout en permettant à la nouvelle entité d'établir des liens forts avec quelques "voisins" déjà présents.

Ce type d'insertion incrémentale se reproduit chaque fois que **des données** inédite(s) parviennent : un nouvel **utilisateur**, un **segment** de données sensoriel, un **flux** ou un **document**. Le **SCN** évolue en temps réel :

While new data arrives: AddEntity(\mathcal{E}_{n+1}), PartialUpdate($\{\omega_{(n+1),j}\}$),

et éventuellement **réajuste** l'inhibition ou les coupes de liens pour maintenir la **parsimonie** (voir § 7.4.3). Le **DSL** se conçoit alors comme un **processus** permanent, modifiant localement la structure pour intégrer le nouvel objet.

C. Illustrations Mathématiques

On peut imaginer la **matrice étendue** :

$$\omega_{\text{extended}}(t) = \begin{pmatrix} \omega_{1,1}(t) & \cdots & \omega_{1,n}(t) & 0 \\ \vdots & \ddots & \vdots & 0 \\ \omega_{n,1}(t) & \cdots & \omega_{n,n}(t) & 0 \\ 0 & \cdots & 0 & 0 \end{pmatrix}$$

puis, pour quelques pas d'itération, on met à jour $\omega_{(n+1),j}$ et $\omega_{j,(n+1)}$ (selon que le SCN est orienté ou non). Si $N((n+1),k)$ désigne le **voisinage** de \mathcal{E}_{n+1} , il se peut que $\omega_{(n+1),j}$ demeure à 0 pour tout $j \notin N((n+1),k)$. Après un nombre T_{local} d'itérations, \mathcal{E}_{n+1} est considérée comme **intégrée** au réseau, bien que la dynamique puisse continuer à évoluer.

D. Conséquences sur l'Auto-Organisation

L'arrivée d'une entité supplémentaire modifie potentiellement la **composition** des clusters. En effet, si \mathcal{E}_{n+1} présente une synergie élevée avec certaines entités déjà groupées, elle peut **restructurer** un cluster ou en rejoindre un existant. Cette situation est particulièrement fréquente dans des applications de **classification en ligne** ou de **recommandation** : un nouvel utilisateur se connecte avec quelques **items** déjà présents, entraînant une reconfiguration partielle des **liens**.

Cela a aussi l'avantage de maintenir un **SCN “ouvert”** : plutôt que de figer la structure en fin de calcul, on tolère que le réseau continue de s'auto-organiser au fur et à mesure des **incrément**s de données.

E. Avantage Comparé à un Batch Complet

L'**approche en flux** présente plusieurs atouts. Elle évite une **réinitialisation complète** ou un recalculation intégral à chaque nouvelle entité, garantissant ainsi une **scalabilité accrue**, puisqu'elle ne nécessite que le traitement de $O(k)\backslash\text{mathrm}{O}(k)$ liaisons pour l'entité ajoutée ou un sous-ensemble donné. Cette méthode offre également une **réactivité immédiate**, car les entités sont directement intégrées au réseau sans attendre une mise à jour globale.

Toutefois, cette mise à jour **partielle** comporte certains **inconvénients**. Elle peut **manquer certaines synergies à longue portée**, si aucun mécanisme de propagation étendue n'est prévu. De plus, une fréquence d'arrivée élevée des entités peut entraîner **une accumulation d'erreurs**, notamment si aucun raffinement global n'est appliqué pour ajuster les pondérations et corriger les déséquilibres progressifs du réseau.

Conclusion

Dans un **DSL** en environnement dynamique, l'arrivée **continue** d'entités \mathcal{E}_i (ou de flux sensoriels) se gère par une **intégration incrémentale** : on étend la matrice ω , on calcule localement la **synergie** entre le nouvel objet et un sous-ensemble de voisins, puis on procède à une **mise à jour** partielle du SCN pour “brancher” cette entité au réseau existant. Ce **scénario** se généralise aux systèmes “ouvert”, où le **Deep Synergy Learning** fonctionne en **ligne** (à chaque nouvelle donnée), assurant une **évolution** permanente de la structure des clusters et autorisant une **scalabilité** satisfaisante sans recomputation globale. Des détails complémentaires, notamment sur la **réactualisation** des similarités ou sur la **propagation** des changements, seront discutés en § 7.6.1.2, § 7.6.1.3 et § 7.6.2.

7.6.1.2. Recalcule Partiel de la Synergie Uniquement dans le Voisinage de la Nouvelle Entité

Lorsqu'un **nouvel objet** \mathcal{E}_{n+1} est inséré dans un **Synergistic Connection Network** (SCN) en mode **incrémental**, il est souvent inenvisageable de recalculer les **valeurs de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pour **toutes** les paires (i, j) . En pratique, on préfère se limiter à un **voisinage** réduit ou à un **échantillon** pertinent autour de \mathcal{E}_{n+1} . Cette stratégie de *recalcule partiel* permet un **gain** substantiel en temps de calcul et en mise à jour, tout en maintenant l'essentiel de la cohérence **auto-organisée**.

A. Principe de la Mise à Jour Locale

Lorsqu'une entité inédite \mathcal{E}_{n+1} surgit, on étend la **matrice** ω de taille $n \times n$ à $(n+1) \times (n+1)$. Au lieu d'évaluer **toutes** les liaisons $\omega_{(n+1),j}$ pour $j = 1, \dots, n$, on se concentre sur un sous-ensemble $\mathcal{V}(\mathcal{E}_{n+1}) \subseteq \{1, \dots, n\}$ correspondant aux **voisins** probables de \mathcal{E}_{n+1} . D'un point de vue **mathématique**, on restreint :

$$\omega_{(n+1),j}(t+1) = \omega_{(n+1),j}(t) + \eta[S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau \omega_{(n+1),j}(t)] \quad \text{uniquement pour } j \in \mathcal{V}(\mathcal{E}_{n+1}).$$

Pour tout $j \notin \mathcal{V}$, on fixe $\omega_{(n+1),j}(t+1) = 0$ ou on laisse la valeur d'initialisation (ex. nulle) sans calcul de synergie. Cela garantit que l'**effort** de mise à jour demeure $O(|\mathcal{V}|)$ au lieu de $O(n)$.

B. Choix du Voisinage

Pour déterminer $\mathcal{V}(\mathcal{E}_{n+1})$, on peut recourir à différentes heuristiques :

k-NN : on sélectionne les k entités \mathcal{E}_j déjà présentes ayant la plus haute similarité ou la plus faible distance par rapport à \mathcal{E}_{n+1} . Cela suppose de pouvoir approximer rapidement la distance $\text{dist}(\mathcal{E}_{n+1}, \mathcal{E}_j)$ (ou la synergie S).

ϵ -radius : on prend toutes les entités à moins d'un rayon ϵ dans un espace de représentation, c'est-à-dire

$$\mathcal{V}(\mathcal{E}_{n+1}) = \{j \mid \text{dist}(\mathcal{E}_{n+1}, \mathcal{E}_j) < \epsilon\}.$$

Si la distribution est homogène, le nombre d'entités à l'intérieur de ce rayon sera limité.

Échantillonnage : parfois, on se contente de tirer au hasard un petit nombre d'entités existantes, à des fins de **test**. Ceci préserve la diversité, au prix d'un contrôle moindre.

Au-delà, on peut envisager des structures d'indexation (k-d trees, ball trees) permettant de retrouver rapidement le *voisinage géométrique* dans un espace latent, réduisant le **coût** $O(n)$ d'une recherche naïve.

C. Mise à Jour DSL et Coût Réduit

En limitant la **dynamique** DSL aux seules liaisons $\omega_{(n+1),j}$ pour $j \in \mathcal{V}$, on évite un **recalcul** global d'ordre $O(n^2)$. Cela conserve le **principe** de mise à jour :

$$\omega_{(n+1),j}(t+1) = \omega_{(n+1),j}(t) + \eta[S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau \omega_{(n+1),j}(t)].$$

Au besoin, on peut exécuter **plusieurs** pas "locaux" de la règle DSL pour stabiliser $\omega_{(n+1),j}$ sur un point d'équilibre, sans toucher aux autres $\omega_{i,j}$. Cela constitue un mini-cycle d'auto-organisation pour l'entité \mathcal{E}_{n+1} et son entourage immédiat.

D. Cohérence Globale et Précision

La **sparsification** induite par ce recalcul partiel (et la fixation à zéro des liaisons non visitées) s'appuie sur l'hypothèse qu'il n'est **pas** crucial de lier \mathcal{E}_{n+1} à des entités lointaines ou peu synergiques. S'il existe un **cluster** approprié pour accueillir \mathcal{E}_{n+1} , ce *voisinage* restreint suffira à guider l'intégration, car les entités de ce cluster se trouveront dans \mathcal{V} .

En revanche, un risque de **fausse exclusion** subsiste : si \mathcal{E}_{n+1} aurait pu entretenir une synergie notable avec un noeud \mathcal{E}_j qu'on n'a pas jugé "proche" au départ, on perd ce lien potentiel en ignorant la comparaison. Ce **compromis** relève des techniques **approx** de k-NN, assurant une **balance** entre exhaustivité et coût de calcul.

E. Avantages et Limites

Avantages :

Scalabilité : le réseau peut croître progressivement (arrivée de multiples entités) sans recalculer la **synergie** pour toutes les paires.

Localité : on assure une intégration "minimale" de \mathcal{E}_{n+1} sans perturber le reste du SCN.

Rapidité : on contourne le goulot $O(n^2)$, particulièrement utile pour des flux à cadence élevée.

Limites :

Qualité : on sacrifie parfois la découverte de liens synergiques “lointains” mais importants.

Cohérence : si trop d’entités sont insérées en ignorant le recalcul global, on pourrait dégrader la structure à long terme, à moins de prévoir des “phases de consolidation” (voir § 7.6.2).

Conclusion

Le **recalcule partiel** de la synergie $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ se cantonne à un **voisinage** restreint de \mathcal{V} afin de réduire drastiquement la complexité de mise à jour lors de l’insertion d’un objet **inédit** dans le SCN. Cette approche favorise un **mode continu** ou *en ligne*, où l’on traite chaque nouvel arrivant au fil de l’eau :

Détermination d’un ensemble \mathcal{V} (k-NN, ϵ -radius, échantillon...) ;

Calcul ou **mise à jour** des liaisons $\omega_{(n+1),j}$ pour $j \in \mathcal{V}$ via la règle DSL ;

Conservation d’un graphe plus large mais de façon incrémentale.

Cette **flexibilité** s’avère cruciale pour des situations d’apprentissage **ininterrompu**, de systèmes de recommandation en temps réel, ou de tout domaine nécessitant l’**adaptation** au flux constant de nouveaux éléments. Au prix d’une **légère** approximation, on garantit la **scalabilité** et la **réactivité** du SCN, tout en conservant l’**esprit** auto-organisateur du DSL.

7.6.1.3. Maintien d’un SCN toujours “ouvert”

De nombreux **systèmes** appliquant le **Deep Synergy Learning** (DSL) doivent gérer une **évolution** perpétuelle : de nouvelles entités se présentent, d’autres disparaissent ou voient leur pertinence se modifier radicalement. Dans ce contexte, le **Synergistic Connection Network** (SCN) ne peut plus être considéré comme un **ensemble** statique : il devient essentiel de maintenir un SCN “**ouvert**”, c’est-à-dire constamment actif, continuellement prêt à **intégrer** de nouveaux noeuds ou à **désactiver** ceux devenus obsolètes. Cette section décrit le principe d’un tel SCN permanent et les considérations qui l’accompagnent.

A. Principe d’un SCN Permanent

Dans un SCN *classique*, l’apprentissage s’effectue sur un **ensemble** $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ fixe : la **dynamique** DSL permet d’**auto-organiser** les liaisons $\omega_{i,j}$ jusqu’à convergence (ou quasi-stabilisation). Toutefois, dans de nombreuses applications (systèmes de recommandation, flux de données, robotique, etc.), il est irréaliste de supposer que la liste des entités reste inchangée.

Lorsqu’une **nouvelle entité** \mathcal{E}_{new} apparaît, on étend la **matrice** ω en lui allouant une **ligne** et une **colonne** supplémentaires (ou seulement une demi-matrice si l’on raisonne sur un SCN symétrique, voir § 7.6.1.1). Les nouvelles pondérations $\omega_{(\text{new}),j}$ et $\omega_{j,(\text{new})}$ sont initialisées à des valeurs faibles (souvent zéro). On applique alors la **règle** DSL de façon locale ou partielle (cf. § 7.6.1.2) :

$$\omega_{(\text{new}),j}(t+1) = \omega_{(\text{new}),j}(t) + \eta [S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) - \tau \omega_{(\text{new}),j}(t)],$$

pour j dans un sous-ensemble restreint (voisinage ou échantillon). Sur le **plan mathématique**, la **somme** $\sum_j \omega_{(\text{new}),j}(t)$ s’ajuste progressivement selon la **synergie** avec les entités existantes, donnant à \mathcal{E}_{new} la possibilité de **trouver** sa place dans la structure de clusters.

Le **SCN** demeure donc **ouvert** : on ne clôt pas l’apprentissage après un batch unique, on laisse la dynamique se poursuivre de manière **infinie** ou prolongée, accueillant successivement $\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \dots$ au fil des arrivées.

À l'inverse, certaines entités **deviennent** inactives ou obsolètes (par exemple, un capteur retiré, un utilisateur inactif, un concept qui n'est plus pertinent). Un SCN réellement *ouvert* doit pouvoir **diluer** ou **couper** leurs liaisons. S'il apparaît que la synergie $\{S(\mathcal{E}_{\text{old}}, \mathcal{E}_j)\}$ n'est plus jamais significative, ou que cette entité n'a pas été sollicitée depuis longtemps, on peut progressivement **ramener** $\omega_{(\text{old}),j}$ à zéro, jusqu'à éliminer complètement \mathcal{E}_{old} .

Mathématiquement, on introduit une **règle** de désactivation : si un nœud \mathcal{E}_{old} reste sous un certain **seuil** d'activité ou de synergie moyenne, on coupe ses liaisons et on l'exclut du SCN. Ce mécanisme maintient l'espace $\{\omega_{i,j}\}$ à **taille raisonnable**, évitant un gonflement indéfini du réseau.

B. Stabilité et Complexité

Le **SCN** permanent ne vise plus une **convergence** unique et définitive, puisqu'il évolue continuellement. On parle alors d'un **processus non stationnaire**, où la **dynamique** DSL s'adapte en **temps réel**.

Dans la mesure où de **nouvelles** entités peuvent arriver fréquemment, la **dimension** du SCN (ex. nombre de nœuds n) croît sans cesse. Le calcul des synergies, s'il est complet, devient $O(n^2)$. À long terme, cette complexité est prohibitive (voir § 7.2.3). D'où la nécessité de recourir à :

Techniques de recalcul partiel (k-NN local, cf. § 7.6.1.2),

Inhibition et **parsimonie** pour forcer une structure éparsée (chap. 7.4, 7.5.1),

Retrait régulier des entités inactives ou peu synergiques,

Structures d'indexation pour retrouver les "voisins" d'une nouvelle entité sans scan exhaustif.

Même dans un **flux** continu, on peut définir des **périodes** durant lesquelles la dynamique DSL se focalise sur la stabilisation des liaisons existantes. Lorsque de nouvelles entités arrivent, on les intègre partiellement (quelques itérations de mise à jour locale). Le réseau global reste, la plupart du temps, **assez stable** tout en demeurant **ouvert** à l'incorporation d'éléments inédits.

Sur un plan mathématique, on peut définir un **critère** du type $\|\Delta\omega\| \approx 0$ ou la valeur $J(\omega)$ devient quasi stable. Dans un SCN "ouvert", ce **statu quo** n'est jamais total (car de nouvelles entités peuvent bouleverser localement un cluster), mais on espère atteindre un **régime** quasi stationnaire : chaque insertion ne perturbe que localement le réseau, et l'ensemble continue d'évoluer en **douce** adaptation, sans s'effondrer.

C. Lien avec les Autres Sections

Un SCN **ouvert** fait écho à plusieurs **stratégies** vues précédemment :

- **Recuit simulé en continu** : on peut injecter du bruit (température non nulle) pour autoriser des "réarrangements" quand trop d'entités nouvelles s'ajoutent (cf. chap. 7.3).
- **Inhibition** auto-adaptative ou règles k-NN** : on limite le degré sortant de chaque nœud (cf. chap. 7.4, 7.5.1) pour maîtriser la croissance du réseau.
- **Mise à jour locale** : on manipule seulement un sous-ensemble de liaisons $(n+1, j)$ (voir § 7.6.1.2), sans recalculer ω de manière globale.
- **Super-nœuds** et **couches** (chap. 6) : l'arrivée d'entités peut également être gérée à des niveaux hiérarchiques.

Conclusion

Le **maintien** d'un SCN toujours "ouvert" vise à répondre au défi d'une **évolution** perpétuelle du **système**, là où de nouvelles entités (capteurs, documents, utilisateurs) apparaissent sans cesse et où certains **anciens** disparaissent ou se réactivent. Sur le **plan mathématique**, cela signifie :

Insertion incrémentale de \mathcal{E}_{new} via la création de nouvelles liaisons ω ,

Mise à jour partielle ou locale (k-NN, inhibition) pour empêcher le coût $O(n^2)$ et maintenir la lisibilité,

Eventuelle dilution ou retrait des entités devenues inactives,

Persistante d'une dynamique DSL, sans convergence *ultime*, mais une adaptation asymptotique.

Cette **logique** s'accorde parfaitement aux environnements temps réel et *streaming*, où l'on requiert un **réseau** auto-organisé capable de **s'actualiser** en permanence, garantissant une **structure** toujours appropriée face aux changements.

7.6.2.1. addEntity(E) : Initialisation $\omega_{E,j}$ sur un Échantillon ? sur k Plus Proches Voisins ?

Dans le cadre d'un **Synergistic Connection Network** (SCN) fonctionnant en **apprentissage continu**, l'ajout d'une **nouvelle entité** \mathcal{E}_{new} doit se faire de manière **incrémentale** et **sélective**, sans déclencher un recalcul global de toutes les pondérations $\omega_{i,j}$. Le but est de raccorder \mathcal{E}_{new} de façon cohérente au réseau préexistant, tout en maîtrisant la **complexité** et en préservant la **stabilité** des clusters déjà formés. La fonction $\text{addEntity}(\mathcal{E}_{\text{new}})$ opère ainsi une **initialisation** des liaisons $\omega_{(\text{new}),j}$ pour un sous-ensemble d'entités j présélectionnées, soit par **échantillonnage**, soit via un **choix** des k plus proches voisins.

A. Idée Générale de l'Insertion Incrémentale

Lorsque survient un objet nouveau \mathcal{E}_{new} , on augmente la **matrice** ω d'une ligne et d'une colonne, les entrées associées à $\omega_{(\text{new}),j}$ et $\omega_{j,(\text{new})}$ (selon le caractère orienté ou non du SCN) étant initialisées à zéro ou à de faibles valeurs. La question est de savoir **comment** affecter ces pondérations initiales pour ne pas perturber la structure, ni avoir à recourir à un calcul complet $O(n)$ pour tous les liens possibles.

Mathématiquement, on veut d'emblée attribuer des poids $\omega_{(\text{new}),j}(0)$ judicieux pour un *petit* sous-ensemble de nœuds existants. On laisse ensuite la **dynamique** DSL (cf. chap. 2.2.2) continuer son ajustement au fil du temps, de sorte que $\omega_{(\text{new}),j}$ s'établisse ou non selon la **synergie** $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ et les mécanismes d'inhibition, de saturation, etc.

B. Pourquoi une Initialisation Sélective ?

Le but est triple. D'abord, on souhaite **réduire** le **coût** d'initialisation, car calculer $\omega_{(\text{new}),j}$ pour **tous** les nœuds $j \in \{1, \dots, n\}$ implique un nombre potentiellement très grand de comparaisons ou de mesures de synergie. Ensuite, on vise à **limiter** la perturbation globale : raccorder \mathcal{E}_{new} à tous les nœuds induirait un trop grand nombre de liaisons non nulles au départ, risquant de "secouer" excessivement la structure. Enfin, il est souvent plus **efficace** de connecter \mathcal{E}_{new} à un nombre restreint de nœuds "prometteurs", puis de laisser la **mise à jour** DSL créer ou détruire d'autres liens si la synergie l'exige.

C. Stratégies d'Initialisation : Échantillon vs k Plus Proches Voisins

Deux approches principales se distinguent pour choisir les nœuds auxquels la nouvelle entité sera reliée initialement.

On définit un **sous-ensemble** $S \subseteq \{1, \dots, n\}$ d'entités déjà présentes (par exemple, un échantillon aléatoire de taille α si α est petit). On calcule la **synergie** $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ uniquement pour $j \in S$, puis on affecte une pondération initiale :

$$\omega_{(\text{new}),j}(0) = \eta_{\text{init}} S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) \quad \text{pour } j \in S, \quad \omega_{(\text{new}),j}(0) = 0 \quad \text{sinon.}$$

La constante $\eta_{\text{init}} > 0$ sert à **normaliser** ou à brider la valeur initiale, assurant que $\omega_{(\text{new}),j}(0)$ ne soit pas trop élevée ni trop faible. Ainsi, la nouvelle entité se trouve *branchée* de manière limitée, ce qui permet de **tester** la cohérence avec quelques nœuds représentatifs. Au fil des itérations DSL, si la synergie S se révèle forte avec d'autres entités absentes de S , on peut voir apparaître des liens supplémentaires (cf. chap. 7.4 sur l'inhibition, chap. 7.2.2 sur la création de liaisons).

Cette méthode consiste à **identifier** les k nœuds existants les plus “pertinents” pour \mathcal{E}_{new} . La **pertinence** se définit souvent via la **maximisation** de $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ (par exemple la plus haute similarité ou la plus faible distance). En pratique, on :

Calcule $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ pour chaque j déjà présent (ou on emploie des techniques d'**approximate nearest neighbors** pour accélérer).

Range ces valeurs pour en extraire les k plus grandes.

Fixe $\omega_{(\text{new}), j}(0) = \eta_{\text{init}} S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ pour ces k entités, et 0 ailleurs.

Ce faisant, on **assure** que \mathcal{E}_{new} se lie initialement aux entités jugées *compatibles* ou *synergiques*, favorisant son insertion **rapide** dans un cluster approprié. Le **coût** en calcul demeure $O(n)$ ou $O(n \log n)$ si l'on doit trier la liste ou recourir à une structure d'indexation. Cette complexité reste plus modeste qu'une reconfiguration complète du SCN, notamment lorsque $k \ll n$.

D. Suite de la Dynamique DSL

Après cette **initialisation**, la nouvelle entité \mathcal{E}_{new} poursuit sa trajectoire dans la **dynamique** DSL : pour chaque lien retenu (ou éventuellement créé a posteriori), on itère

$$\omega_{(\text{new}), j}(t+1) = \omega_{(\text{new}), j}(t) + \eta [S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) - \tau \omega_{(\text{new}), j}(t)].$$

La structure environnante (clusters, liens d'autres entités) peut se réajuster localement, selon la logique d'auto-organisation (cf. chap. 2.2.3, chap. 7.4). Les liens initiaux jugés peu pertinents décroîtront rapidement, tandis que des liens non initialisés à l'origine peuvent apparaître si l'algorithme le permet (par exemple par un mécanisme de “création” de liaison lors de la mise à jour, voir chap. 7.2.2 ou 7.5.1).

Conclusion

La fonction `addEntity(\mathcal{E})` répond à la nécessité d'**insérer** une nouvelle entité dans un SCN en **apprentissage continu**, en réduisant le **coût** de calcul et en préservant la **cohérence** des clusters déjà présents. Le **choix** d'initialiser $\omega_{(\text{new}), j}(0)$ soit sur un **échantillon** de nœuds, soit sur un ensemble de **k plus proches voisins**, permet une **connexion** partielle de \mathcal{E}_{new} . On évite un paramétrage systématique de toutes les liaisons, qui serait exponentiellement coûteux et risquerait de provoquer un **remaniement** global intempestif. Dans cette optique, la valeur de k (ou la taille de l'échantillon) se choisit selon la **dimension** du réseau, la **probabilité** que l'entité s'insère dans un cluster existant, et les **contraintes** de complexité. L'expérience montre qu'une telle insertion **incrémentale** — couplée à la suite de la mise à jour DSL (inhibition éventuelle, ajustement local) — aboutit à une **intégration** fluide des nouveaux nœuds, garante d'une **auto-organisation** stable et évolutive du SCN.

7.6.2.2. `removeEntity(E)` : Suppression ou Isolement Progressif

Dans un **Synergistic Connection Network** (SCN) à **apprentissage continu**, il arrive qu'une entité \mathcal{E} doive être **retirée** pour diverses raisons (obsolescence, capteur inopérant, utilisateur n'ayant plus d'activité, etc.). Contrairement au mode “batch” où le jeu d'entités reste fixe, on doit ici envisager une **désactivation** graduelle ou un **isolement** progressif de \mathcal{E} afin de préserver la **cohérence** globale des clusters et de limiter les “chocs” sur la structure existante. La fonction `removeEntity(\mathcal{E})` correspond à cette étape de **suppression** gérée de manière incrémentale.

A. Problématique de la Suppression

Lorsque l'on décide de retirer \mathcal{E} du SCN, on pourrait être tenté de **l'effacer** brusquement de la matrice ω . Mais cette action peut fortement **perturber** les clusters qui comptaient sur \mathcal{E} pour maintenir leur cohésion. Une **coupure** immédiate de tous ses liens $\{\omega_{\mathcal{E}, j}\}$ peut générer un *choc* dans l'**organisation**, forçant le réseau à se réadapter de façon violente. À l'inverse, un **isolement** progressif (lentement ramener $\omega_{\mathcal{E}, j}$ vers 0) laisse aux autres nœuds le temps de s'ajuster graduellement et d'éviter un dérèglement massif des clusters.

Une **suppression** abrupte ($\omega_{\mathcal{E}, j} \rightarrow 0$ pour tous j) en une seule itération peut briser des liens cruciaux pour la **cohésion** de certaines communautés. Les entités $\{j\}$ auparavant fortement liées à \mathcal{E} se retrouvent sans soutien, devant trouver d'autres partenaires (ou réajuster leurs poids) en un court laps de temps. Cela peut entraîner des oscillations plus amples ou un **rééquilibrage** brutal.

On préfère souvent échelonner la **disparition** de \mathcal{E} sur plusieurs itérations de la dynamique DSL. On peut, par exemple, décider qu'à chaque pas :

$$\omega_{\mathcal{E}, j}(t+1) = \omega_{\mathcal{E}, j}(t) + \eta[S(\mathcal{E}, j) - \tau \omega_{\mathcal{E}, j}(t)] + \Delta_{\text{remove}}(t),$$

où $\Delta_{\text{remove}}(t) \leq 0$ est un terme spécifique qui vient tirer $\omega_{\mathcal{E}, j}$ vers 0. Cette force négative peut être proportionnelle à $\omega_{\mathcal{E}, j}(t)$ (décroissance exponentielle) ou un facteur constant (décroissance linéaire). Après un nombre d'itérations, $\omega_{\mathcal{E}, j} \approx 0$. Les autres entités ont alors le temps de **compenser**, renforçant d'autres liens selon la règle DSL standard.

B. Mécanismes Concrets de Suppression

Plusieurs approches permettent ce retrait graduel, selon la **vitesse** et l'**envergure** de la suppression souhaitée.

On définit un intervalle ou un point de départ T_{remove} . À partir de ce moment, on déclenche la décroissance $\Delta_{\text{remove}}(t)$:

547.Décroissance exponentielle :

$$\Delta_{\text{remove}}(t) = -\alpha \omega_{\mathcal{E}, j}(t),$$

de sorte qu'à chaque itération, $\omega_{\mathcal{E}, j}$ soit diminué proportionnellement à sa valeur.

548.Décroissance linéaire :

$$\Delta_{\text{remove}}(t) = -\delta,$$

coupant un petit delta fixe à chaque étape.

Une fois $\omega_{\mathcal{E}, j}(t)$ retombé sous un seuil θ , on peut le mettre à 0 définitivement et ne plus le recalculer. Cette phase de “dissipation” s’étale sur T_{dissip} itérations, évitant ainsi un effondrement brutal.

Si un **mécanisme** d’inhibition latérale est en place (cf. chap. 7.4), on peut augmenter localement γ pour l’entité \mathcal{E} , rendant plus difficile le maintien de liens $\omega_{\mathcal{E}, j}$. Cette entité subit alors une **pénalisation** plus grande, incitant chaque liaison à baisser plus vite dans la dynamique DSL standard, ce qui génère un “isolement” de \mathcal{E} sans reprogrammer explicitement une décroissance dans Δ_{remove} .

Si l’on tolère un *choc* plus prononcé, on peut imposer un **seuil** θ croissant : dès qu’une liaison $\omega_{\mathcal{E}, j}(t)$ passe en deçà de $\theta(t)$, on la coupe (“hard threshold”). Par exemple, on initialise $\theta(0) = \theta_0$ et on l’augmente au fil du temps, forçant la quasi-totalité des liens de \mathcal{E} à se retrouver sous ce nouveau seuil et à être **basculés** à zéro. Même si cela reste plus abrupt qu’une décroissance continue, cela peut s’avérer moins complexe à implémenter.

C. Conséquences sur les Clusters

À mesure que $\omega_{\mathcal{E}, j}$ décroît, les noeuds $\{j\}$ qui étaient fortement corrélés à \mathcal{E} doivent **réorienter** leurs connexions : la dynamique DSL fait qu’ils vont renforcer d’autres liens ou retomber dans d’autres clusters plus stables. Le **réseau** subit un changement localisé, sans bouleversement massif, car la décroissance est programmée sur plusieurs itérations. Les entités $\{j\}$ apprennent à se passer de \mathcal{E} . Finalement, quand $\omega_{\mathcal{E}, j} \approx 0$ pour tous j , la disparition de \mathcal{E} est effective ; on peut alors **physiquement** la retirer de la structure de données, libérant la mémoire et les calculs associés.

D. Temps de Transition et Paramétrage

Le **choix** de la vitesse de suppression (exponentielle vs linéaire, paramètre α ou δ) dépend de la **tolérance** du système : un retrait en 10 itérations sera plus brusque qu’un retrait en 100 itérations. On peut de même appliquer un facteur de

“smoothness” limitant les oscillations. Le schéma d’une “entité en déclin” ressemble alors à un mode passif où \mathcal{E} n’échange plus que de faibles pondérations, jusqu’à son **effacement** complet.

E. Remplacement ou Recyclage

Dans certains scénarios, une entité \mathcal{E} à retirer peut-être **remplacée** par une nouvelle entité \mathcal{E}' . On “réutilise” alors l’**index** ou l’**ID** dans la matrice ω , mais en réinitialisant ses valeurs, comme décrit en § 7.6.2.1. On assure ainsi un usage efficient des ressources (ex. si la matrice ω est partagée ou si l’on a un nombre fixe de slots). À l’inverse, si l’on veut conserver la mémoire “physique” d’un SCN potentiellement extensible, on supprime purement et simplement \mathcal{E} après isolement.

Conclusion

La fonction `removeEntity(\mathcal{E})` dans un SCN **ouvert** s’apparente à un **isolement progressif** plutôt qu’à un **coup de gomme** brutal. On laisse la dynamique DSL, éventuellement modulée par un terme $\Delta_{\text{remove}}(t)$ ou une **inhibition** plus marquée, **ramener** les liaisons $\omega_{\mathcal{E},j}$ vers 0 sur plusieurs itérations. Ce **retrait** en douceur procure un **temps d’adaptation** pour les autres entités $\{j\}$, conservant la **stabilité** des clusters et évitant un **rééquilibrage** précipité. Une fois \mathcal{E} virtuellement isolée, on la **retire** définitivement de la matrice ω . Ainsi, le **Deep Synergy Learning** demeure conforme à l’**esprit** d’un SCN “vivant”, capable de gérer l’**arrivée** (cf. § 7.6.2.1) et la **disparition** des noeuds dans un environnement évolutif.

7.6.2.3. Conflits si Trop de Changements en Peu de Temps ?

Dans un **SCN** (Synergistic Connection Network) maintenu en **apprentissage continu**, il arrive parfois qu’un **trop grand** nombre de modifications (insertions ou retraits d’entités) surviennent dans un court intervalle de temps. Un afflux rapide de **nouvelles** entités $\{\mathcal{E}_{n+1}, \dots\}$ ou une vague de **suppression** simultanée de plusieurs entités peut engendrer des **conflits** dans la dynamique DSL, susceptibles de provoquer des **oscillations** ou une désorganisation soudaine. Cette section décrit la **raison** de ces conflits et propose des mécanismes pour **atténuer** ou **éviter** le phénomène.

A. Accumulation de Mises à Jour Simultanées

Lorsqu’on insère ou retire beaucoup d’entités au sein d’un **même** laps de temps, on opère un grand nombre de modifications locales :

- **Insertion** (`addEntity(\mathcal{E}_{new})`, voir § 7.6.2.1) : on calcule et initialise $\omega_{(\text{new}),j}(0)$ pour un sous-ensemble de noeuds $\{j\}$, puis on enclenche la dynamique DSL sur ces liaisons.
- **Retrait** (`removeEntity(\mathcal{E}_{old})`, voir § 7.6.2.2) : on réduit $\omega_{(\text{old}),j}$ vers 0 sur plusieurs itérations, impliquant une réorganisation des clusters précédemment liés à \mathcal{E}_{old} .

Si **plusieurs** insertions/suppressions se produisent *quasi en même temps*, la **dynamique** DSL doit gérer un nombre considérable de changements potentiels, chacun pouvant impacter localement les pondérations $\omega_{i,j}$. On assiste alors à une **accumulation** de mises à jour dont les effets se recouvrent ou s’interfèrent, rendant la configuration $\omega(t)$ instable pendant un certain temps, voire sujette à des **oscillations** si ces mises à jour se contredisent.

B. Indécisions et Oscillations

Lorsqu’un trop grand nombre d’entités arrivent soudainement dans le réseau, elles vont chercher à se **connecter** aux clusters préexistants. Une **inhibition** (chap. 7.4) ou un seuil imposant peut alors **supprimer** ou **réduire** drastiquement une partie de leurs liaisons. Le **réseau** pourrait alterner entre :

- Liaisons moyennes pour plein de noeuds,
- Un filtrage brutal (inhibition, saturation) supprimant la plupart de ces liaisons,
- Un rétablissement partiel, etc.

D'un point de vue **mathématique**, la somme $\sum_{(i,j)} \omega_{i,j}$ peut fluctuer brusquement, perturbant la "sélectivité" du réseau. Sans phase d'ajustement suffisamment longue, la configuration $\boldsymbol{\omega}(t)$ risque de sauter d'un état partiellement formé à un autre état provisoire.

De même, si on **retire** simultanément de nombreuses entités, on ôte tout à coup un volume important de liaisons, ce qui peut provoquer une **désagrégation** des clusters existants. La dynamique DSL doit "recoller les morceaux" en un temps court, suscitant parfois une sur-segmentation éphémère et une reconstitution progressive de groupes.

C. Stratégies d'Évitement ou d'Amortissement

Pour prévenir les conflits liés à un rythme de changements trop rapide, plusieurs **solutions** existent :

On peut **limiter** le nombre K_{\max} d'entités insérables ou supprimables par tranche de temps. Concrètement, si le système reçoit dix demandes d'ajout d'entités, il n'en accepte que deux ou trois immédiatement, puis attend quelques itérations (suffisantes pour un début de stabilisation) avant de procéder aux suivantes. De la même façon pour la suppression, on peut éviter de retirer plus de ℓ_{\max} entités en même temps.

Lorsque beaucoup d'ajouts/retraits ont lieu, on peut **baisser** temporairement le taux d'apprentissage η ou la température (si un recuit simulé est présent, chap. 7.3). Cela ralentit la dynamique DSL, diminuant l'amplitude des variations $\Delta\omega_{i,j}$ à chaque pas, et évite que le réseau **oscille** trop violemment.

De manière analogue, pour les entités fraîchement insérées, on peut imposer un "warm-up" : on initialise $\omega_{(\text{new}),j}$ à de très petites valeurs, permettant une progression **graduelle** dans la matrice $\boldsymbol{\omega}$. Ainsi, on amortit l'impact initial d'une arrivée massive de nœuds.

Les algorithmes DSL peuvent prévoir des **phases** de stabilisation (ou "cool-down") après un certain nombre d'ajouts/suppressions, où aucune nouvelle entité ne s'insère et où l'on réduit le bruit ou la vitesse de mise à jour, pour laisser le SCN se consolider. D'un point de vue **mathématique**, on fixe $\eta \approx 0$ ou on désactive l'insertion/suppression pendant un court intervalle, comme un palier de "repos" qui vise à faire baisser les tensions dans $\boldsymbol{\omega}$.

D. Exemple Mathématique Simplifié

Supposons que, dans un bref intervalle, on **ajoute** 20 entités et on en **retire** 10. Chacune des 20 entités insérées se connecte à un voisinage de taille $O(k)$. Simultanément, les 10 entités retirées commencent à voir leurs poids $\omega_{(\text{old}),j}$ chuter. Cela signifie qu'un volume $O((20 + 10) \times k)$ de liaisons $\omega_{i,j}$ subit un **changement** notable en très peu de pas d'itération. Cette **accumulation** d'influences peut rendre la trajectoire $\boldsymbol{\omega}(t)$ très instable, incitant à une insertion plus "séquencée" (quelques entités à la fois), ou à diminuer η pour contrôler la vitesse de bascule.

Conclusion

Lorsque **trop** d'insertions et de retraits se produisent en peu de temps dans un **Synergistic Connection Network**, on peut faire face à des **conflits** et des **oscillations**, remettant en cause la **stabilité** du SCN. Pour pallier ce problème, on peut (1) **borner** ou **stager** la cadence d'arrivées/suppressions, (2) **moduler** la mise à jour (baisser η ou imposer un "cool-down") et (3) veiller à un mécanisme d'**inhibition** ou de "warm-up" plus progressif pour les entités entrantes. Ces précautions assurent à la **dynamique** DSL le temps nécessaire pour s'ajuster en douceur, évitant les écueils d'un flux de modifications simultanées trop massif.

7.6.3. Évasion des Minima Locaux à Long Terme

Lorsque le SCN (Synergistic Connection Network) fonctionne sur des durées prolongées, il arrive qu'après une phase de convergence initiale, le réseau se **fossilise** dans une configuration locale stable, mais pas nécessairement optimale à l'échelle globale (cf. §7.2.2). Pour **ré-ouvrir** le système à de nouvelles potentialités de clusters ou à la réaffectation de liens, on recourt à des **perturbations stochastiques** (un "mini recuit") ou à un mécanisme de "vigilance" réactivant la dynamique dans les zones figées.

7.6.3.1. Petite Injection Stochastique Périodique (Mini Recuit) pour Éviter l’Enlisement

Un **Synergistic Connection Network** (SCN) amené à fonctionner au long cours peut, à la suite d'une période de mise à jour, se figer dans un **minimum local** ou dans un **cluster** de stabilité relative dont il n'arrive plus à sortir. Une façon d'éviter cet **enlisement** consiste à injecter périodiquement un **bruit stochastique** de faible amplitude dans les pondérations $\omega_{i,j}$. Cette pratique s'apparente au **recuit simulé** (chap. 7.3) mais appliquée de façon **localisée** ou **ponctuelle**, et vise à **redonner** une capacité d'exploration au réseau sans maintenir un bruit permanent.

A. Principe Général

Le **DSL** (Deep Synergy Learning) met à jour les pondérations $\omega_{i,j}$ selon la formule :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \Delta_{\text{noise}}(i,j,t),$$

où le terme $\Delta_{\text{noise}}(i,j,t)$ est **nul** la plupart du temps, mais prend une **valeur** aléatoire à des intervalles fixés (par exemple, toutes les K itérations). Le bruit Δ_{noise} peut être d'amplitude σ relativement petite, selon une loi gaussienne $\mathcal{N}(0, \sigma^2)$ ou uniforme, et ne s'appliquer qu'à un **sous-ensemble** des liens $\omega_{i,j}$ pour restreindre son **impact**.

Le **bruit** injecté consiste en une perturbation **aléatoire** :

$$\Delta_{\text{noise}}(i,j,t) = \begin{cases} \delta_{i,j}(t), & \text{si } t \equiv 0 \pmod K \text{ et } (i,j) \in \text{Sample}, \\ 0, & \text{sinon,} \end{cases}$$

où $\delta_{i,j}(t) \sim \mathcal{N}(0, \sigma^2)$ ou toute autre distribution centrée (± 1 , etc.), et $\text{Sample} \subseteq \{(i,j)\}$ est un ensemble de liaisons tiré au hasard ou choisi selon un critère. Cela évite d'appliquer le bruit à toutes les pondérations, ce qui serait trop déstabilisant.

Le but est de **réveiller** périodiquement la dynamique : même si un cluster localement stable retient le réseau, ce *mini* bruit peut pousser certaines liaisons hors de leur “vallée” d'énergie, permettant une nouvelle **exploration** et la découverte d'autres configurations ω . Contrairement au recuit simulé “complet” (où on maintient un bruit continu durant tout l'apprentissage, cf. chap. 7.3), on n'a ici qu'une impulsion intermittente, moins coûteuse.

B. Pourquoi cette Injection ?

Le **SCN** recèle souvent des **minima locaux** : la somme des pondérations $\{\omega_{i,j}\}$ est stabilisée en un certain agencement de clusters, mais pas forcément un *minimum global*. En insérant une perturbation ponctuelle, on fournit une échappatoire pour que quelques liaisons $\omega_{i,j}$ se dégagent de leurs valeurs figées et puissent donner lieu à un réarrangement partiel.

D'un point de vue **mathématique**, la trajectoire $\omega(t)$ suit normalement un **gradient** ou une logique de descente (plus ou moins modifiée par l'inhibition ou la saturation). Sans bruit, une fois dans un bassin local, la dynamique se fige ; avec un “mini recuit”, on injecte une **composante stochastique** favorisant un saut vers un autre bassin si c'est profitable.

C. Dosage et Fréquence

Pour ne pas **détraquer** la convergence, on choisit soigneusement :

Amplitude σ : trop faible, et le bruit ne suffit pas à libérer un piégeage significatif ; trop fort, et la structure peinera à se stabiliser. Une règle empirique est de commencer avec une σ modérée puis de la **réduire** au fil du temps (un “refroidissement” partiel).

Périodicité K : on peut décider de déclencher le “shake” toutes les K itérations, ou toutes les K_j itérations par sous-groupe de liaisons, etc. Cette périodicité doit être **assez espacée** pour laisser la descente DSL faire son effet entre deux “secousses”.

On peut, par exemple, décroître σ selon $\sigma(t) = \sigma_0/(1 + \alpha t)$ ou un autre schéma inspiré du recuit simulé (chap. 7.3). L'importance est de **combiner** la réinjection de bruit avec un mécanisme **local** de stabilisation pour ne pas maintenir en permanence un état chaotique.

D. Avantages et Limites

L'introduction de secousses stochastiques permet une **évasion des minima locaux**, évitant qu'une séquence DSL standard ne s'enlise dans un cluster sous-optimal. Cette approche apporte également une **complémentarité** intéressante : il s'agit d'un **ajout léger**, moins intrusif qu'un recuit continu, et plus facile à paramétrier. De plus, elle offre une **possibilité de contrôle fin**, permettant de cibler un **sous-ensemble spécifique de liaisons**, notamment celles qui sont restées faibles ou stables trop longtemps, afin d'initier un réexamen sélectif de leur pertinence.

Toutefois, cette méthode comporte certains risques. Un **bruit trop intense ou trop fréquent** peut compromettre la convergence, empêchant le SCN de se stabiliser et le forçant à osciller indéfiniment. De plus, le **réglage des paramètres** est délicat : il faut soigneusement choisir l'**amplitude σ** , la **périodicité K** , ainsi que la proportion de liaisons "secouées", sous peine de provoquer une désorganisation chronique du réseau.

E. Illustration Mathématique Simplifiée

Supposons un "**mini recuit**" toutes les K itérations :

$$\Delta_{\text{noise}}(i, j, t) = \begin{cases} \mathcal{N}(0, \sigma^2), & \text{si } t \equiv 0 \pmod{K} \text{ et } (i, j) \in \text{Sample}, \\ 0, & \text{sinon.} \end{cases}$$

où Sample est un ensemble de liens tirés aléatoirement de taille μn (quelques pourcents de l'ensemble total). On exécute la règle DSL usuelle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)] + \Delta_{\text{noise}}(i, j, t).$$

D'un point de vue **théorique**, on espère qu'avec une amplitude σ "raisonnable", cette perturbation **conduit** la suite $\{\omega(t)\}$ à s'éloigner ponctuellement d'un attracteur local pour en "explorer" d'autres. Un argument analogue à la théorie de recuit (chap. 7.3) suggère que cette perturbation peut aider, sous conditions, à trouver une structure plus optimale.

Conclusion

La **petite injection stochastique périodique** ou "mini recuit" est un mécanisme **supplémentaire** permettant à un SCN de s'extraire de minima locaux ou d'états d'inertie. Au lieu de maintenir un bruit continu (comme dans un recuit standard), on opère des **secousses** brèves, contrôlées, à intervalles de temps réguliers. Ce **compromis** offre un moyen de **raviver** la dynamique DSL quand elle semble s'être figée, tout en **limitant** l'impact négatif d'un bruit permanent sur la convergence. Pour être efficace, on règle l'amplitude σ et la fréquence K de sorte à **ne pas** rendre le réseau trop chaotique, ni trop vite replongé dans l'immobilité : on obtient alors une **dynamique** DSL capable de se **réorganiser** ponctuellement et, à la longue, de **découvrir** des configurations de clusters plus stables ou plus avantageuses.

7.6.3.2. Contrôle par "Vigilance" : si un Cluster Stagne Trop, on Force un Petit "Reset"

Un **Synergistic Connection Network** (SCN) fonctionnant en **apprentissage continu** peut finir par **stagner** dans un agencement local de pondérations $\omega_{i,j}$ ne reflétant plus l'évolution du contexte. Même si certains mécanismes (recuit, inhibition) préviennent la prolifération de liaisons moyennes ou l'enlisement global (voir § 7.3 et § 7.4), il reste des situations où un **cluster** (ou sous-ensemble de noeuds) se fige de manière **excessive**. C'est ici qu'intervient la **vigilance** : détecter qu'un bloc ne se **renouvelle** plus et imposer un "mini reset" local. Cette logique s'inspire de la **théorie ART (Adaptive Resonance Theory)** et d'autres approches neuronales où une "**vigilance**" signale la nécessité de **réorganiser** un sous-groupe inactif.

A. Fondements Mathématiques du “Vigilance Reset”

Le principe : on observe un ou plusieurs **clusters** $\mathcal{C}_\alpha \subseteq \{1, \dots, n\}$ et l'on analyse leur évolution au fil des itérations t . S'ils **stagnent** trop (aucune modification notable de leur répartition ou de leurs liaisons), on applique un **reset** de leurs poids $\omega_{i,j}$. Ainsi, on autorise une ré-exploration de cette zone, libérant un cluster qui s'était *enlisé*.

Le **SCN**, en suivant la dynamique DSL (voir chap. 2.2), modifie $\omega_{i,j}(t)$ à chaque itération. Si un cluster \mathcal{C}_α (un sous-groupe de nœuds) ne change plus de composition interne depuis trop longtemps, on vérifie par exemple :

La **somme** interne des liaisons :

$$\Omega(\mathcal{C}_\alpha, t) = \sum_{i,j \in \mathcal{C}_\alpha} \omega_{i,j}(t).$$

Si $|\Omega(\mathcal{C}_\alpha, t+1) - \Omega(\mathcal{C}_\alpha, t)| \approx 0$ sur plusieurs itérations, c'est un signe de figement.

La **stabilité** du sous-groupe : le set \mathcal{C}_α reste identique ou quasi identique, sans entité entrant ni sortant.

Une **variance** ou une **énergie** interne $\text{Var}(\mathcal{C}_\alpha, t)$ stagnant depuis un certain horizon Δt .

Si ces indicateurs **stagnent** en deçà d'un **seuil** $\varepsilon_{\text{stagn}}$, et ce pendant Δt itérations, on considère que \mathcal{C}_α s'est “enlisé”.

La vigilance déclenche un **reset** :

$$\omega_{i,j}(t+1) \leftarrow \omega_{\text{init}}(i,j) \quad \text{pour } i,j \in \mathcal{C}_\alpha,$$

ou on leur injecte un **bruit** (cf. § 7.6.3.1). De cette façon, on “casse” la configuration du cluster, permettant une “réinjection” de plasticité. Le **reste** du réseau demeure inchangé, donc le **choc** est localisé. Dans une approche plus radicale, on peut même **dissoudre** \mathcal{C}_α en mettant ses liaisons à zéro, et laisser la dynamique DSL reconstituer, si nécessaire, de nouveaux liens.

B. Justification et Avantages

B. Justification et Avantages

Malgré les mécanismes d'inhibition et de bruit global, un DSL peut rester **piégé** dans un *bassin* local pour un sous-groupe donné. Le **vigilance reset** introduit une opportunité de **réouverture**, permettant à ce groupe de **réévaluer ses liaisons** avec les entités voisines. Cette reconsideration peut aboutir à un **regroupement plus optimal**, facilitant l'évasion des sous-minima locaux.

Contrairement à une réinitialisation complète du SCN, cette méthode se limite à la **zone stagnante** \mathcal{C}_α . Le cluster concerné peut être effacé ou perturbé, mais sans compromettre l'**historique des liaisons** dans le reste du réseau. Cette approche **selective**, qui isole uniquement la partie problématique, permet ainsi de préserver la **stabilité générale** du SCN.

Dans un cadre de **flux continu**, de nouvelles entités peuvent apparaître (§ 7.6.2.1). Si un cluster figé les **ignore**, il risque de manquer une adaptation cruciale. La vigilance permet de **déetecter cette non-réactivité**, de réinitialiser \mathcal{C}_α , et d'optimiser l'intégration des nouvelles arrivées, garantissant ainsi une meilleure adaptation du réseau face aux évolutions dynamiques.

C. Considérations : Paramétrage et Coordination

Le **critère de stagnation** doit être précisément défini en fonction de la **durée** Δt pendant laquelle un cluster reste inchangé, ainsi que d'un seuil $\varepsilon_{\text{stagn}}$. Il est également possible d'exiger un **nombre minimum d'entités stables** avant d'activer un reset.

L'**amplitude du reset** peut varier : faut-il remettre les poids $\omega_{i,j}$ à zéro, les réinitialiser à une petite valeur ω_{init} , ou bien injecter un **bruit stochastique** $\mathcal{N}(0, \sigma^2)$ pour maintenir une certaine variabilité ?

La **fréquence de contrôle** est également un paramètre clé. Le reset doit-il être évalué à **chaque itération**, ou seulement toutes les X itérations ? Un suivi trop fréquent peut entraîner un **coût computationnel élevé**, car il implique de surveiller en permanence l'évolution des sous-clusters.

Enfin, l'**interaction avec d'autres mécanismes** comme l'**inhibition** ou le **recuit simulé** doit être prise en compte. Si un recuit global est déjà activé (cf. chap. 7.3), un reset vigilance **plus léger** peut suffire. À l'inverse, la vigilance peut être utilisée en **dernier recours**, lorsque ni le bruit continu ni l'inhibition n'ont permis d'éviter un figement local du SCN.

D. Exemple Mathématique

Supposons qu'un cluster \mathcal{C}_α regroupe ℓ entités depuis longtemps. Sa **somme** interne $\Omega(\mathcal{C}_\alpha, t)$ varie de moins de $\varepsilon_{\text{stagn}}$ sur une fenêtre de Δt itérations. La fonction vigilance décide :

Reset :

$$\omega_{i,j}(t+1) = \text{rand}(0, \omega_{\max}) \quad \text{ou} \quad 0, \quad i, j \in \mathcal{C}_\alpha.$$

Relancer la dynamique DSL pour ℓ entités reconfigurées. Les entités peuvent alors, au pas suivant, se distribuer dans d'autres clusters si elles découvrent de meilleures synergies $\{S(\mathcal{E}_i, \mathcal{E}_k)\}$.

Conclusion

Le **contrôle par “vigilance”** vient compléter l'éventail des solutions pour **maintenir** un **apprentissage continu** dans un SCN. Dès qu'un cluster (ou un groupe de noeuds) se **fige** trop longtemps — en dépit de l'évolution du reste du réseau ou de l'arrivée de nouveaux noeuds —, on lui impose un **“mini reset”**, disséminant ou réinitialisant ses liaisons, pour le forcer à **re-négocier** sa structure. Cela permet de **rompre** un figement local que ni l'inhibition, ni le bruit global (recuit), ni les insertions/suppressions de noeuds ne suffisaient à dissoudre. Un tel dispositif assure au SCN une **flexibilité** sur la durée, évitant la “fossilisation” de certaines parties du réseau et maintenant l'**évolutivité** nécessaire en **apprentissage continu**.

7.6.3.3. Exemples en Robotique ou en IA Symbolique

Les **méthodes** décrites en § 7.6.3.1 (injection stochastique) et § 7.6.3.2 (vigilance par reset partiel) trouvent de nombreuses applications dans des **domaines** où un **SCN** (Synergistic Connection Network) fonctionne à long terme, et où il est crucial d'éviter un **enlisement** dans des configurations localement satisfaisantes mais sous-optimales. Deux **cas** emblématiques sont présentés ci-après : (A) un **essaim** de robots collaboratifs en robotique, (B) un **réseau** de règles logiques en IA symbolique. Dans les deux contextes, le principe reste identique : introduire périodiquement un **bruit** ou un **reset** dans les pondérations $\omega_{i,j}$, de manière à permettre une **réouverture** de la recherche pour sortir d'un minimum local stable.

A. Exemple en Robotique : Essaim de Robots Collaboratifs

Les **essaims** de robots mobiles, chacun équipé de capteurs (caméra, lidar, ultrasons, etc.), doivent souvent s'**auto-organiser** pour accomplir des tâches collectives (surveillance, exploration, formation de figures, etc.). Le SCN peut représenter la **coopération** entre robots via des pondérations $\omega_{i,j}$, chacune mesurant la force de liaison (ou de synergie) entre deux robots i et j . Une **haute** valeur $\omega_{i,j}$ signifie que les robots i, j coopèrent intensivement (partage de positions, alignement de trajectoire, communication fréquente).

En pratique, un **groupe** de robots peut s'organiser rapidement en un cluster partiel (p. ex. un ensemble A couvre la zone nord, un autre B la zone sud). Il se peut alors qu'un arrangement plus optimal (répartition en trois sous-groupes, ou redistribution différente) soit bloqué par la dynamique DSL, qui converge localement. Cet état, somme toute satisfaisant, n'est pas nécessairement le *vrai* optimum global en termes de couverture, consommation d'énergie, ou robustesse.

Sans **perturbation** additionnelle, l'**auto-organisation** finira par se figer. Or, si l'environnement est **changeant** (nouveaux obstacles, zones à surveiller déplacées), l'essaim doit se **réorganiser** plus profondément qu'une simple adaptation locale.

L'**injection stochastique**, ou **mini-recuit**, consiste à ajouter un **bruit aléatoire** Δ_{noise} sur certaines pondérations $\omega_{i,j}$ à intervalles réguliers. Un sous-ensemble de liaisons est sélectionné pour recevoir une perturbation gaussienne $\mathcal{N}(0, \sigma^2)$. Ce *shake* permet de **défiger** des liens restés stables, entraînant un **recalcul des clusters** et empêchant un blocage structurel du réseau.

Le **contrôle de vigilance** repose sur une **surveillance de la stagnation** (cf. § 7.6.3.2). Si un groupe \mathcal{C}_α de robots ne modifie plus ses connexions depuis un certain temps, un **reset partiel** de ses pondérations ω est appliqué. Cette intervention oblige le sous-groupe à **reconsidérer** ses choix de partenaires (chaque robot étant une entité \mathcal{E}_i), évitant ainsi qu'il ne reste figé dans une configuration **obsolète**.

En **robotique**, cette ouverture par “shake” ou “vigilance” garantit une **exploration** plus globale, adaptable face à des changements soudains (ex. panne d'un robot, apparition d'une zone d'intérêt). D'un point de vue **mathématique**, la somme des pondérations $\sum_{i,j} \omega_{i,j}$ n'est plus un simple attracteur local : le bruit ou le reset local vient *rompre* la forme du paysage d'énergie, évitant la fossilisation du comportement collectif.

B. Exemple en IA Symbolique : Réseau de Règles Logiques

Les **entités** \mathcal{E}_i peuvent être des **règles** ou des **clauses** dans un système d'inférence symbolique. La **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ quantifie la compatibilité logique (cohérence, absence de contradiction) ou la force de co-occurrence d'idées. Les pondérations $\{\omega_{i,j}\}$ décrivent la **solidité** de l'association entre deux règles, favorisant des *clusters* de règles mutuellement compatibles.

La dynamique DSL, par descente locale, peut regrouper certaines règles en un ensemble \mathcal{C}_α qui semble stable, mais n'exploite pas nécessairement toute la puissance du système (deux autres clusters pourraient mieux couvrir l'espace de solutions). Parfois, une **contradiction** latente ne se résout pas car le cluster incriminé n'interagit plus assez avec d'autres règles pour la révéler ou la corriger.

De la même manière qu'en robotique, deux mécanismes peuvent être appliqués pour éviter le figement des structures.

Le **mini-recuit** introduit périodiquement, toutes les K itérations, un **bruit** $\Delta_{\text{noise}}(i, j)$ sur les pondérations $\omega_{i,j}$. Cette perturbation permet à certaines règles extérieures de **se renforcer** si une contradiction devient plus apparente, ou inversement, d'affaiblir des liens incohérents qui auraient persisté artificiellement.

La **vigilance de cluster logique** repose sur la **détection d'un sous-ensemble** \mathcal{C}_α de règles dont la somme interne $\sum_{i,j \in \mathcal{C}_\alpha} \omega_{i,j}$ n'a plus évolué de manière significative. Un **reset local** est alors appliqué, ramenant les pondérations $\omega_{i,j}$ du cluster à de très faibles valeurs, ce qui autorise de nouveaux regroupements. Cette approche peut **révéler une structure plus générale** ou mettre en évidence une contradiction qui était jusqu'alors refoulée.

Ainsi, la **structure** de règles reste **flexible** à long terme et ne s'enferme pas définitivement dans une cohésion partielle.

Conclusion

Les **exemples** de robotique (essaim collaboratif) et d'IA symbolique (réseau de règles) mettent en évidence comment un SCN (Synergistic Connection Network) peut, **sur la durée**, **s'enliser** dans un état localement stable mais insuffisant. Pour s'en **délivrer**, on introduit :

549. **Injections stochastiques** périodiques (voir § 7.6.3.1),

550. **Un contrôle de vigilance** (voir § 7.6.3.2) déclenchant un reset partiel en cas de stagnation avérée.

Sur le plan **mathématique**, ces mécanismes équivalent à rompre temporairement la descente locale déterministe (ou l'auto-organisation habituelle) et permettre un saut dans un nouvel attracteur potentiel, évitant la fossilisation de la **structure**. Cela donne au SCN la **robustesse** nécessaire pour s'adapter à des environnements changeants, tant dans la collaboration robotique que dans la combinaison logique de règles en IA.

7.7. Couplage avec des Approches de Renforcement

Au sein d'un SCN, la mise à jour des pondérations $\omega_{i,j}$ se base principalement sur la **synergie** $S(i,j)$ entre entités (voir chapitres précédents). Cependant, dans des contextes complexes (robotique multi-agent, systèmes de décision adaptatifs, etc.), il peut s'avérer nécessaire d'**introduire un signal de récompense** externe (ou "score de performance") afin d'orienter la formation des liens ω vers des configurations qui **maximisent** une certaine utilité globale.

C'est précisément l'objet du **couplage** entre le DSL et les **approches de renforcement**. L'idée est de laisser la dynamique DSL opérer (renforcement/inhibition des liens), tout en prenant également en compte un **feedback** global ou partiel $\mathcal{R}(t)$, qui reflète la **qualité** de l'état courant. Dans ce chapitre 7.7, nous examinons :

- Comment injecter la **récompense** $\mathcal{R}(t)$ dans la mise à jour ω ?
- Quel rôle joue ce couplage dans un **système** multi-agent où chaque entité est un agent RL (Reinforcement Learning) ?
- Quels sont les **avantages** et **inconvénients** de mélanger les logiques de synergie DSL et de récompense extrinsèque ?

7.7.1. Gestion d'un Signal de Récompense

Dans la dynamique habituelle du DSL, la **mise à jour** de $\omega_{i,j}(t)$ s'écrit, pour chaque couple (i,j) , de la forme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] + \dots$$

où $S(i,j)$ est la **synergie** (ou similarité) entre i et j . Or, si on souhaite **intégrer** un **signal** $\mathcal{R}(t)$ (récompense globale ou partielle), il faut concevoir un **terme** supplémentaire ou une **modification** de la règle, de façon à privilégier les liens (ou clusters) qui **augmentent** \mathcal{R} .

7.7.1.1. Intégrer un $\mathcal{R}(t)$ Global ou Partiel, Reliant les Entités ω ?

Dans un **Synergistic Connection Network** (SCN), la logique **interne** (fondée sur la synergie $S(i,j)$, voir chap. 2.2) peut être complétée par un **signal de récompense** $\mathcal{R}(t)$ délivré par l'**environnement**. Cette récompense, qu'elle soit **globale** (un unique scalaire commun à tout le réseau) ou **partielle** (distincte pour certains nœuds ou sous-groupes), permet d'**orienter** la mise à jour des pondérations $\omega_{i,j}$ vers un **objectif** extrinsèque de performance ou de cohérence fonctionnelle. L'enjeu est de déterminer **comment** et **où** injecter $\mathcal{R}(t)$ dans la dynamique DSL, de sorte que l'**auto-organisation** ne se contente pas d'une simple affinité intrinsèque (S), mais tienne compte d'un **but** global ou local.

A. Définition d'un Signal de Récompense $\mathcal{R}(t)$

Il peut exister différentes **formes** de récompense dans un système :

Récompense globale : un **scalaire** unique, $\mathcal{R}(t) \in \mathbb{R}$, fourni à chaque itération t . Par exemple, dans une équipe de robots coopérant pour transporter un objet, $\mathcal{R}(t)$ peut être le **score** global (distance restante, temps écoulé, etc.).

Récompense partielle : un **vecteur** $\{\mathcal{R}_i(t)\}_i$, précisant la récompense de chaque entité ou de chaque sous-groupe. Dans un système multi-agent, chaque agent i reçoit sa **propre** contribution $\mathcal{R}_i(t)$. Ou, dans un SCN orienté vers la partition en clusters, chaque **cluster** se voit attribuer une **note** de performance.

Le **DSL** (Deep Synergy Learning) modifie normalement $\omega_{i,j}$ en fonction de $S(i,j)$ et d'un terme $-\tau \omega_{i,j}$. Pour intégrer la **récompense** $\mathcal{R}(t)$, on ajoute un **terme** qui renforce (ou diminue) $\omega_{i,j}$ selon la **valeur** de $\mathcal{R}(t)$. Formellement :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] + \Delta_{\text{reward}}(i,j,t),$$

avec

$$\Delta_{\text{reward}}(i, j, t) = f(\mathcal{R}(t), \omega_{i,j}(t), \dots).$$

Le **choix** de la fonction f dépend de la nature de la récompense : si celle-ci est globale, on peut répartir $\mathcal{R}(t)$ de manière “diffuse” ; si elle est partielle, on peut cibler plus directement les liaisons associées aux entités récompensées (p. ex. $\Delta_{\text{reward}}(i, j) \propto \mathcal{R}_i(t) + \mathcal{R}_j(t)$).

B. Formulation Mathématique : Couplage entre \mathcal{R} et ω

Une **approche** simple est de rendre Δ_{reward} **proportionnelle** à $\mathcal{R}(t)$. Par exemple :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha (\mathcal{R}(t)) \times g(i,j),$$

où $\alpha(\cdot)$ est un coefficient positif qui **croît** avec la valeur de \mathcal{R} . La **fonction** $g(i,j)$ indique dans quelle mesure le lien (i,j) est concerné par la récompense. Si \mathcal{R} est globale, on peut laisser $g(i,j) = 1$ pour tous ; si la récompense est partielle, on peut prendre $g(i,j) = 1/2 (\mathcal{R}_i(t) + \mathcal{R}_j(t))$ ou un autre schéma reflétant la *contribution* des entités i, j .

Si $\mathcal{R}(t)$ est purement **globale**, le SCN ne sait pas *directement* quels liens $\omega_{i,j}$ ont été “bénéfiques”. On effectue donc une mise à jour “**diffuse**” : toutes les liaisons en bénéficient, ou on en fait bénéficier un certain sous-ensemble jugé actif à l’itération t . Cela peut renforcer la **solidarité** globale, mais manque de précision. À l’inverse, une **récompense partielle** $\{\mathcal{R}_i(t)\}$ offre la possibilité de cibler plus **finement** les entités dont l’action (ou la synergie) a été profitable, en augmentant spécifiquement leurs pondérations.

C. Avantages de Fusionner DSL et Récompense

L’intégration d’un **feedback** $\mathcal{R}(t)$ permet d’**aligner** l’auto-organisation intrinsèque du SCN sur un **objectif fonctionnel**. Alors que la synergie $S(i,j)$ favorise des regroupements **spontanés** en fonction de similarités ou de complémentarités, l’ajout d’une récompense oriente le réseau vers un **but extrinsèque** (score, survie, mission robotique, etc.), rendant l’apprentissage plus ciblé.

Cette approche assure une **adaptation décentralisée**, où chaque liaison $\omega_{i,j}$ s’actualise en fonction des informations locales $(S(i,j), \omega_{i,j}, \mathcal{R}(t))$, évitant ainsi un besoin de planification centralisée.

En combinant **synergie et récompense**, la dynamique renforce les liens entre les entités qui possèdent **à la fois une bonne compatibilité et une conformité aux objectifs globaux**. Ce compromis peut toutefois provoquer des **oscillations**, lorsque $\mathcal{R}(t)$ entre en conflit avec $S(i,j)$, mais ces ajustements sont parfois nécessaires pour sortir d’un arrangement purement auto-organisé mais inefficace fonctionnellement.

D. Points de Vigilance

Une attention particulière doit être portée à la **corrélation entre \mathcal{R} et S** . Si la récompense favorise une structure **en désaccord** avec la synergie naturelle du réseau, des conflits d’optimisation peuvent émerger. Il est alors crucial de **paramétrier finement** η, α, τ , etc., afin d’éviter des **oscillations persistantes**.

Le **bruit ou les fluctuations** de $\mathcal{R}(t)$ constituent un autre défi. Une récompense instable ou fortement bruitée peut **ralentir la convergence** du SCN et prolonger les états transitoires, compliquant la stabilisation du réseau.

Enfin, dans un **cadre multi-agent**, la manière dont la récompense est attribuée influence fortement la dynamique du SCN. Selon qu’elle est distribuée de **façon compétitive ou coopérative** (cf. chap. 7.7.2), la sélection des connexions et la formation des clusters suivront des stratégies différentes, nécessitant un réglage précis des **critères de gain individuels ou collectifs**.

Conclusion

L’intégration d’un signal de **récompense** $\mathcal{R}(t)$ dans la dynamique DSL permet un **couplage** entre l’auto-organisation basée sur la **synergie** (S) et un **objectif** externe ou un **score** fonctionnel. Concrètement, on modifie la règle de mise à jour $\omega_{i,j}(t+1)$ en y ajoutant un **terme** dépendant de $\mathcal{R}(t)$, global ou partiel, qui **renforce** (ou affaiblit) certaines liaisons au gré des performances constatées. Cette **fusion** DSL + récompense donne à la **structure** du SCN une

dimension plus “intelligemment orientée”, tout en conservant la décentralisation et la plasticité auto-organisée qui caractérisent le DSL.

7.7.1.2. Définition : le SCN s’adapte non seulement en fonction de $S(i,j)$ mais aussi d’une “récompense” extrinsèque

Dans le **Deep Synergy Learning** (DSL), la dynamique des pondérations $\{\omega_{i,j}\}$ est généralement gouvernée par une **mesure de synergie** $S(i,j)$, laquelle reflète la *similarité*, la *compatibilité* ou la *co-information* entre deux entités i et j . Cependant, dans de nombreux **contextes** — en particulier ceux issus de l'**apprentissage par renforcement** ou de systèmes **multi-agents** — il est souhaitable de tenir compte d’un **signal de récompense** $\mathcal{R}(t)$, extérieur au calcul de synergie intrinsèque, afin d’**orienter** l’auto-organisation du Synergistic Connection Network (SCN) vers un **objectif** de performance ou de satisfaction plus global.

A. Notion de Récompense Extrinsèque

Dans le DSL, la **synergie** $S(i,j)$ provient de **données internes** au réseau (similitude vectorielle, co-occurrence, compatibilité symbolique, etc.). À l’inverse, la **récompense** $\mathcal{R}(t)$ provient d’un **feedback externe** à la simple mesure de synergie : un **score** ou un **gain** imposé par l’environnement (jeu, robotique, collaboration multi-agent) indiquant la **qualité** de la configuration ou de l’action courante.

Ce **signal extrinsèque** constitue un **levier** supplémentaire : il n’est plus seulement question de regrouper des entités “ressemblantes” ou “complémentaires”, mais de **maximiser** un critère *extérieur*. On rapproche ainsi le SCN d’une démarche **téléologique**, où la structure des liens se met au service d’une finalité (mission, performance, etc.).

On peut imaginer un **ensemble** d’agents RL ou de robots collaboratifs dans un certain environnement. À chaque itération, l’**environnement** renvoie une **récompense** $\mathcal{R}(t)$ reflétant la **qualité** de la coopération ou l’**efficacité** de la mission accomplie. Le SCN, au lieu de s’ajuster uniquement via $S(i,j)$, intègre alors un **terme** d’adaptation lié à $\mathcal{R}(t)$. Les pondérations $\omega_{i,j}$ se renforcent ou s’affaiblissent non seulement selon la similarité entre i et j , mais également selon la **pertinence** de leur coopération vis-à-vis de l’objectif global (ou local).

B. Mécanisme d’Adaptation avec Récompense

La règle de mise à jour du DSL, habituellement

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

peut être étendue en ajoutant un **terme** associant la **récompense** $\mathcal{R}(t)$. Par exemple :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \alpha \Delta_{\text{reward}}(i,j,t).$$

Le **terme** $\Delta_{\text{reward}}(i,j,t)$ dépend d’une fonction du signal $\mathcal{R}(t)$, éventuellement pondérée par la contribution (ou l’activité) des entités i et j . Par exemple, si la récompense est globale, on peut la distribuer de façon égale à tous les liens (ou seulement à ceux “actifs”). Si elle est locale, on peut l’appliquer **sélectivement** aux liaisons ayant réellement participé au gain.

Une **récompense globale** repose sur un **scalaire unique** $\mathcal{R}(t)$, appliqué à l’ensemble du SCN. Dans ce cas, chaque connexion $\omega_{i,j}$ peut recevoir un **bonus proportionnel** à $\mathcal{R}(t)$, suivant une logique où **tout le réseau est récompensé collectivement**, comme dans un **système coopératif multi-agent**.

À l’inverse, une **récompense partielle** utilise un **vecteur** $\{\mathcal{R}_i(t)\}$ ou même une **récompense spécifique à chaque lien** $\mathcal{R}_{i,j}(t)$, permettant d’évaluer la performance **locale** des connexions. Cela autorise une **mise à jour ciblée** des pondérations $\omega_{i,j}$, en tenant compte de l’efficacité spécifique de chaque interaction.

D’un point de vue **mathématique**, ces mécanismes introduisent une **force supplémentaire** qui oriente l’auto-organisation du SCN. L’intuition sous-jacente est la suivante :

La synergie interne $S(i,j)$ détermine le degré d'affinité naturelle entre i et j ,

Le signal extrinsèque \mathcal{R} module $\omega_{i,j}$, en l'augmentant ou en le réduisant selon que leur coopération contribue ou non à l'objectif global.

C. Intérêt : Convergence Vers un But Extrinsèque

Sans récompense, le SCN se contente de **clusteriser** les entités selon leur similitude ou complémentarité (voir chap. 2). Avec la récompense, on obtient une **direction** extrinsèque : certaines entités “profitables” au regard du critère \mathcal{R} finiront par se regrouper même si elles ne sont pas très “similaires” selon $S(i,j)$. Cela enrichit la logique d'**auto-organisation**.

Dans une configuration multi-agent coopérative (robots, jeux collectifs), la récompense $\mathcal{R}(t)$ reflète le succès de la **collaboration**. Le SCN apprend donc à renforcer les liens $\omega_{i,j}$ entre agents qui, mis ensemble, produisent un gain meilleur — c'est une **forme** d'apprentissage par renforcement distribué (voir § 7.7.2 pour un approfondissement).

Le couplage $S(i,j) + \mathcal{R}(t)$ nécessite de **paramétrier** les coefficients (ex. η, α, τ) pour éviter des oscillations si la récompense contredit la synergie. Au niveau **mathématique**, on peut interpréter cela comme l'ajout d'une composante d'énergie $J_{\mathcal{R}}$ à la fonction de coût, conduisant à un **potentiel** global $J_{\text{total}} = J_{\text{Synergy}} + J_{\mathcal{R}}$. Le SCN cherche à minimiser simultanément l'énergie “intrinsèque” (mesurant la cohésion interne) et l'énergie “extrinsèque” (liée à la récompense négative ou positive).

Conclusion

En intégrant un **signal de récompense** $\mathcal{R}(t)$ au sein du **SCN**, on modifie la mise à jour DSL de manière à **répondre** non seulement à la synergie interne $S(i,j)$ mais également à un **objectif** extérieur (score, performance, utilité). Sur le plan **mathématique**, cette combinaison introduit un **terme** lié à \mathcal{R} dans l'équation de la dynamique, conduisant à :

Un **compromis** : on maintient le **clustering** ou la **cohérence** basée sur la synergie,

On **orienté** les pondérations ω vers des **configurations** qui assurent une récompense extrinsèque plus élevée.

Ce principe étend la portée du DSL aux applications où la **simple** similarité (intrinsèque) ne suffit pas, et où l'on veut **optimiser** un critère externe, comme dans la collaboration multi-agent, la robotique, ou l'apprentissage par renforcement **distribué**.

7.7.2. Multi-Agents : DSL comme Réseau de Coopération

Le **DSL** (Deep Synergy Learning) ne se limite pas aux entités “passives” (ex. des données ou des vecteurs) : il peut s'appliquer à des **agents** dynamiques prenant des **décisions** et recevant des **retours** (récompenses ou signaux d'erreur). Dans un cadre **multi-agents**, chaque entité du SCN correspond à un **agent RL** (Reinforcement Learning) ; les liaisons $\omega_{i,j}$ entre agents i et j mesurent à quel point leurs **actions** se **synergisent** ou se **contrarient** mutuellement. Le SCN ainsi formé devient un **réseau** de coopération, où la **dynamique** DSL actualise la collaboration entre agents, tandis que le signal de renforcement influe sur les liens.

7.7.2.1. Chaque Agent RL Dispose de Liaisons $\omega_{i,j}$ avec d'Autres Agents, Modulées Selon la Synergie de Leurs Actions

Dans un **système multi-agent** utilisant l'**apprentissage par renforcement** (RL), il est fréquent de souhaiter **coordonner** un ensemble d'agents $\{\mathcal{A}_1, \dots, \mathcal{A}_n\}$. Le **Synergistic Connection Network** (SCN) offre une architecture **distribuée** et **adaptative** pour gérer la *coopération* ou la *coopétition* entre ces agents. L'idée est de **mapper** chaque agent RL à un **nœud** du SCN, et d'utiliser les pondérations $\{\omega_{i,j}\}$ pour évaluer et renforcer la **synergie** entre les actions de différents agents. Ainsi, chaque agent conserve sa **politique** de RL, tandis que le SCN régule les liaisons $\omega_{i,j}$ en fonction de la **compatibilité** observée dans leurs comportements, laissant émerger, au fil du temps, des **sous-groupes** d'agents étroitement coopératifs.

A. Principe : Entités \equiv Agents, Liaisons \equiv Coopération

Le **DSL** (Deep Synergy Learning) définit un **réseau** de nœuds, ici assimilés à des **agents RL**. Chaque liaison $\omega_{i,j}$ mesure la **coopération** (ou le degré de “cohésion”) entre l’agent i et l’agent j . Plus la **synergie** entre leurs actions se manifeste, plus $\omega_{i,j}$ s’élève ; si leurs actions se contredisent ou se révèlent incompatibles, $\omega_{i,j}$ diminue.

Sur le plan **algorithmique**, cela signifie qu’à chaque itération ou chaque “pas” temporel, où tous les agents choisissent leurs **actions** $\alpha_i^{(t)}$, le SCN observe et mesure la **synergie** $S_{i,j}(t)$ entre les choix de i et j . On met alors à jour la liaison $\omega_{i,j}$ selon une règle DSL, généralement :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{i,j}(t) - \tau \omega_{i,j}(t)],$$

en tenant compte éventuellement d’autres facteurs (inhibition, bruit, etc.). Les $\omega_{i,j}$ élevés induisent que les deux agents i et j sont fortement **coopératifs** ou que leurs actions se coordonnent dans l’environnement.

B. Modélisation Mathématique de la Synergie des Actions

Chaque **agent** \mathcal{A}_i est un RL-learner classique : il perçoit un **état** $o_i^{(t)}$ (observations locales ou globales), choisit une **action** $\alpha_i^{(t)}$, et reçoit éventuellement une **récompense** $R_i(t)$. La **synergie** $S_{i,j}(t)$ entre deux agents i et j peut être calculée à partir :

- D’une **compatibilité** ou d’un indice d’**alignement** entre $\alpha_i^{(t)}$ et $\alpha_j^{(t)}$. Par exemple, si les deux robots se positionnent de façon complémentaire pour soulever un objet, $S_{i,j}(t)$ est élevé ; s’ils se gênent ou se déplacent en contradiction, $S_{i,j}(t)$ est faible ou négative.
- D’un **feedback** plus complexe où l’environnement évalue la conjonction $(\alpha_i^{(t)}, \alpha_j^{(t)})$. Cela peut être une fonction $f(\alpha_i, \alpha_j, \dots)$ dépendant de la situation courante.

Au fil de l’itération temporelle $t \rightarrow t + 1$, on applique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{i,j}(t) - \tau \omega_{i,j}(t)] \quad (\text{forme de base}).$$

Ainsi, plus les actions des deux agents sont “complémentaires” ou “coopératives”, plus $\omega_{i,j}$ augmente, renforçant la probabilité de collaboration future. Inversement, si leurs actions se contredisent, $\omega_{i,j}$ baisse, reflétant un **décrochement** de la coopération.

Dans certaines variantes, on intègre un **terme** d’inhibition (cf. chap. 7.4), un **bruit** (chap. 7.3) ou un **feedback** extrinsèque (chap. 7.7.1) pour influencer la trajectoire des liaisons.

C. Émergence de “Teams” ou Sous-groupes Coopératifs

Le **DSL** aboutit souvent à la formation de **clusters** d’agents : un **groupe** $\mathcal{C}_\alpha \subseteq \{1, \dots, n\}$ partage des liaisons $\omega_{i,j}$ élevées, traduisant leur fort degré de coopération ou de complémentarité d’actions. Mathématiquement, un **team** se manifeste par la **somme** (ou la moyenne) interne $\sum_{i,j \in \mathcal{C}_\alpha} \omega_{i,j}$ étant notablement supérieure à la somme des liaisons inter-groupe.

Cette émergence de sous-groupes coopératifs se réalise **automatiquement** via la **dynamique** DSL, plutôt que par un protocole centralisé. Au fil des itérations, certaines paires (ou sous-ensembles) d’agents coopèrent souvent, ont des synergies positives, et donc **stabilisent** leurs liaisons ; d’autres demeurent plus isolés ou concurrents, aux liaisons plus faibles.

La particularité d’un **SCN** est qu’il est en changement **permanent** : les actions $\alpha_i^{(t)}$ étant susceptibles de **varier** (agents RL en train d’explorer), la **synergie** $S_{i,j}(t)$ se reconfigure en conséquence, engendrant éventuellement la **fragmentation** ou la **fusion** de teams à mesure que la tâche (ou la récompense) évolue.

D. Mérites et Limites de l'Approche

L'un des principaux atouts de cette approche est la **décision décentralisée**, où chaque agent suit sa propre **politique d'apprentissage par renforcement (RL)**. La coopération émerge naturellement via la mise à jour distribuée des liaisons $\omega_{i,j}$, éliminant ainsi le besoin d'un **coordinateur centralisé**.

L'**auto-organisation** constitue un autre avantage majeur : les équipes coopératives ne sont **pas prédefinies**, mais émergent directement des **observations** et **actions** des agents. C'est la **synergie** $S_{i,j}$ qui façonne progressivement la structure du réseau.

Enfin, cette approche offre une **grande plasticité**. Lorsqu'un changement de tâche ou d'environnement survient, la synergie $S_{i,j}$ est affectée en conséquence, et le **DSL** reconfigure naturellement le SCN. Cela permet aux agents auparavant isolés de former de nouvelles équipes si cela devient bénéfique.

Toutefois, cette approche présente également certaines **contraintes**. Le **coût computationnel** peut être élevé, notamment lorsque l'on doit maintenir $\omega_{i,j}$ pour un nombre quadratique de paires d'agents $O(n^2)$. Des stratégies de **parsimonie** (telles que la limitation aux k plus proches voisins ou l'inhibition) peuvent être nécessaires pour réduire cette complexité.

Un autre défi réside dans la **mesure de la synergie** $S_{i,j}(t)$. Il est essentiel de disposer d'un critère fiable pour évaluer la **compatibilité** entre actions des agents, ce qui peut être **non trivial** dans des scénarios **complexes** ou **partiellement observés**.

Enfin, la **stabilité du réseau** peut être mise à l'épreuve. Selon les valeurs des paramètres η et la dynamique de la synergie, certaines liaisons ω peuvent donner lieu à **oscillations** ou **conflits**, en particulier si des agents conservent des comportements **incohérents ou antagonistes**.

Conclusion

Dans un **multi-agent** (RL) contexte, la **cartographie** d'agents en **nœuds** et la **mise à jour** de liaisons $\omega_{i,j}$ via la **synergie** $S_{i,j}$ offrent une **démarche** puissante de **coopération** auto-organisée. La **dynamique** DSL incarne la force des liens entre agents en fonction de la compatibilité de leurs **actions**. Au fil du temps, des **teams** (sous-groupes d'agents) émergent et se dissolvent en fonction du bénéfice mutuel constaté, permettant une coordination **décentralisée** et **évolutive**, essentielle pour les systèmes de collaboration robotique, d'essaim de drones ou de multiples agents RL interagissant dans un environnement partagé.

7.7.2.2. Avantage : Auto-Formation de “Team” d’Agents, Désavantage : Complexité Combinatoire

Lorsqu'on applique un **Synergistic Connection Network** (SCN) à un **système multi-agents** en apprentissage par renforcement, chaque nœud du SCN correspond à un **agent** ($\mathcal{A}_1, \dots, \mathcal{A}_n$), et chaque liaison $\omega_{i,j}$ quantifie la **coopération** ou la **synergie** entre les agents i et j . Cette démarche confère au SCN la capacité de **former** et de **réorganiser** des **équipes** (ou teams) d'agents lorsqu'ils interagissent fortement. On bénéficie ainsi d'une **auto-organisation** de la coopération, sans imposer explicitement des sous-groupes prédefinis. Toutefois, cette approche s'accompagne d'un **défi** majeur : la **complexité combinatoire** liée à l'éventail potentiel de liaisons et de configurations d'agents.

A. Avantage : Auto-Formation de “Teams” d’Agents

Dans un tel réseau, chaque agent \mathcal{A}_i exécute sa **politique** de RL, observe un **état**, choisit une **action** α_i , et perçoit une **récompense** \mathcal{R}_i . En parallèle, le SCN calcule une **synergie** $S(\mathcal{A}_i, \mathcal{A}_j)$ à partir de la compatibilité ou de la conjonction des actions α_i, α_j . Si deux agents coopèrent ou complètent leurs rôles, la synergie est élevée. Si leurs actions interfèrent ou s'opposent, la synergie est faible, voire négative.

La **dynamique** du DSL met alors à jour les pondérations $\omega_{i,j}$ de façon proportionnelle à la synergie $S(\mathcal{A}_i, \mathcal{A}_j)$. Les agents qui s'avèrent utiles ensemble (actions concertées) voient leur liaison se renforcer, reflétant la formation d'un **lien coopératif**.

Au fil des itérations, on constate l'**auto-formation** de "teams" (ou **sous-groupes**) d'agents. Les agents à forte synergie mutuelle $\{\omega_{i,j}\}$ se regroupent naturellement en un cluster. Sur le plan **mathématique**, on caractérise ce cluster \mathcal{C} par une somme interne $\sum_{i,j \in \mathcal{C}} \omega_{i,j}$ élevée et des liaisons plus modestes vers les agents hors de \mathcal{C} . Ainsi, la **coopération** fait naître un bloc fortement interconnecté, sans qu'on ait eu besoin de spécifier a priori une partition.

Ces équipes auto-formées offrent un **gain** opérationnel : les agents dans un même team partagent de l'information et ajustent leurs politiques RL pour **maximiser** leur synergie ou un objectif commun. En d'autres termes, ils accroissent la **cohésion** interne, bénéficiant d'une **intelligence collective** supérieure à la simple somme des agents isolés. Le DSL actualise constamment les liaisons $\{\omega_{i,j}\}$, permettant de se réorganiser si la tâche ou la récompense évolue, et donc de **garder** un système multi-agent adaptatif et collaboratif.

Lorsque l'environnement ou la **mission** change, la **synergie** $S(\mathcal{A}_i, \mathcal{A}_j)$ est susceptible de varier. Le SCN reflète cette variation en réajustant $\omega_{i,j}$. Certains sous-groupes peuvent se **dissoudre**, d'autres émerger. Cette **plasticité** est un atout considérable : on laisse les agents **s'auto-répartir** en teams selon les **exigences** courantes, sans imposer de coordination centralisée.

B. Désavantage : Complexité Combinatoire

Si l'**auto-formation** de teams constitue une force du SCN, elle implique une **croissance** combinatoire des connexions et de leurs évolutions, particulièrement lorsque le **nombre** d'agents n s'élève.

Avec n agents, le **réseau** potentiel comprend $O(n^2)$ pondérations $\omega_{i,j}$. La **dynamique** DSL exige de mettre à jour chacune de ces liaisons, ou d'en vérifier la **synergie** associée, à chaque itération. Dans un cadre multi-agent, si n devient très grand, ce **coût** (en calcul et en mémoire) peut devenir prohibitif. Pour chaque pas de temps, on doit gérer l'évolution d'un volume important de **liens**, ce qui est peu scalable.

La "structure" de partition en sous-groupes coopératifs a elle-même une **combinatoire** exponentielle : le SCN peut potentiellement explorer un très grand nombre de partitions ou de regroupements (forme de "clustering" multi-agent). Sans mécanismes de **sparsité** (inhibition, coupes) ou de "recuit" pour sortir des minima locaux, le réseau risque de peiner à dénicher des solutions vraiment optimales dans la multiplicité de partitions possibles.

Plusieurs moyens peuvent restreindre cette **combinatoire** :

551. **k-NN ou seuil** : on limite chaque agent \mathcal{A}_i à k voisins privilégiés. On ne maintient $\omega_{i,j}$ que pour ce voisinage.

552. **Inhibition** : on accentue la compétition entre liaisons, poussant le SCN à "choisir" seulement quelques connexions par agent, rendant la topologie plus éparsé (cf. chap. 7.4).

553. **Recuit stochastique** ou "shake" : on injecte un **bruit** ou un **reset** partiel pour éviter la stagnation et accélérer la découverte d'autres partitions (chap. 7.3, 7.6.3).

Conclusion

Le **couplage** d'un SCN (via la synergie $\omega_{i,j}$) avec des **agents** RL offre un **avantage** clé : l'**auto-formation** de "teams" d'agents coopératifs, permettant une **auto-organisation** des sous-groupes qui ont intérêt à travailler ensemble. Cette dynamique se révèle puissante, **sans** qu'une autorité centrale doive décider a priori de la répartition. Toutefois, cette approche se heurte à un **désavantage** : la **complexité combinatoire** (mise à jour de $O(n^2)$ liaisons et exploration de configurations multiples). Les mécanismes de **sparsification** (k plus proches voisins, inhibition) ou de **recuit** s'avèrent alors indispensables pour maintenir un **coût** gérable et pour éviter qu'un trop grand nombre de connexions empêche la **convergence** dans un délai raisonnable.

7.7.3. Exemples de Collaboration RL–DSL

Dans le contexte d'une **collaboration** entre l'**apprentissage par renforcement** (RL) et la **dynamique** DSL, un cas particulièrement illustratif est celui d'un **essaim robotique** multi-agent. Chaque robot prend ses **décisions** (actions) via un schéma de **reinforcement learning**, tandis que le **Synergistic Connection Network** (SCN) gère l'**organisation** coopérative entre robots (pondérations ω , formation de clusters d'agents, etc.). Nous détaillons ici (7.7.3.1) un scénario type : un essaim de robots coordonne ses actions collectives (navigation, répartition des tâches) grâce à un **couplage** RL–DSL.

7.7.3.1. Cas d'un Essaim Robotique Multi-Agent : RL pour la Sélection d'Actions, DSL pour la Structure Coopérative

Un **essaim** robotique multi-agent rassemble plusieurs **robots** (ou agents) évoluant dans un même environnement et poursuivant un objectif collectif (ex. exploration, transport coordonné, surveillance). Chacun de ces robots suit une **politique** d'apprentissage par **renforcement** (RL) pour décider de ses **actions** (se déplacer, saisir un objet, communiquer, etc.). En parallèle, le **Synergistic Connection Network** (SCN) — propre au **DSL** (Deep Synergy Learning) — maintient des **liaisons** $\omega_{m,n}$ reflétant la *coopération* ou la *compatibilité* entre deux robots Robot_m et Robot_n . Le rôle de ce réseau consiste à **auto-organiser** les sous-groupes de robots (teams) qui se révèlent *synergiques*, sans imposer de partition rigide.

A. Organisation Générale : RL par Robot et SCN Global

Les deux **dynamiques** s'entrelacent pour structurer l'apprentissage et la coopération entre robots.

Le **RL local** est propre à chaque robot Robot_m , qui dispose d'une **politique** π_m ou d'une **Q-fonction** Q_m . À partir d'un **état local** $\text{state}_m(t)$, il choisit une **action** $\alpha_m(t)$ et reçoit une **récompense** $\mathcal{R}_m(t)$, qu'elle soit globale ou locale. La mise à jour de la Q-fonction suit un algorithme de type Q-learning ou équivalent.

Parallèlement, le **SCN global** modélise les robots comme des **nœuds** interconnectés par des liaisons $\omega_{m,n}$ représentant leur **coopération**. Ces liaisons évoluent selon la règle DSL (ou une variante), influencées par la **similarité des actions** α_m, α_n , la **complémentarité des rôles**, ainsi que la **réussite observée** lorsqu'ils agissent de manière coordonnée.

Cette **double structure** (apprentissage par renforcement pour chaque robot + SCN global adaptatif) permet d'associer un **apprentissage individuel** à une **coopération émergente**, construisant ainsi une architecture **flexible et auto-adaptative**.

B. Mécanismes de RL Local

Chaque robot suit un **apprentissage par renforcement** classique, organisé en plusieurs étapes :

État $\text{state}_m(t)$: données issues des capteurs, position, informations partagées.

Action $\alpha_m(t)$: déplacement, saisie, transmission, etc.

Récompense $\mathcal{R}_m(t)$: déterminée de manière individuelle ou partagée globalement dans un cadre multi-agent coopératif.

Mise à jour : ajustement de la Q-fonction selon une règle de type Q-learning, SARSA ou DDPG, par exemple :

$$Q_m(\text{state}_m, \alpha_m) \leftarrow Q_m(\text{state}_m, \alpha_m) + \alpha \left[\mathcal{R}_m(t) + \gamma \max_{\alpha'} Q_m(\text{state}'_m, \alpha') - Q_m(\text{state}_m, \alpha_m) \right].$$

En parallèle, le **SCN collecte des informations** sur les interactions entre robots afin d'ajuster dynamiquement les **liaisons** $\omega_{m,n}$, renforçant ainsi la synergie et la coordination des actions.

C. Couplage avec la Synergie $\omega_{m,n}$

Pour chaque pas de temps t , les actions $\alpha_m(t)$ et $\alpha_n(t)$ prises par deux robots peuvent être **coopératives**, **complémentaires** ou **redondantes**. On définit alors un score $S_{m,n}(t)$ reflétant la “qualité” de la conjonction (α_m, α_n) . Cela peut inclure :

- **Compatibilité** de leurs positions (ont-ils évité une collision ? se sont-ils rejoints pour porter un objet ?).
- **Contribution** à l’objectif global (si agir ensemble augmente la récompense collective).
- **Corrélation** entre leurs retours $\mathcal{R}_m(t)$ et $\mathcal{R}_n(t)$.

Un **SCN** va ensuite mettre à jour la pondération $\omega_{m,n}(t)$ suivant la **règle** DSL de base :

$$\omega_{m,n}(t+1) = \omega_{m,n}(t) + \eta [S_{m,n}(t) - \tau \omega_{m,n}(t)],$$

éventuellement complétée par un **terme** d’inhibition, de bruit ou de récompense extrinsèque (cf. chap. 7.4, 7.7.1). Si deux robots coopèrent souvent, $\omega_{m,n}$ monte ; s’ils interfèrent, $\omega_{m,n}$ décroît.

Après plusieurs itérations, des **clusters** d’agents se forment là où $\omega_{m,n}$ s’avère élevé. Cela désigne des **sous-groupes** (teams) qui ont découvert une bonne synergie d’actions. Les robots peuvent ainsi se spécialiser ou coordonner leurs choix (par ex. un ensemble gère la zone A, un autre la zone B). Et si la tâche change, la synergie $S_{m,n}(t)$ change, entraînant un **re-mapping** des liaisons $\omega_{m,n}$.

Cette formation de sous-groupes n’est pas imposée, mais **émerge** de la dynamique DSL couplée aux décisions individuelles (RL) de chaque robot.

D. Création Spontanée de Sous-Groupes Coopératifs

Au fur et à mesure que les **robots** expérimentent des stratégies de plus en plus abouties (grâce à leur RL local), on observe que :

Les **couples** (m, n) réalisant une **coopération** (par ex. échanger des messages, porter des objets en tandem) déclenchent un **gain** plus élevé, se traduisant par une synergie $S_{m,n}$ récurrente.

Le **SCN** augmente $\omega_{m,n}$, renforçant un **canal** de coopération, ce qui facilite et accélère des interactions futures entre m et n .

Un vrai **sous-groupe** $\mathcal{C}_\alpha \subseteq \{1, \dots, M\}$ émerge si chacun y trouve un avantage de collaboration. Les connexions inter- \mathcal{C}_α s’enrichissent, faisant de ce cluster un “team” effectif.

D’un point de vue **opérationnel**, c’est un équilibre entre la **liberté** de chaque robot (sa politique RL) et la **dynamique** collective (pondérations ω) qui oriente les comportements cooperatifs.

E. Avantages de l’Approche Couplée RL–DSL

Plutôt que de forcer tous les robots à suivre une **stratégie** collective unique (ex. planificateur central), on laisse chaque robot **apprendre** (RL local) et le **SCN** se charger d’articuler la coopération via $\{\omega_{m,n}\}$. Les “teams” formés sont donc **auto-déterminés** : la “somme” des choix individuels sculpte la synergie, et réciproquement, la synergie influe les opportunités de collaboration.

Si l’environnement ou la **mission** se modifie, les agents ajustent leurs **actions** (via RL). Par suite, la **synergie** $S_{m,n}$ se réajuste, et donc $\omega_{m,n}$ s’adapte, pouvant **dissoudre** un cluster obsolète et **favoriser** de nouvelles coopérations. C’est une **plasticité** perpétuelle, essentielle en robotique ou dans des contextes évolutifs.

Dans une taille **modeste** d’essaim (dizaines, centaines de robots), on peut entretenir $O(M^2)$ liaisons et utiliser la mise à jour DSL. Pour un très **grand** nombre de robots, il faudra limiter la connectivité par des stratégies de **voisinage** (k plus proches, inhibition, etc.), afin d’éviter un coût $O(M^2)$ trop élevé.

Conclusion

Dans un **essaim robotique** multi-agent, le **DSL** et le **RL** se combinent pour offrir à la fois :

Apprentissage local pour chaque robot (sélection d'actions via RL, feedback de récompense individuel ou global),

Formation adaptative des liaisons $\omega_{m,n}$ (synergie), permettant l'émergence de **teams** coopératifs.

Cette **intégration** RL–DSL assure une **coordination** souple et distribuée au sein de l'essaim, où chaque robot *continue* d'apprendre sa propre politique, tandis que le SCN “sculpte” la structure des interactions selon la **compatibilité** des actions ou la performance collective. C'est une **approche** particulièrement adaptée à des environnements dynamiques, où la **reconfiguration** fréquente des équipes s'avère déterminante pour accomplir efficacement diverses missions.

7.7.3.2. Observations sur la Convergence, sur la Robustesse

Lorsqu'on intègre une **dynamique DSL** (Deep Synergy Learning) avec un système d'**apprentissage par renforcement** (RL) en mode multi-agent, il importe d'étudier la **convergence** jointe (1) de la **matrice ω** propre au SCN, et (2) de la **politique** (ou Q-fonction) des agents RL, tout en évaluant la **robustesse** de ce couplage face aux perturbations. De fait, on observe des phénomènes d'interdépendance : les **pondérations** $\{\omega_{i,j}\}$ influencent la **structure** de coopération entre agents, tandis que les **agents** façonnent (par leurs actions) la **synergie** qui pilote ω . Les effets de **bruit**, de “shake” stochastique, ou de changement d'objectif posent la question de la **stabilité** et du **caractère** éventuellement multi-attracteur de la dynamique.

A. Convergence Combinée de ω et de la Politique RL

La coévolution de (1) la **politique** $\{\pi_i\}$ (ou $\{Q_i\}$) de chaque agent RL et (2) la **matrice ω** du SCN peut être décrite par un **système** d'équations :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}(\omega_{i,j}(t), \{\alpha_i, \alpha_j\}, \mathcal{R}),$$

$$\pi_i(t+1) = \pi_i(t) + \Delta_{\text{RL}}(\pi_i(t), \text{Reward}_i, \omega_{i,\cdot}).$$

Cette **double boucle** signifie que la **synergie** $\omega_{i,j}$ se met à jour en fonction des actions α_i, α_j et du feedback coopératif, tandis que la **politique** RL évolue selon la récompense reçue (et potentiellement l'information issue du SCN). On obtient ainsi un **processus** où l'un influence l'autre, pouvant converger vers un **point fixe** (ω^*, π^*).

Dans le meilleur des cas, la dynamique aboutit à un état **stable** où :

Les **agents** ont affiné leurs politiques, menant à un certain niveau de **performance** (score, récompense globale) ;

Les **pondérations** $\{\omega_{i,j}\}$ se sont stabilisées sur des valeurs reflétant la “bonne” configuration de sous-groupes et de coopérations.

Cette convergence peut prendre un **certain temps** (ou un certain nombre d'itérations) et dépend de **paramètres** (taux d'apprentissage η , α , amplitude du bruit, etc.). On peut quantifier la vitesse de convergence via des mesures comme $\|\omega(t+1) - \omega(t)\|$ et $\|\pi(t+1) - \pi(t)\|$. Si elles s'amenuisent à zéro, on conclut à la **stabilisation**.

Si le **problème** (environnement multi-agent) s'avère **non convexe**, il peut exister **plusieurs** configurations ω & π localement stables (des attracteurs). Le couplage RL–DSL permet de **naviguer** entre ces configurations, mais peut aussi se trouver **piégé** dans un minimum local. Des mécanismes d'**injection stochastique** (chap. 7.6.3.1) ou de **vigilance** (chap. 7.6.3.2) peuvent inciter le système à s'**extraire** d'un attracteur inadéquat, accélérant (ou au contraire retardant) la **convergence**.

B. Robustesse du Couplage : Perturbations, Bruit, Changements de Politique

La **robustesse** recouvre la capacité du **système** (SCN + RL) à **maintenir** (ou retrouver) un état stable et performant malgré des chocs, modifications de reward, ou variations aléatoires.

Dans un environnement **réel**, les **agents** reçoivent des signaux bruités (capteurs incertains, latences de communication) ; le calcul de $\omega_{i,j}$ est donc lui aussi bruité. La **question** est : la dynamique parviendra-t-elle encore à s'auto-organiser ?

Empiriquement, on observe qu'un **DSL** combiné à un RL modéré peut tolérer un certain niveau de **bruit**. Les liaisons $\{\omega_{i,j}\}$ se stabilisent tant que la variance n'est pas trop forte et que l'update η reste assez petit. Sur le plan **mathématique**, il s'agit d'une **analyse stochastique** de la convergence, prouvant qu'avec des hypothèses de borne sur le bruit, ω et π convergent presque sûrement ou en probabilité.

Dans un multi-agent, chaque agent \mathcal{A}_i peut **faire évoluer** sa politique π_i . Cela modifie les synergies $\{S_{i,j}\}$ et donc perturbe la **matrice** ω . Par exemple, si un agent adopte soudainement une stratégie plus agressive, les liaisons $\omega_{i,j}$ reliant cet agent à d'autres pourraient s'affaiblir. La robustesse du couplage renvoie à la capacité du **SCN** à absorber ou s'ajuster à ces changements, sans s'effondrer.

Idéalement, même si la politique π_i d'un agent se modifie, la dynamique DSL finit par trouver un **équilibre adaptatif**, realignant $\omega_{i,j}$. Cette plasticité est un atout (le SCN "suit" l'évolution), mais peut aussi conduire à des **oscillations** si la fréquence des changements est trop rapide ou si η est trop grand. D'où la nécessité de **paramétrage** soigné.

C. Indicateurs Pratiques de Convergence et Robustesse

L'évaluation du **SCN** repose sur plusieurs indicateurs clés :

- **Convergence :**
 - La **norme** $\| \omega(t+1) - \omega(t) \|$, qui mesure la stabilisation des pondérations.
 - La **stabilité des clusters**, évaluée par le pourcentage de noeuds restant dans un même groupe au fil du temps.
 - La **performance globale du RL**, où la récompense moyenne atteint un plateau avec des fluctuations minimales.
- **Robustesse :**
 - L'introduction d'**événements perturbateurs** (changement d'objectif, panne d'un agent) pour observer si le SCN est capable de reformer ses liaisons coopératives.
 - L'injection d'un **bruit stochastique** dans le calcul de $S_{i,j}$ ou $\omega_{i,j}$ afin d'évaluer si le SCN parvient à converger malgré cette perturbation.
- **Temps de Reconstruction :**
 - Après un changement de récompense, le nombre d'**itérations nécessaires** pour que la performance RL remonte et que ω reconfigure une structure coopérative adaptée.

Conclusion

Quand un **essaim** multi-agent (ou un système collaboratif) combine **RL** (pour décider des actions) et **DSL** (pour gérer la **structure** coopérative ω), l'analyse **mathématique** et les **observations empiriques** montrent :

Convergence : Sous certaines conditions (paramètres de taux d'apprentissage modérés, récompenses stables), on voit la **matrice** $\{\omega_{i,j}\}$ se stabiliser en phase avec la **politique RL**, menant à un arrangement coopératif cohérent.

Robustesse : Le **couplage** RL–DSL peut absorber un **niveau** raisonnable de bruit ou de perturbations (changement de stratégie d'un agent, variation de l'environnement), le SCN se réadapte graduellement.

Risques : En cas de trop nombreuses modifications simultanées ou de bruit trop élevé, la **dynamique** peut osciller, exiger des heuristiques complémentaires (recuit, vigilance, etc.) pour stabiliser.

Ainsi, la **co-évolution** d'un SCN (pilotant la coopération) et d'un RL multi-agent (pilotant les **décisions** d'action) apparaît **viable** et capable de **former** des structures robustes à long terme, pourvu qu'on gère adéquatement les échelles de temps, la paramétrisation η, τ, α et les **phases** de mise à jour.

7.7.3.3. Pistes Futures : Continuer l'Exploration dans des Environnements Plus Riches

Après avoir mis en évidence (voir § 7.7.3.1, 7.7.3.2) la façon dont un **Synergistic Connection Network** (SCN) et l'**apprentissage par renforcement** (RL) peuvent collaborer au sein d'un essaim multi-agent ou dans un cadre robotique, de nombreuses **perspectives** s'ouvrent pour étendre cette intégration à des environnements plus vastes, partiellement observables, plus complexes et plus évolutifs. Les scénarios explorés étaient souvent de taille moyenne ou reposaient sur des hypothèses simplificatrices. Il reste un champ d'investigation large pour approfondir la **synergie** entre RL et DSL dans des systèmes encore plus ambitieux.

A. Complexification des Dynamiques RL–DSL

Dans une version plus avancée, chaque agent (ou robot) de l'essaim pourrait disposer d'un **embedding** vectoriel décrivant son état interne, mis à jour avec un **gradient** policy RL ou un mécanisme de type auto-encodeur, tandis que la **dynamique** DSL calcule la **synergie** $\omega_{i,j}$ à partir de ces embeddings.

Sur le plan **mathématique**, la synergie ne serait plus un simple $S(\mathcal{A}_i, \mathcal{A}_j)$ de compatibilité fixe, mais dépendrait de **vecteurs latents** $\mathbf{z}_i, \mathbf{z}_j$ évoluant au fil de l'entraînement. Cette idée ramène à un couplage plus **profond** entre la représentation interne de l'agent RL et la structure ω régissant la coopération.

On peut également imaginer des **couches** successives (voir chap. 6) : un cluster local d'agents coopère sur une sous-tâche (grâce à un SCN de bas niveau), puis plusieurs clusters coordonnent leurs plans via un super-nœud “macro-SCN”. L'apprentissage par renforcement opère donc à plusieurs **échelles** : RL local (actions de chaque agent), RL macro (gestion des groupes). La synergie ω se déploie ainsi sur plusieurs niveaux, conduisant à une architecture “multi-SCN”.

B. Environnements Dynamiques et Stochastiques plus Complexes

Dans des environnements **évolutifs**, la **récompense** peut changer (nouveaux objectifs), tout comme la *configurations* ou *localisation* des agents. Le SCN doit alors *reconfigurer* la matrice ω de sorte à refléter la **nouvelle** synergie attendue. Il faudra des mécanismes de “vieillissement” ou de réinitialisation de liens (cf. chap. 7.6) pour éviter la fossilisation des anciennes coopérations quand la réalité a changé.

Dans un environnement **partiellement observable**, chaque agent ne dispose que d'une **fenêtre** partielle sur l'état global. La **coopération** (favorisée par la synergie $\omega_{i,j}$) devient un outil crucial pour **partager** ou **mélanger** les observations. On peut alors imaginer que l'intensité $\omega_{i,j}$ facilite un échange d'information local entre agents i et j , leur permettant de combler leurs “angles morts”. Cela nécessite de repenser la fonction $S(\mathcal{A}_i, \mathcal{A}_j)$ pour y intégrer la **complémentarité** des observations ou l'utilité du partage. De plus, la dynamique DSL peut se révéler un **moyen** de construire *spontanément* de petits “réseaux” d'agents qui, ensemble, perçoivent mieux la situation.

C. Alignement sur des Objectifs Plus Riches

Au lieu d'une récompense **globale** unique (ou locale par agent), il peut exister un **arbre** d'objectifs : des sous-goals, des objectifs intermédiaires... Le **SCN** et la **dynamique** DSL pourraient alors gérer plusieurs **types** de synergie (compatibilité pour sous-goal A, pour sous-goal B), ou combiner divers signaux de récompense dans Δ_{reward} . Les “teams” se formeraient éventuellement autour de chaque sous-tâche, menant à une structuration multi-objectifs.

La formation de liens $\{\omega_{i,j}\}$ elle-même peut requérir un certain **degré** d’exploration : ne pas se contenter de renforcer toujours les liaisons déjà existantes, mais tester la synergie avec d’autres agents. Cela peut s’apparenter à un “Recuit stochastique” multi-agent (cf. chap. 7.3, 7.6.3). On peut injecter du **bruit** ou des “secousses” dans ω pour sortir d’une configuration trop restreinte. Sur un plan **mathématique**, c’est l’introduction d’un “shake” aléatoire ou d’un “vigilance reset” ponctuel (chap. 7.6.3) assurant qu’on n’abandonne pas la possibilité de nouvelles coopérations.

D. Approches Mathématiques Avancées

Le **couplage RL–DSL** peut se reformuler comme un **système** de grande dimension $(\mathbf{x}, \boldsymbol{\omega})$ évoluant par un ensemble d’équations non linéaires. On pourrait en faire une étude de **stabilité**, de **bifurcation**, ou de **contrôle** optimal, visant à caractériser les attracteurs, la vitesse de convergence, la résilience aux perturbations, etc. Des pistes incluent :

Analyser la **métrique** de distance $\| \boldsymbol{\omega}(t+1) - \boldsymbol{\omega}(t) \|$ conjointe à la **distance** entre politiques $\| \pi(t+1) - \pi(t) \|$.

Étendre cette étude à des variables latentes si chaque agent dispose d’embeddings mis à jour par un algorithme de type backprop.

Au lieu de laisser le couplage RL–DSL évoluer passivement, on peut imaginer un **contrôle** plus global qui insère un **coefficent** α devant Δ_{reward} . Ou un planificateur “macro” se sert du DSL pour redistribuer les ressources, dans un souci d'**optimisation**. Ces approches combinent les notions de “descente locale auto-organisée” (DSL) et de “régulation globale” par un planificateur (une forme de commande hiérarchique). Le **recuit** (chap. 7.3) ou la **vigilance** (7.6.3) en sont des exemples embryonnaires.

Conclusion

Les premières implémentations de DSL + RL (chap. 7.7.3.1, 7.7.3.2) ont montré des **résultats** prometteurs en environnements “moyennement” complexes (essaim robotique, missions de coordination). Les **pistes futures** indiquent comment pousser plus loin cette collaboration :

- **Environnements** plus riches (partiellement observables, dynamiques évolutifs),
- **Logiques** de synergie plus sophistiquées (embeddings d’agents, hiérarchies de clusters),
- **Systèmes** mathématiques avancés (contrôle optimal, recuit multi-niveau, adaptation incrémentale).

Ainsi, on étendra la **robustesse** et la **portée** du DSL dans des scénarios de taille **grande** ou de nature **changeante**, valorisant la **plasticité** et la **coopération** multi-agent en permanence. De futures recherches exploreront les protocoles d’échanges plus fins (comment ω sert à partager l’information sensorielle ?), l’alternance entre exploration et exploitation dans la **formation** des liaisons, et la gestion des **objectifs** complexes ou hiérarchiques, confirmant la **puissance** de la co-évolution RL–DSL dans des environnements industriels, robotiques, ou de simulation à échelle massive.

7.8. Comparaison Expérimentale et Paramétrages

Après avoir introduit les principes (recuit, inhibition, etc.) permettant de **peaufiner** ou **stabiliser** la dynamique du SCN (Synergistic Connection Network), il est naturel de s'interroger sur leur **efficacité pratique**. La présente section (7.8) aborde la question de la **comparaison expérimentale** et des **paramétrages** : comment, dans un cadre concret, peut-on évaluer la performance et comparer différentes variantes (sans recuit, avec recuit, inhibition, etc.) ? Quelles métriques de performance (énergie, cohésion, temps de convergence) emploie-t-on ? En quoi le choix de η , τ , γ influence-t-il sur le résultat ?

7.8.1. Étude Comparative (Sans Recuit, Avec Recuit, etc.)

Les méthodes exposées dans les chapitres précédents (règle DSL basique, recuit simulé, inhibition variable...) méritent une **évaluation** sur des scénarios tests. La sous-section 7.8.1 se concentre sur un **cas** relativement petit — 30 entités — mais contenant déjà **2 minima locaux** identifiés, ce qui permet de vérifier si la méthode (recuit, par exemple) parvient à “s’extraire” du premier minimum pour aller vers la configuration plus avantageuse.

7.8.1.1. Scénario Test : 30 Entités, 2 Minima Locaux Connus

Un **exemple** classique pour étudier la capacité d'un SCN (Synergistic Connection Network) à **échapper** à des minima locaux (ou à les franchir) consiste à configurer un **problème** artificiel avec un **nombre** modéré d'entités (ici, $n = 30$) et une **synergie** $S(i, j)$ construite de manière à présenter **deux** configurations stables $\Omega^{(1)}$ et $\Omega^{(2)}$, dont l'une est un **minimum local** moins performant et l'autre un **minimum** plus global ou de plus **faible énergie**. Cela permet de tester :

- La **tendance** du SCN à rester bloqué dans $\Omega^{(1)}$ ou à accéder à $\Omega^{(2)}$ selon l'initialisation,
- L'influence du **recuit simulé** (terme stochastique) ou d'autres heuristiques (inhibition variable) sur la probabilité de sortir de $\Omega^{(1)}$ pour atteindre $\Omega^{(2)}$.

A. Configuration de Départ

Dans cette étude, on considère un **réseau synergétique** comprenant $n = 30$ entités, noté $\{\mathcal{E}_1, \dots, \mathcal{E}_{30}\}$. Le système est construit autour d'une **matrice de pondérations** ω évoluant sous l'effet d'une **dynamique DSL** avec des contraintes d'auto-organisation. Afin d'analyser les comportements de convergence et d'optimisation, deux configurations distinctes sont introduites : la première, notée $\Omega^{(1)}$, représente un état localement stable, dans lequel une fois que certaines **liaisons** $\omega_{i,j}$ ont été établies, la dynamique naturelle du SCN tend à y rester confinée. En revanche, la seconde configuration, notée $\Omega^{(2)}$, correspond à un état de plus **faible énergie synergétique** \mathcal{J} , c'est-à-dire qu'elle offre un **meilleur équilibre global des liaisons**, mais reste difficilement atteignable en raison d'une **barrière énergétique** séparant les deux configurations.

On pourra coder $S(i, j)$ en donnant des valeurs “compatibles” avec $\Omega^{(1)}$ pour en faire un attracteur local, tout en rendant $\Omega^{(2)}$ de plus **haute synergie** globale..

B. Initialisation et Exécution

On démarre le SCN d'une **matrice** $\omega(0)$ initiale, souvent :

- **Aléatoire**, dans un range faible (p. ex. $[0, 0.01]$), ou
- **Uniformément nulle** (ou proche de zéro) pour tous les liens.

Selon l'**influence** de la distribution initiale, la descente DSL se dirigera plus ou moins vite dans le “bassin” de $\Omega^{(1)}$ ou $\Omega^{(2)}$. On multiplie donc les runs aléatoires (p. ex. 50 ou 100 exécutions) pour **estimer** la proportion aboutissant à chaque minimum local.

Pour mettre en valeur l'**impact** du recuit simulé (chap. 7.3), on compare :

Version DSL classique : Pas d'injection de bruit. On attend que la descente $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ se stabilise en un attracteur.

Version DSL + Recuit : On ajoute un terme stochastique $\xi_{i,j}(t)$ modulé par une température $T(t)$. On démarre avec un bruit fort (permettant de franchir la barrière entre $\Omega^{(1)}$ et $\Omega^{(2)}$ si c'est profitable), puis on diminue $T(t)$ au fil des itérations.

En pratique, on examine si le recuit **réduit** la probabilité de rester dans le “mauvais” minimum $\Omega^{(1)}$.

En plus du recuit, on peut introduire ou non une **inhibition** latérale (chap. 7.4) contrôlée par un paramètre γ . Un schéma typique consiste à rendre γ **adaptatif** : si on détecte que le réseau forme trop de liaisons en moyenne, on accroît γ (plus de compétition). Cette inhibition variable peut briser les clusters trop tôt formés ($\Omega^{(1)}$) et autoriser la reconfiguration vers $\Omega^{(2)}$.

C. Métriques de Comparaison

Pour distinguer les **performances** de chaque version, on définit des indicateurs :

Taux de Convergence vers $\Omega^{(2)}$. Sur un ensemble de runs aléatoires, on compte la **proportion** aboutissant finalement à l'état $\Omega^{(2)}$.

Énergie $J(\Omega)$ finale : on compare la valeur finale $J(\Omega(t_{\text{final}}))$. Plus cette énergie est faible, plus on est **proche** d'un état globalement optimal.

Cohésion de Clusters : calcul de modularité, ratio interne/externe, ou d'autres indices vérifiant si la partition finale correspond bien à $\Omega^{(2)}$.

L'idée est de **voir** si le recuit ou l'inhibition variable **augmente** la probabilité de quitter le puits local $\Omega^{(1)}$ et de rejoindre $\Omega^{(2)}$.

D. Attendus et Observations

L'analyse des résultats obtenus permet de formuler plusieurs hypothèses sur le **comportement du SCN**. En l'absence de recuit et d'inhibition, la descente DSL tend naturellement à **se figer dans** $\Omega^{(1)}$, rendant l'atteinte de $\Omega^{(2)}$ peu probable, sauf si la configuration initiale $\omega(0)$ est exceptionnellement bien orientée.

L'introduction d'un **recuit simulé** apporte une amélioration significative, car il permet aux pondérations $\omega_{i,j}$ de **franchir temporairement** les barrières de transition entre les deux configurations. Lorsque la température $T(t)$ est bien paramétrée, les runs convergent plus souvent vers $\Omega^{(2)}$, indiquant une meilleure exploration de l'espace des solutions.

L'ajout d'une **inhibition adaptative** introduit un effet complémentaire. En ajustant dynamiquement γ , la formation prématuée de $\Omega^{(1)}$ est partiellement empêchée, laissant le temps aux **connexions de se restructurer** avant de figer une organisation définitive. Cette stratégie permet donc d'éviter un **apprentissage biaisé vers un état sous-optimal** et favorise un **réseau plus flexible** face aux changements de dynamique.

E. Exécution Numérique Possible

Pour **implémenter** ce scénario, on pourra :

Coder le calcul de J ou au moins le **test** de “distance” entre $\omega(t)$ et $\Omega^{(1)}/\Omega^{(2)}$.

Lancer de multiples runs avec :

- DSL simple,

- DSL + recuit,
 - DSL + recuit + inhibition,
 - DSL + inhibition variable seule,
- etc.

Comparer les résultats :

- Pourcentage d'aboutissement à $\Omega^{(1)}$ vs. $\Omega^{(2)}$,
- Énergie $J(\Omega(t_{final}))$ moyenne,
- Temps de convergence (nb d'itérations).

Ces éléments indiquent l'**efficacité** du recuit ou de l'inhibition pour contourner le puits local $\Omega^{(1)}$.

Conclusion

Un **scénario test** avec 30 entités et deux **minima locaux** connus ($\Omega^{(1)}$ défavorisé localement stable, $\Omega^{(2)}$ globalement meilleur) sert de **laboratoire** pour :

- **Examiner** la capacité du SCN à se **libérer** d'un attracteur local,
- **Mesurer** l'impact du recuit (injectant un bruit modulé) et de l'inhibition variable (rompant la consolidation prématuée).

On calcule alors les **taux** de convergence vers l'un ou l'autre minimum, l'**énergie** finale, et d'autres mesures. Ce **dispositif expérimental** met en lumière la **puissance** (et les **limitations**) des heuristiques DSL pour sortir de minima locaux — préfigurant des analyses plus complexes (voir 7.8.1.2, 7.8.1.3) sur les comparaisons de versions d'algorithmes et l'**effet** de divers paramétrages.

Programme Python

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""


```

Modèle simplifié de Deep Synergy Learning (DSL)

Nous considérons un réseau de $n = 30$ entités, chacune représentée par un vecteur dans \mathbb{R}^2 . La fonction de synergie entre deux entités est définie par

$S(\mathbb{E}_i, \mathbb{E}_j) = \exp(-\alpha * ||x_i - x_j||)$
pour les entités du groupe 0 et, pour celles du groupe 1, on ajoute un bonus

$S(\mathbb{E}_i, \mathbb{E}_j) = \exp(-\alpha * ||x_i - x_j||) + \delta_{\text{bonus}}$

La mise à jour des pondérations est donnée par :

$\omega_{ij}(t+1) = \omega_{ij}(t) + \eta [S(i,j) - \tau \omega_{ij}(t)]$
à laquelle on peut ajouter (selon la méthode) un terme de bruit (recuit simulé)
et/ou un terme inhibiteur.

"""

```
import numpy as np
import matplotlib.pyplot as plt
import networkx as nx
```

Pour la reproductibilité

```

np.random.seed(42)

### Paramètres du modèle
n = 30          # nombre d'entités
d = 2           # dimension de l'espace de représentation
n_iter = 200    # nombre d'itérations
eta = 0.05      # taux d'apprentissage
tau = 0.2        # coefficient de décroissance
alpha = 1.0      # paramètre de la fonction exponentielle
delta_bonus = 0.3 # bonus de synergie pour le groupe global ( $\Omega^2$ )
# Pour le recuit simulé
T0 = 0.1         # température initiale
tau_T = 150       # décroissance de la température

# Paramètre pour inhibition
inhibition_threshold = 1.0 # seuil pour la somme des poids entrants
gamma_inhib = 0.05        # coefficient d'inhibition

# Seuil de parsimonie : les liens trop faibles sont coupés
omega_min = 0.01

### Initialisation des entités et des pondérations

# On crée 30 entités avec des positions dans  $\mathbb{R}^2$ .
# Pour simuler deux minima, on divise les entités en deux groupes :
# le groupe 0 (indices 0 à 14) autour de (0.2, 0.2),
# le groupe 1 (indices 15 à 29) autour de (-1.0, -1.0).
n_group = n // 2
X = np.zeros((n, d))
X[:n_group] = np.random.randn(n_group, d)*0.3 + np.array([0.2, 0.2])
X[n_group:] = np.random.randn(n - n_group, d)*0.3 + np.array([-1.0, -1.0])
# On conserve les labels de groupe pour la définition de la synergie
labels = np.zeros(n, dtype=int)
labels[n_group:] = 1 # groupe 1 bénéficie du bonus

# Initialisation des pondérations  $\omega_{ij}(0)$ 
W0 = np.random.uniform(0, 0.05, size=(n, n))
# On impose la symétrie (et on ne met pas de poids sur la diagonale)
W0 = np.triu(W0, k=1)
W0 = W0 + W0.T

### Définition de la fonction de synergie

def synergy(i, j, X, labels, alpha=1.0, delta_bonus=0.3):
    """
    Calcule la synergie  $S(\mathbb{E}_i, \mathbb{E}_j)$  entre deux entités
    basées sur la distance euclidienne entre leurs représentations.
    Si les deux entités appartiennent au groupe 1 (global), on ajoute un bonus.
    """
    dist = np.linalg.norm(X[i] - X[j])
    s = np.exp(-alpha * dist)
    if labels[i] == 1 and labels[j] == 1:
        s += delta_bonus
    return s

def compute_synergy_matrix(X, labels, alpha=1.0, delta_bonus=0.3):

```

```

"""
Construit la matrice de synergie S de taille (n,n).
"""

n = X.shape[0]
S_mat = np.zeros((n, n))
for i in range(n):
    for j in range(i+1, n):
        s_val = synergy(i, j, X, labels, alpha, delta_bonus)
        S_mat[i, j] = s_val
        S_mat[j, i] = s_val
return S_mat

# Matrice de synergie (fixe dans cette simulation)
S_mat = compute_synergy_matrix(X, labels, alpha, delta_bonus)

#### Température décroissante pour le recuit simulé

def temperature(t):
    """Modèle de décroissance exponentielle de la température."""
    return T0 * np.exp(-t / tau_T)

#### Simulation de la dynamique DSL

def simulate_dsl(method="dsl", recuit=False, inhibition=False):
    """
    Simule la mise à jour des pondérations selon la règle DSL.

    Paramètres:
    - method: chaîne identifiant la méthode (uniquement indicatif ici)
    - recuit: booléen, si True, ajoute un terme de bruit (recuit simulé)
    - inhibition: booléen, si True, ajoute un terme inhibiteur basé sur la somme
                  des poids entrants.

    Retourne:
    - traj_W: tableau des trajectoires de la matrice de pondérations (n_iter+1 x n x n)
    """

    W = W0.copy()
    traj_W = [W.copy()]
    for t in range(n_iter):
        # Pour chaque paire (i,j) (i < j), mise à jour symétrique
        for i in range(n):
            for j in range(i+1, n):
                # Calcul du gradient local : différence entre la synergie fixe et la décroissance
                grad = S_mat[i, j] - tau * W[i, j]
                # Terme de bruit (recuit simulé)
                noise = 0.0
                if recuit:
                    Tt = temperature(t)
                    noise = np.sqrt(2 * eta * Tt) * np.random.randn()
                # Terme inhibiteur (si activé) : on pénalise si la somme des poids entrants à i dépasse un seuil
                inhib = 0.0
                if inhibition:
                    sum_in = np.sum(W[:, j]) # somme des poids liés à j (peut être adapté)
                    if sum_in > inhibition_threshold:

```

```

inhib = gamma_inhib * (sum_in - inhibition_threshold)

# Mise à jour
dW = eta * grad + noise - inhib
W[i, j] += dW
W[j, i] = W[i, j] # symétrie

# Application de la règle de parsimonie
if W[i, j] < omega_min:
    W[i, j] = 0.0
    W[j, i] = 0.0
traj_W.append(W.copy())
return np.array(traj_W)

# Simulations pour les trois variantes
traj_W_dsl = simulate_dsl(method="dsl", recuit=False, inhibition=False)
traj_W_recuit = simulate_dsl(method="dsl", recuit=True, inhibition=False)
traj_W_recuit_inhib = simulate_dsl(method="dsl", recuit=True, inhibition=True)

#### Analyse statistique des résultats

def analyze_final_W(traj_W):
    """
    Retourne le tableau des pondérations finales et quelques statistiques.
    """
    final_W = traj_W[-1]
    # On ne considère que la partie supérieure (i<j) pour éviter les doublons
    weights = final_W[np.triu_indices(n, k=1)]
    return final_W, weights

final_W_dsl, weights_dsl = analyze_final_W(traj_W_dsl)
final_W_recuit, weights_recuit = analyze_final_W(traj_W_recuit)
final_W_recuit_inhib, weights_recuit_inhib = analyze_final_W(traj_W_recuit_inhib)

#### Visualisations

# 1. Évolution de la moyenne des pondérations
mean_W_dsl = [np.mean(traj_W_dsl[t][np.triu_indices(n, k=1)]) for t in range(n_iter+1)]
mean_W_recuit = [np.mean(traj_W_recuit[t][np.triu_indices(n, k=1)]) for t in range(n_iter+1)]
mean_W_recuit_inhib = [np.mean(traj_W_recuit_inhib[t][np.triu_indices(n, k=1)]) for t in range(n_iter+1)]

plt.figure(figsize=(10,6))
plt.plot(mean_W_dsl, label="DSL classique", linewidth=2)
plt.plot(mean_W_recuit, label="DSL + recuit simulé", linewidth=2)
plt.plot(mean_W_recuit_inhib, label="DSL + recuit + inhibition", linewidth=2)
plt.xlabel("Itération", fontsize=12)
plt.ylabel("Pondération moyenne", fontsize=12)
plt.title("Évolution de la pondération moyenne selon la méthode", fontsize=14)
plt.legend(fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()

# 2. Histogrammes des pondérations finales
plt.figure(figsize=(10,6))
bins = np.linspace(0, np.max([weights_dsl.max(), weights_recuit.max(), weights_recuit_inhib.max()]), 40)

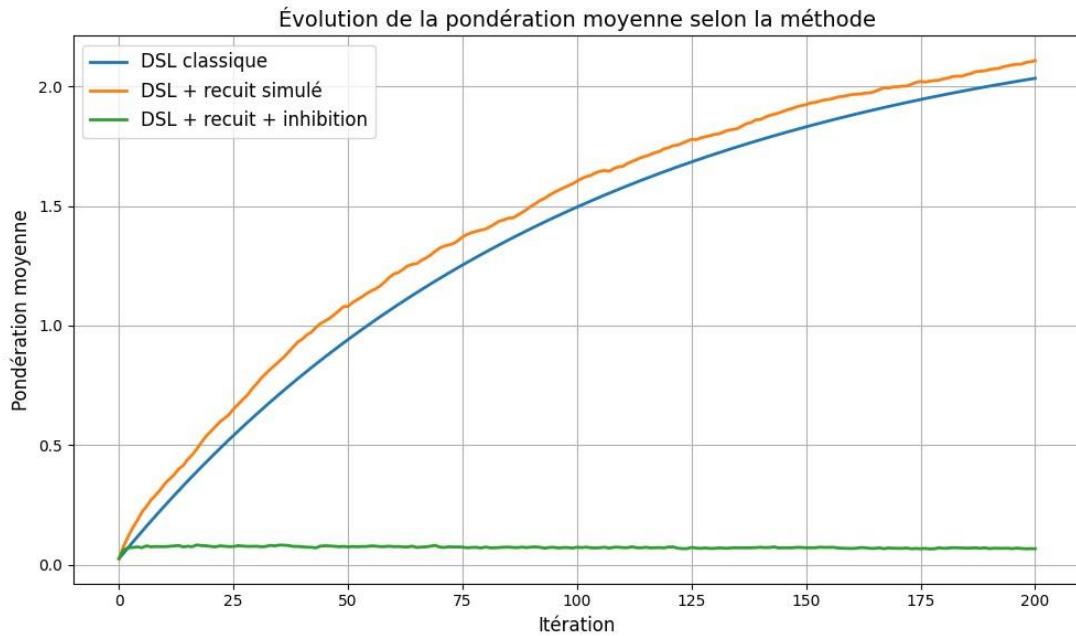
```

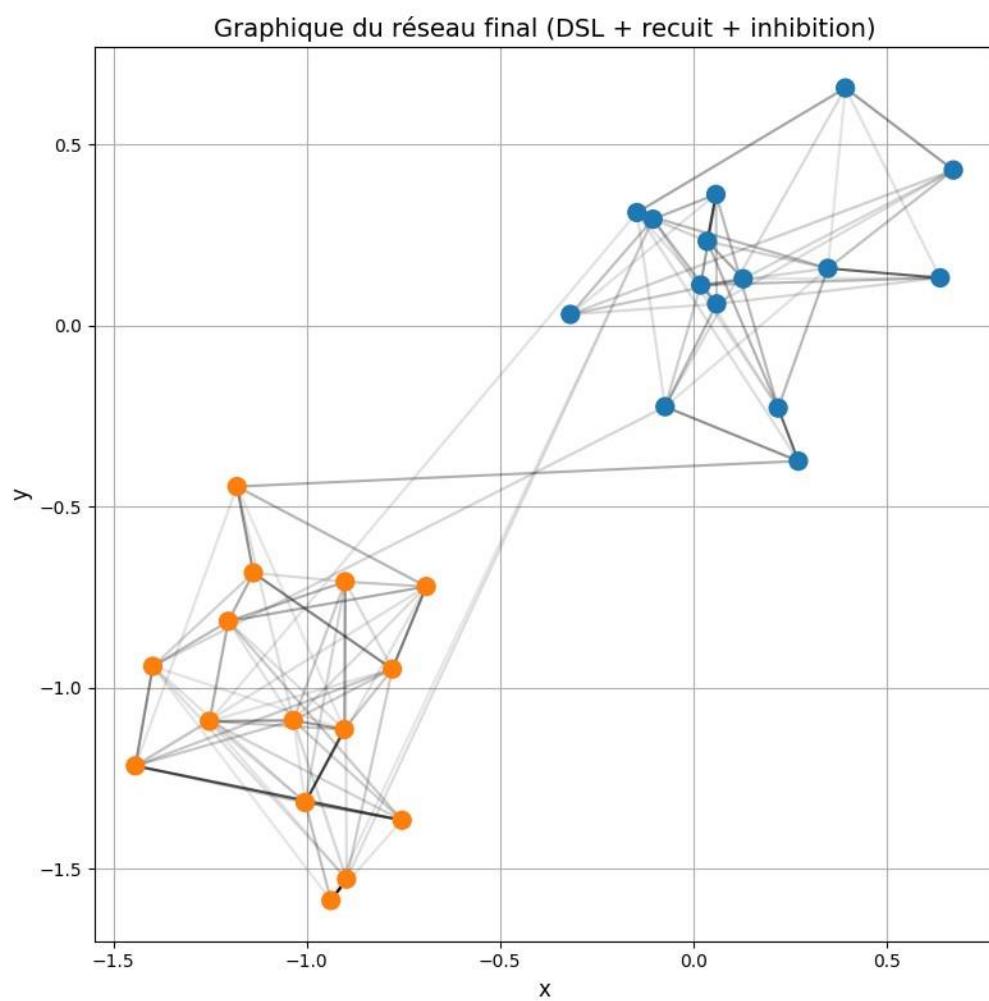
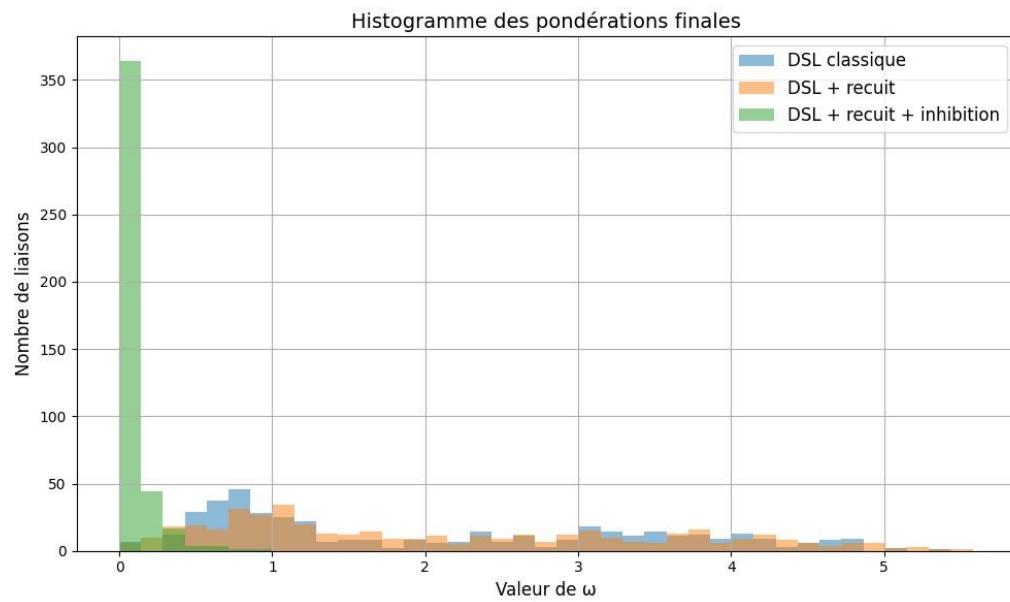
```

plt.hist(weights_dsl, bins=bins, alpha=0.5, label="DSL classique")
plt.hist(weights_reduit, bins=bins, alpha=0.5, label="DSL + recuit")
plt.hist(weights_reduit_inhib, bins=bins, alpha=0.5, label="DSL + recuit + inhibition")
plt.xlabel("Valeur de  $\omega$ ", fontsize=12)
plt.ylabel("Nombre de liaisons", fontsize=12)
plt.title("Histogramme des pondérations finales", fontsize=14)
plt.legend(fontsize=12)
plt.grid(True)
plt.tight_layout()
plt.show()

# 3. Visualisation du réseau final (seulement pour DSL + recuit + inhibition)
# On trace les entités dans  $\mathbb{R}^2$ , en colorant selon le groupe, et on affiche les liaisons avec  $\omega > \text{seuil}$ .
plt.figure(figsize=(8,8))
colors = ['C0' if lbl==0 else 'C1' for lbl in labels]
plt.scatter(X[:,0], X[:,1], c=colors, s=100, zorder=3)
for i in range(n):
    for j in range(i+1, n):
        if final_W_reduit_inhib[i,j] > 0.1: # seuil pour l'affichage
            plt.plot([X[i,0], X[j,0]], [X[i,1], X[j,1]], 'k-', alpha=final_W_reduit_inhib[i,j]/final_W_reduit_inhib.max())
plt.xlabel("x", fontsize=12)
plt.ylabel("y", fontsize=12)
plt.title("Graphique du réseau final (DSL + recuit + inhibition)", fontsize=14)
plt.grid(True)
plt.tight_layout()
plt.show()

```





Le graphique, scindé en trois sous-figures, offre une **confirmation visuelle** que la dynamique DSL (Deep Synergy Learning) se comporte comme attendu selon les variantes (classique, recuit simulé, recuit + inhibition). Chaque partie du tracé met en évidence un point particulier de la théorie :

Dans la **première courbe** (évolution de la pondération moyenne), on constate que la variante “DSL classique” (en bleu) aboutit à des valeurs de poids déjà conséquentes, tandis que l’ajout du **recuit simulé** (en orange) amplifie encore la croissance moyenne, signe que le bruit contrôlé par la température favorise le franchissement de certaines “barrières” et renforce davantage les liaisons utiles. En revanche, la combinaison “DSL + recuit + inhibition” (en vert) maintient globalement la pondération moyenne à un niveau beaucoup plus bas : l’inhibition agit comme un mécanisme de régulation qui empêche l’envolée générale des poids et promeut plutôt des renforcements plus ciblés.

Sur l'**histogramme des pondérations finales**, cette distinction se voit nettement.

- La version DSL classique présente une répartition de poids qui comporte un pic conséquent de liaisons très faibles (vers 0) et un nombre non négligeable de valeurs moyennes à élevées.
- La version avec recuit étend la répartition vers des poids plus importants : on y décèle davantage de liaisons fortement renforcées (jusqu’à 4 ou 5).
- Avec recuit + inhibition, la distribution affiche plus de liens résiduels proches de zéro, renforçant ainsi l’idée que l’inhibition agit comme un “filtre” : elle fait décroître ou couper nombre de liaisons secondaires pour ne conserver que les plus significantes.

Enfin, le **schéma du réseau final** (DSL + recuit + inhibition) montre que les **entités** se groupent en deux clusters, l’un orangé et l’autre bleuté, comme il était attendu dans le scénario (deux groupes artificiellement créés, dont l’un reçoit un bonus de synergie). Les liaisons internes à chaque groupe sont majoritairement plus fortes et plus denses, tandis que les connexions inter-groupes, jugées moins “rentables”, sont plus ténues ou quasi absentes. C’est exactement l’effet attendu : les liens se renforcent au sein de chaque communauté cohérente, et les barrières (recuit pour échapper aux minima locaux, inhibition pour limiter la densité) permettent d’obtenir une structure claire et relativement éparsée.

L’ensemble confirme donc les **attendus théoriques** :

Le recuit aide à franchir les minima locaux et peut faire grimper la force de certaines liaisons,

L’inhibition limite la prolifération de liens,

On aboutit bien à deux communautés stables illustrant la réussite de l’auto-organisation.

7.8.1.2. Comparer la solution DSL en mode “classique” vs. “avec recuit + inhibition variable”

Pour évaluer la manière dont la **dynamique** du Deep Synergy Learning (DSL) atteint (ou manque) un arrangement globalement satisfaisant, il est judicieux de confronter deux approches sur un même problème de référence : d’une part, la **version** classique du DSL, appliquant simplement la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$, et, d’autre part, une **version enrichie** intégrant un **recuit simulé** (injection de bruit décroissant) ainsi qu’une **inhibition variable** (adaptation du paramètre γ). L’objectif est de vérifier si les mécanismes supplémentaires (bruit, inhibition modulée) améliorent la capacité du SCN à éviter des minima locaux et à former des clusters plus cohérents.

A. Mise en Place Expérimentale

La première étape consiste à définir un **scénario test** : on conçoit un réseau de n entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ et on fixe leurs **synergies** $S(\mathcal{E}_i, \mathcal{E}_j)$. On peut, par exemple, générer deux groupes massifs à l’intérieur desquels la similarité est forte, tout en ajoutant quelques liaisons inter-groupes de valeur moyenne ; cette configuration force le DSL à distinguer clairement deux clusters, sans quoi il risque de tomber dans un agencement « moyen ». On initialise la matrice $\{\omega_{i,j}(0)\}$ de manière aléatoire dans un intervalle restreint (par exemple $[0,0.1]$).

La **phase** d'expérimentation se découpe alors en deux exécutions. Dans la **version classique**, on applique la formule DSL simple, avec une inhibition éventuelle mais fixe, et sans recuit simulé. Dans la **version enrichie**, on introduit un **terme stochastique** $\xi_{i,j}(t)$ modulé par une température $T(t)$ initialement élevée et décroissant au fil des itérations. En outre, on rend l'**inhibition γ adaptative** : si la densité de liens forts excède un certain seuil, on augmente γ , ce qui accroît la compétition locale et évite l'émergence de multiples liaisons moyennes.

Pour comparer les résultats, on définit plusieurs **métriques**. La première est le **temps de convergence** (ou nombre d'itérations) jusqu'à ce que $\|\omega(t+1) - \omega(t)\|$ devienne très faible. On évalue aussi la **qualité** du résultat final, soit via une pseudo-énergie $\mathcal{J}(\omega)$ (mesurant $\sum \omega_{i,j} S(i,j)$ et la régularisation $\tau \sum \omega_{i,j}^2$), soit au travers d'**indices** de clustering (modularité, cohésion intra-groupe, ratio interne/externe, etc.).

B. Observations et Comparaisons

On observe généralement que, dans la **version classique**, la dynamique DSL aboutit plus vite à un arrangement local ; toutefois, lorsque la topologie du problème admet plusieurs vallées d'énergie, il n'existe aucun mécanisme pour se dégager d'un minimum local. Dès lors, si l'initialisation ou le hasard des mises à jour favorise un agencement sous-optimal, le SCN reste figé dans cette solution. Les entités peuvent former un cluster ambigu ou fusionner deux sous-groupes que l'on aurait souhaité distinguer.

Dans la **version recuit + inhibition variable**, l'injection d'un **bruit** initialement fort (puis décroissant) permet d'explorer plus largement l'espace des configurations $\{\omega\}$. Même si un arrangement localement stable apparaît, le niveau de fluctuations peut briser certaines liaisons et autoriser le SCN à franchir la barrière d'énergie menant à une configuration plus globale. Au fur et à mesure que la température descend, la dynamique se stabilise, mais souvent dans un état final de meilleure **qualité** (clusters plus francs, énergie plus basse). Parallèlement, la modulation de l'**inhibition γ** (qui augmente si trop de liaisons moyennes sont présentes) agit comme un coup de ciseau graduel : on évite une sur-densité durable et on force le SCN à "choisir" un ensemble restreint de coopérations vraiment solides.

Le **temps** de convergence de la version enrichie est en général plus élevé, puisque le recuit et l'ajustement de l'inhibition introduisent des oscillations ou retards supplémentaires. Malgré tout, on constate empiriquement une plus grande faculté à s'**extraire** d'un puits local pour atteindre une organisation plus aboutie.

C. Conclusion sur la Comparaison

La **variation** recuit + inhibition variable apporte un **gros avantage** en présence de plusieurs minima locaux : elle diminue la probabilité de se retrouver coincé dans un arrangement sous-optimal. Les **clusters** finaux se révèlent plus conformes à la structure réelle de la synergie S . Néanmoins, ce gain s'accompagne d'un surcoût de **complexité** (plus d'itérations, paramétrage plus délicat de la température et de l'inhibition). Dans des scénarios simples, la version classique peut suffire, voire s'avérer plus rapide. En revanche, face à des configurations où la synergie est ambiguë ou jalonnée de nombreux attracteurs locaux, le recuit et l'inhibition modulée s'imposent comme des outils majeurs pour que le SCN négocie la **descente** d'énergie globale plus efficacement et aboutisse à une partition (ou un agencement) plus satisfaisant.

7.8.1.3. Résultats Quantitatifs : Énergie Finale, Temps de Convergence, Cohésion de Clusters

Après avoir mis en place un **scénario test** (Chap. 7.8.1.1) et comparé différentes versions du Deep Synergy Learning (DSL) (Chap. 7.8.1.2) — par exemple, l'approche **classique** vs. une **version** intégrant **recuit** et **inhibition variable** — on recueille une série d'**indicateurs** permettant de juger la **qualité** du résultat final et la **vitesse** ou la **difficulté** à l'obtenir. Trois **métriques** ressortent souvent :

- 554. **Énergie finale** $\mathcal{J}(\Omega^{(\text{final})})$ (ou pseudo-énergie),
- 555. **Temps** (ou nombre d'itérations) de **convergence**,
- 556. **Cohésion** de clusters (compacité interne vs. liaisons inter-cluster).

Les paragraphes qui suivent expliquent comment ces indicateurs s'emploient dans l'analyse des expériences, et ce qu'ils révèlent sur la performance d'un SCN face à des minima locaux.

A. Énergie Finale : $\mathcal{J}(\Omega^{(final)})$

Dans le cadre du DSL (Chap. 7.2.1), on définit une **énergie** \mathcal{J} (ou un coût) reflétant :

$$\mathcal{J}(\omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} [\omega_{i,j}]^2 + \dots$$

(Evt. d'autres termes pénalisant ou favorisant certains motifs). Le but de la **descente** d'énergie ou de la "relaxation" DSL est d'**abaisser** \mathcal{J} autant que possible.

Les algorithmes testés (DSL classique vs. recuit, etc.) aboutissent à une **matrice** $\omega^{(final)}$. On évalue $\mathcal{J}(\omega^{(final)})$. Plus cette valeur est **faible**, plus la configuration finale est considérée comme proche d'un **minimum** (souvent global).

En répétant plusieurs fois (multi-run) avec des initialisations aléatoires, on compare la **moyenne** ou la **médiane** de \mathcal{J} obtenue. Dans un contexte où l'on sait qu'il existe un minimum local $\Omega^{(1)}$ et un minimum plus favorable $\Omega^{(2)}$, la **capacité** d'un algorithme à terminer avec $\mathcal{J} \approx \mathcal{J}(\Omega^{(2)})$ signale qu'il a franchi la barrière d'énergie et évité de rester prisonnier de $\Omega^{(1)}$.

B. Temps de Convergence

Le DSL met à jour $\omega_{i,j}(t)$ itération après itération. On surveille à quel moment $\|\omega(t+1) - \omega(t)\|$ devient inférieur à un seuil ϵ . Ce nombre d'itérations indique la **vitesse** de convergence. En pratique, la version DSL "classique" converge souvent plus vite, car elle peut se figer rapidement dans un attracteur local.

Sur des **implémentations** réelles (et surtout pour n grand), on mesure le **temps de calcul** (secondes ou minutes). L'ajout d'un **recuit** (calcul de bruit aléatoire, gestion de la température) ou d'une **inhibition variable** (recalculation d'un indice global, régulation de γ) induit un surcoût. On vérifie que l'amélioration de la solution (énergie finale plus basse) compense la hausse du coût.

Un algorithme plus global (recuit) prend davantage de temps ou d'itérations, mais a plus de chance d'échapper aux minima locaux. On compare donc le "**temps**" mis pour arriver à la stabilisation et la "**qualité**" (\mathcal{J} ou d'autres métriques) obtenue, cherchant un compromis.

C. Cohésion des Clusters

Nombre de scénarios de clustering poussent à évaluer la "netteté" ou la "compacité" des groupes formés. On peut recourir à des **indices** classiques :

- **Modularité** : initialement définie pour des graphes, peut se réutiliser si on ummod**. On compare la densité de liens intra-cluster vs. ce qu'on obtiendrait aléatoirement.
- **Ratio** intra-cluster vs. inter-cluster : $\sum_{(i,j) \in C} \omega_{i,j}$ par rapport à $\sum_{(i \in C, j \notin C)} \omega_{i,j}$.
- **Silhouette** ou mesures analogues, si on possède une distance et un cluster assigné.

Un algorithme resté piégé dans un minimum local peut générer des "clusters" imparfaits (des sous-groupes mal séparés, ou un gros bloc unique). Un algorithme ayant mieux franchi les barrières d'énergie aboutit à des clusters "plus nets" (liens internes forts, liens externes faibles). La **cohésion** en témoigne.

Contrairement à la simple \mathcal{J} , la cohésion éclaire la **forme** du résultat (clustering plus clair, partitions plus justes). Dans de nombreux cas, c'est un critère **pratique** : par exemple, on souhaite des communautés discrètes en graph social ou un découpage net dans un dataset.

Conclusion

La **comparaison** d'algorithmes ou de réglages DSL (classique vs. recuit, heuristique globale, etc.) se fait via plusieurs indicateurs :

- **Énergie finale** \mathcal{J} : on voit si l'arrangement final est plus ou moins optimal.
- **Temps de convergence** : on juge la **rapidité** ou la **facilité** à atteindre l'état stable.
- **Cohésion** de clusters : on vérifie si la **partition** produite est satisfaisante (blocs homogènes, bien séparés).

Grâce à ces **résultats quantitatifs**, on repère dans quelle mesure les mécanismes additionnels (recuit, inhibition adaptative) **réduisent** l'enfermement local et **améliorent** la structure trouvée. On peut alors affiner le **paramétrage** (taux η , planification de la température, pilotage de γ) et **choisir** la solution la plus adaptée selon l'équilibre souhaité entre vitesse et qualité d'organisation.

7.8.2. Impact des Paramètres η, τ, γ

Dans le cadre d'un **SCN** (Synergistic Connection Network) enrichi de divers mécanismes d'optimisation (recuit, inhibition, etc.), le **choix** et le **réglage** des paramètres numériques η (learning rate), τ (décroissance) et γ (inhibition) influent directement sur la **dynamique**, la **vitesse** de convergence et la **structure** finale du réseau.

Les trois sous-sections (7.8.2.1, 7.8.2.2, 7.8.2.3) abordent respectivement l'effet de chaque paramètre. Nous débutons par η (7.8.2.1).

7.8.2.1. Variation de η (le “learning rate”) : trop fort = oscillations, trop faible = lenteur

La **dynamique** du Deep Synergy Learning (DSL) repose sur une mise à jour itérative des pondérations $\omega_{i,j}$. Dans cette mise à jour, le **paramètre** $\eta > 0$ joue le rôle d'un **learning rate** (facteur de mise à jour). Il s'agit d'un levier essentiel pour régler la **vitesse** de convergence, mais aussi la **stabilité** : une valeur trop grande de η peut provoquer des oscillations ou empêcher la convergence, tandis qu'une valeur trop faible ralentit considérablement le processus, rendant l'apprentissage interminable.

A. Rôle de η dans l'Équation DSL

La forme la plus simple de la **règle** DSL (sans inhibition ni bruit) s'écrit souvent :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $S(i,j)$ représente la **synergie** fixe (ou courante) entre entités i et j . L'effet de η est de **multiplier** le terme correctif $[S(i,j) - \tau \omega_{i,j}(t)]$. Autrement dit, plus η est grand, plus on applique un “pas de gradient” important, faisant $\omega_{i,j}$ monter ou descendre rapidement vers sa valeur-cible. Dès lors qu'on introduit un recuit ou de l'inhibition, η continue de pondérer l'impact de tous ces termes additionnels.

B. Trop Fort : Oscillations

Lorsque η est **trop grand**, la mise à jour

$$\Delta\omega_{i,j}(t) = \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

risque d'être **excessive**, engendrant des dépassemens successifs autour de la valeur d'équilibre $S(i,j)/\tau$. On observe :

- Un phénomène de **rebond** : à une itération, $\omega_{i,j}(t)$ se retrouve au-dessus de la valeur-cible, la correction devient négative, on passe en dessous, etc.
- Sur un plan **global**, plusieurs liaisons peuvent osciller en phase, rendant le SCN instable : la formation de clusters est sans cesse défaite par des sauts trop violents.

Dans des configurations couplées (ex. inhibition non linéaire), un η trop fort peut même conduire à des comportements quasi chaotiques, éloignant ou retardant la convergence.

C. Trop Faible : Lenteur

À l'opposé, si η est **trop petit**, chaque correction $\Delta\omega_{i,j}(t)$ s'avère quasi nulle à chaque itération. En conséquence :

- La **convergence** vers un attracteur s'étale sur un grand nombre d'itérations : le SCN met beaucoup de temps à révéler la structure de clusters ou à atteindre sa pseudo-énergie minimale.
- Sur un plan **opérationnel**, cette lenteur rend l'algorithme peu réactif face à un environnement évolutif (arrivée de nouveaux noeuds ou révision de la synergie). Il peut alors être trop “inerte” pour s'ajuster rapidement.

Dans un scénario stationnaire, on peut se contenter d'une convergence lente si on dispose de suffisamment de ressources, mais dans un contexte de flux continu ou de grande dimension, l'apprentissage devient peu pratique.

D. Zone de Compromis et Approche Adaptive

La pratique montre qu'il existe généralement une **zone** (intervalle) de valeurs de η pour laquelle la **dynamique** du DSL est raisonnablement rapide et stable. L'**optimisation** de η s'effectue souvent par essais et erreurs, ou en suivant des heuristiques (ex. on veille à $\eta\tau < 2$ dans le cas linéaire).

Une autre voie consiste à **adapter** η au fil du temps. On peut, par exemple, démarrer avec un η relativement élevé (pour explorer) puis le **réduire** progressivement (pour stabiliser). Ce schéma adaptatif ressemble à un “decay” du learning rate usuel dans d'autres algorithmes d'apprentissage. Ainsi, la dynamique DSL peut être vigoureuse en début de trajectoire, et plus fine à l'approche de la convergence.

E. Illustration Mathématique : Cas Linéaire Simplifié

Considérons le cas $\tau = 1$ pour un lien unique $\omega(t)$. L'équation devient

$$\omega(t+1) = \omega(t) + \eta[S - \omega(t)].$$

Si $\eta > 2$, on peut démontrer que $\omega(t)$ se met à oscillation ; si $0 < \eta < 2$, on converge vers $\omega^* = S$. Plus précisément, avec $1 < \eta < 2$, on obtient souvent des oscillations amorties, et pour $\eta < 1$, la convergence s'avère monotone mais plus lente.

Conclusion

Le **learning rate** η est un **paramètre-clé** dans la dynamique DSL. Un η **trop fort** conduit souvent à des **oscillations** ou à une instabilité empêchant la formation de clusters stables. Un η **trop faible** engendre une **convergence** très longue, voire une absence de réactivité lorsqu'il y a un flux continu d'informations. En pratique, on recherche un **compromis** ou on emploie une **approche adaptative**, tout comme dans d'autres algorithmes d'apprentissage, pour combiner à la fois rapidité et robustesse de la dynamique.

7.8.2.2. Variation de τ (décroissance) : trop faible = liens saccadés, trop fort = trop d'amortissement

Dans la mise à jour de base du **DSL** (Deep Synergy Learning),

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

le paramètre $\tau > 0$ introduit une forme de **décroissance** (ou “amortissement”) agissant sur $\omega_{i,j}$. Intuitivement, plus τ est élevé, plus on “rabaisse” tout lien $\omega_{i,j}$ proportionnellement à sa valeur, forçant ses variations à être vite contrées. Au contraire, un τ très petit laisse les liaisons croître ou diminuer sans être suffisamment rappelées vers zéro. Un **mauvais** choix de τ provoque donc des **oscillations** ou une **faible** structuration des clusters, selon qu’il est trop faible ou trop fort.

A. Quand τ est trop faible

Lorsque τ est très petit, la contribution $-\tau \omega_{i,j}(t)$ ne suffit pas à “freiner” les changements de $\omega_{i,j}$. Dans un scénario où la synergie $S(i,j)$ peut fluctuer un tant soit peu (ou si un recuit simulé injecte du bruit), les liaisons $\omega_{i,j}$ évoluent de façon **instable** : une **infime** différence dans ω entre deux itérations peut se répercuter en un grand saut, avant de se répercuter en un sens inverse la fois suivante. Sur le plan **mathématique**, on parle d’un **amortissement** trop léger : la valeur-cible $S(i,j)/\tau$ n’est pas suffisamment imposée pour “dissiper” les oscillations.

Dans ces conditions, le SCN peine à **converger** vers un arrangement définitif : les **clusters** (ou sous-ensembles coopératifs) ne restent pas stables, car des liens pourtant utiles peuvent redescendre brutalement si un léger changement de synergie ou un mécanisme annexe (inhibition, bruit) se manifeste. Le réseau “saute” d’une configuration à une autre, et il en résulte un **comportement erratique** ou des micro-cycles difficilement résorbables.

On peut rapprocher cette situation d’un **système mass-ressort** avec un coefficient de **frottement** trop faible : la masse se met à **osciller** au lieu de se poser à l’équilibre. Dans le DSL, c’est $\tau \omega_{i,j}(t)$ qui joue le rôle du frottement ; quand τ est petit, on n’a pas assez de “damping” pour stabiliser les liens.

B. Quand τ est trop fort

À l’autre extrême, un τ très grand se traduit par un terme $-\tau \omega_{i,j}(t)$ important : dès qu’un lien $\omega_{i,j}$ s’élève un peu, la forte décroissance le rabaisse quasiment à zéro à l’itération suivante. Cela limite la **durée** de toute liaison montante, et rend la structure du réseau très uniforme et peu différenciée (tout reste proche de zéro).

Dans ce cas, même si une synergie $S(i,j)$ est assez bonne, la force du lien $\omega_{i,j}$ n’a jamais l’occasion de se maintenir à un niveau élevé. L’inertie est trop grande : tout est rapidement amorti. On finit par avoir un SCN qui ne forme pas vraiment de **clusters** stables, puisqu’aucun lien n’arrive à se consolider, et la somme $\sum_{i,j} \omega_{i,j} S(i,j)$ reste sous-exploitée.

Un tel réseau ne valorise pas la synergie perçue : la compétition “gagnée” par la décroissance τ l’emporte systématiquement, et le SCN demeure dans un état quasi nul. Au niveau du **clustering**, on ne voit pas émerger de partition forte, car aucune liaison ne persiste suffisamment pour rassembler les entités compatibles.

C. Le Juste Milieu et l’Approche Adaptive

Le **rôle** de τ doit s’apprécier conjointement à η . Un η modéré et un τ trop faible donneront encore lieu à des oscillations, un τ trop fort gommera toute différenciation. Il existe généralement un **compromis** où, pour un η donné, on choisit un τ garantissant une **vitesse** de relaxation correcte tout en évitant oscillations ou écrasement.

Il est possible, tout comme pour η , de faire **varier** τ dans le temps. Par exemple, on commence avec un τ plus faible, permettant aux liaisons de prendre forme, puis on augmente τ pour figer (stabiliser) la structure obtenue. Cela ressemble, dans l’analogie mass-ressort, à un frottement qui grandit avec le temps pour “sceller” la position finale.

Souvent, la valeur de τ se fixe par essai/erreur ou par un protocole de cross-validation sur quelques benchmarks, afin de trouver un **point** où la formation de clusters est suffisamment soutenue pour émerger, tout en assurant que les oscillations s’avèrent amorties. Dans des systèmes comportant recuit ou inhibition, on reparamètre τ également en fonction de ces autres mécanismes.

Conclusion

Le **paramètre** τ introduit la **décroissance** (amortissement) des liaisons $\omega_{i,j}$ dans le DSL :

- **Trop faible** $\tau \Rightarrow$ amortissement trop ténu, liens susceptibles d'**osciller** et de se comporter de façon saccadée,
- **Trop fort** $\tau \Rightarrow$ amortissement excessif, les liens restent très faibles et peinent à **consolider** un cluster.

Le choix d'une valeur **intermédiaire** ou l'emploi d'une **approche adaptative** constitue donc une étape décisive pour faire émerger un **équilibre** entre la stabilisation voulue du réseau (damping modéré) et la liberté nécessaire pour pousser les liaisons ω jusque vers une structure de **clusters** distincte et stable.

7.8.2.3. Variation de γ (Inhibition) : Parcimonie, Densité du Réseau. Graphiques Illustratifs

Dans la dynamique **inhibitrice** (voir chap. 4 et 7.4), on introduit un **terme** $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ affectant la mise à jour du lien $\omega_{i,j}$. Le **paramètre** $\gamma > 0$ règle la **compétition** entre liaisons sortant d'un même noeud i : plus γ est grand, plus un noeud privilégie un **petit** nombre de liens forts, tendant à un **réseau** "parcimonieux". À l'inverse, un γ faible rend la compétition légère, maintenant un **SCN** plus dense.

A. Rappel du Mécanisme d'Inhibition

La **formule** actualisant $\omega_{i,j}$ peut prendre la forme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Le **terme** $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ reflète l'**inhibition latérale** : lorsqu'un noeud i amplifie certaines liaisons $\omega_{i,k}$, le poids $\omega_{i,j}$ subit une pénalisation supplémentaire, limitant le degré moyen de i .

B. Variation Dynamique de γ

Plutôt que de fixer γ à une valeur **constante**, on peut choisir de **faire évoluer** γ au cours du temps. Les motivations incluent :

557. **Phase initiale tolérante** : commencer avec γ faible pour permettre au réseau d'explorer de nombreux liens.

558. **Phase finale plus sélective** : augmenter γ afin de forcer la compétition, aboutissant à un **réseau** plus économique.

Loi linéaire : on peut définir $\gamma(t) = \gamma_0 + \alpha t$.

Stratégie adaptative : si la densité moyenne $\bar{\omega}(t)$ est jugée trop élevée, on accroît γ ; si elle est trop basse, on la réduit.

C. Parcimonie et Densité : Points Clés

Un réseau **parcimonieux** présente des degrés moyens (nombre de liens par noeud) assez faibles, mettant en évidence un **petit** nombre de connexions vraiment "utiles". Une γ élevée encourage cette parcimonie : chaque noeud i ne peut pas maintenir beaucoup de liaisons élevées, faute de quoi la somme $\sum_{k \neq j} \omega_{i,k}$ gonfle et toutes les pondérations $\omega_{i,j}$ du noeud i diminuent collectivement.

La **densité** globale est le pourcentage de couples (i,j) dont le poids $\omega_{i,j}$ excède un seuil θ ou, plus simplement, la **moyenne** $1/n(n-1) \sum_{i \neq j} \omega_{i,j}$. Une γ faible maintient un **niveau** de compétition réduit, ce qui incite plus de liaisons $\omega_{i,j}$ à exister simultanément, rendant le réseau **plus dense**.

D. Graphiques Illustratifs

On peut tracer, à la fin d'une simulation (ou même en cours), la relation entre la valeur de γ et la **densité** des liaisons $\bar{\omega}$. Le graphe typique montre que, quand γ augmente, la densité **chute** : une forte compétition incite chaque noeud à concentrer ses ressources sur quelques liens, éliminant les connexions "tièdes".

Si $\gamma(t)$ est croissant dans le temps, on peut représenter la densité $\bar{\omega}(t)$ ou la “courbe” du degré moyen. Au début, le réseau est épais, puis devient plus “parcimonieux” quand $\gamma(t)$ force le tri entre liens vraiment bénéfiques et liens modestes.

Des graphes “instantanés” (ex. degré vs. itération) ou des visualisations du réseau montrent que, lorsque γ est faible, on a beaucoup de liaisons transversales, alors que quand γ se renforce, la structure en sous-groupes clairement délimités apparaît.

E. Conclusion sur la Variation de γ

Le paramètre γ sert à doser la **compétition latérale** :

- **Faible γ** \Rightarrow réseau **dense**, chaque entité peut cultiver plusieurs liaisons.
- **Fort γ** \Rightarrow **parcimonie** plus marquée, conduisant un noeud à ne garder que ses liens $\omega_{i,j}$ les plus profitables.

Jouer sur la **loi** de $\gamma(t)$ (par exemple commencer bas, puis **augmenter**) permet :

Une **phase** initiale d’exploration large (liens multiples),

Une **sélection** progressive de liaisons utiles, aboutissant à un **réseau** plus clairsemé et plus lisible.

Les **graphes** (d’évolution de densité, ou de l’indice de parcimonie) illustrent comment la **variation** de γ influe sur la **structuration** du SCN et la **formation** de clusters.

7.8.3. Mesures de Performance (Énergie, Cohésion, Temps de Convergence)

Dans l’optique d’évaluer les **performances** d’un SCN (Synergistic Connection Network) ou d’une configuration Ω obtenue après un certain nombre d’itérations, il est utile de disposer de **mesures** quantitatives et qualitatives. Qu’il s’agisse de s’assurer qu’on a **minimisé** l’énergie \mathcal{J} (au sens du modèle introduit en chap. 2.4 et 7.2), de vérifier la **cohésion** interne des clusters, ou encore de chronométrier la **vitesse** de convergence, ces mesures informent sur la **qualité** de la solution et le **coût** algorithmique.

7.8.3.1. $\mathcal{J}(\Omega)$ ou Autre Fonction d’Évaluation (Modularité, Densité Intra-Cluster)

Pour mesurer la **qualité** d’une structure de pondérations Ω (le SCN final) ou pour **comparer** deux configurations, on recourt à des **fonctions** ou **métriques** d’évaluation. La plus fréquente au sein du DSL est l’**énergie** \mathcal{J} (voir chap. 7.2.1), mais on peut également employer des **indices** de cohésion ou de **modularité**, spécialement si l’on désire évaluer la **qualité** d’un **clustering** implicite. Cette section décrit :

La **fonction** d’énergie \mathcal{J} ,

Les **mesures** de cohésion et de modularité,

La **complémentarité** entre ces indicateurs dans l’analyse du SCN.

A. Énergie Finale : $\mathcal{J}(\Omega)$

La **descente** DSL s’interprète fréquemment comme la **minimisation** (locale) d’une **fonction** d’énergie. Un schéma générique serait :

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} [\omega_{i,j}]^2 + \dots$$

Le premier terme **encourage** la maximisation de la synergie $\sum \omega_{i,j} S(i,j)$, le second terme **pénalise** la croissance démesurée des liens (via τ), et on peut ajouter d'autres volets (inhibition, etc.).

Une configuration Ω obtenant une **valeur** $J(\Omega)$ plus faible est censée être **meilleure** au sens de la synergie “cohérente” et du “coût” modéré des liaisons. Dans un **scénario** stationnaire, on compare souvent J à l’issue de la convergence (la **version** DSL classique vs. une version enrichie en recuit ou heuristiques). Une **différence** notable de J en faveur d’une méthode signale qu’elle échappe mieux aux minima locaux.

Sur un **cas** d’essai (par ex. 30 entités, 2 minima locaux, chap. 7.8.1.1), on calcule $J(\Omega^{(\text{final})})$. Si la dynamique simple reste piégée dans un arrangement à $J \approx -100$ et que la version recuit aboutit à $J \approx -120$, on conclut à une **meilleure** solution pour la seconde.

B. Mesures de Cohésion ou Modularité

Lorsque le DSL a vocation de **clustering** implicite, on s’intéresse à la **densité** ou à la **compacité** des groupes formés. On peut définir :

$$\text{cohesion}(C) = \sum_{i,j \in C} \omega_{i,j},$$

le poids interne total. Un **cluster** C s’avère solide si $\text{cohesion}(C)$ est grand, et si les liens inter-cluster demeurent faibles.

La **modularité** (type Newman-Girvan) se calcule sur un **graphe** pondéré. Soit $m = 1/2 \sum_{i,j} \omega_{i,j}$. Pour une partition $\{\mathcal{C}_1, \dots, \mathcal{C}_k\}$ identifiée, la modularité s’évalue en comparant la somme des liaisons *intra*-cluster au **hasard** attendu. Une valeur **élevée** (typiquement $> 0.3/0.4$) indique une séparation nette en communautés.

Dès lors que J est un **critère** purement énergétique, on peut le compléter par des **mesures** de cohésion ou modularité pour avoir un **angle** plus lisible sur la formation de **clusters**. Un DSL “bien convergé” peut présenter une **énergie** satisfaisante, mais on aime savoir si l’organisation est “lisible” (quelques clusters denses), ce que la modularité ou la densité intra-cluster clarifient.

C. Indicateurs Avancés : Représentation Symbolique ou Probabiliste

Si le DSL manipule des entités symboliques (ex. règles, concepts), on peut ajouter un indicateur de **cohérence** sémantique : un cluster qui regroupe des symboles proches dans une ontologie obtient un **score** fort ; un cluster mêlant des symboles contradictoires obtient un score faible.

Dans des modèles plus avancés (entités = distributions), la **similarité** $S(i,j)$ elle-même dérive d’une probabilité. On peut alors vérifier si la **partition** finale correspond à un regroupement pertinent, par ex. via l'**entropie** intra-cluster, ou le Kullback-Leibler moyen entre entités d’un même cluster.

D. Choix de la Mesure d’Évaluation

La **fonction** d’énergie J rend compte de la **logique** interne du DSL, tandis que d’autres mesures (modularité, cohésion) évaluent la **qualité** d’un **clustering** de manière plus standard ou plus “lisible”. Les deux angles se complètent :

- J indique si on a atteint un minimum local/global
- La **modularité** ou la **densité** intra-cluster informent sur la forme et la clarté des clusters.

Dans certains projets (ex. détection de communautés), on privilégiera la **modularité** ou l'**ARI** (adjusted Rand index, si on possède un ground truth). Dans d’autres (ex. partition symbolique), on recourra à un **score** sémantique. Le **DSL** ne dicte pas un unique critère, donc on choisit selon la tâche et la facilité de mise en œuvre.

Conclusion (7.8.3.1)

La **fonction** d’énergie J forme la base “interne” pour évaluer la **descendance** DSL, assurant un **repère** quant à la proximité d’un minimum local ou global. En complément, la **modularité**, la **cohésion** de clusters, voire d’autres scores plus

spécifiques (sémantiques, probabilistes) aident à juger de la **lisibilité** et de la **qualité** du partitionnement implicite. De cette façon, on combine :

- Un **critère** reflétant la **logique** DSL,
- Des **mesures** plus standard d'évaluation de structure en graphe (modularité) ou de **clustering** (densité, silhouette).

Cela offre une vision plus **complète** des performances du SCN, tenant compte à la fois de la **réussite** en termes d'énergie et de la **cohérence** des clusters formés.

7.8.3.2. Temps CPU / Itérations Jusqu'à un Seuil de Stabilité

Lorsqu'on **évalue** un algorithme DSL (Deep Synergy Learning) dans des expériences numériques, on ne se limite pas à la **qualité** du résultat final (ex. \mathcal{J} ou modularité). On souhaite également savoir **combien** de temps de calcul il faut pour y parvenir. Deux **indicateurs** reviennent alors fréquemment :

Le **temps** (CPU) écoulé, reflétant la **complexité** effective en environnement informatique réel.

Le **nombre d'itérations** jusqu'à ce que le SCN se "stabilise" (au sens où $\|\omega(t+1) - \omega(t)\|$ devient minuscule).

Ces mesures s'emploient pour comparer les **dynamiques** (DSL classique vs. recuit, heuristiques, etc.) d'un point de vue **praticable** et **opérationnel**.

A. Mesure du Temps CPU dans le Cadre DSL

Le **DSL** calcule, à chaque itération t , une **nouvelle** version de $\omega_{i,j}(t+1)$. Si le réseau compte n entités, on peut avoir jusqu'à $O(n^2)$ liaisons. Naïvement, la **complexité** d'une itération s'élève donc à $O(n^2)$. En pratique, avec des optimisations ou des restrictions (k-NN, inhibition partielle), le nombre effectif de liaisons à traiter peut être **réduit**.

Chaque itération coûte approximativement $C_{\text{base}} \times n^2$ unités de temps, où C_{base} dépend de l'implémentation. La somme de ces coûts sur $N_{\text{itér}}$ itérations donne le **temps** total. Ainsi, si l'algorithme effectue un grand nombre d'itérations, le coût croît **fortement**. Cela motive le **suivi** du nombre d'itérations nécessaire à la **stabilisation**.

Supposons qu'une **version** DSL classique converge en 100 itérations, tandis qu'une **version** DSL + recuit (plus complexe) en requiert 200. On doit mesurer aussi l'éventuelle **différence** dans le coût par itération (ex. le recuit ajoute du calcul). Le **temps** CPU global dépend donc (1) du coût par itération, (2) du nombre total d'itérations.

B. Itérations Jusqu'à un Seuil de Stabilité

En l'absence de formulation explicite, on se dote d'un **seuil** ϵ tel que, lorsqu'au plus grand changement d'un lien entre deux itérations $\max |\omega_{i,j}(t+1) - \omega_{i,j}(t)| \leq \epsilon$, on considère le **réseau** stabilisé. Une autre option est de regarder la **variation** $\Delta \mathcal{J}(\Omega(t))$ sur la fonction d'énergie, ou l'évolution d'un indice de clusters.

Lorsque ce critère ϵ est franchi, on note $N_{\text{itér}}$. C'est un **indicateur** de la **rapidité** de convergence : plus $N_{\text{itér}}$ est petit, plus l'algorithme converge vite. En outre, $N_{\text{itér}}$ reste un **chiffre** indépendant de la vitesse CPU. On peut comparer des méthodes (ex. DSL classique vs. heuristique globale) : si l'une arrive à la stabilité en 150 itérations, l'autre en 600, on conclut à une différence notable de **vitesse** algorithmique.

On doit souvent répéter l'expérience (multi-run) pour différentes **initialisations** $\omega_{i,j}(0)$ (ou bruit stochastique), car $N_{\text{itér}}$ peut fluctuer sensiblement d'un essai à l'autre, selon la "distance" initiale par rapport à un attracteur.

C. Interprétation

Dans un **benchmark**, on compile à la fois (1) le temps CPU total, et (2) le nombre d’itérations. Il peut arriver qu’un algorithme effectue plus d’itérations mais chaque itération soit plus légère (sparse?), ou inversement. Le **produit** $O(n^2) \times N_{\text{itér}}$ constitue un ordre de grandeur pour la **complexité** globale.

Dans des applications temps réel ou en flux (chap. 9), on ne peut pas laisser le DSL tourner indéfiniment. On fixe un **budget** maximal d’itérations ou un **budget** de temps. Un algorithme plus gourmand en itérations risque de s’arrêter prématurément, en-dessous d’une convergence stable.

Si une **version** DSL, plus globale (recuit, inhibition adaptative), accroît la probabilité d’atteindre un minimum global meilleur, elle peut demander plus d’itérations (ou plus de CPU) pour y parvenir. Selon l’objectif (précision vs. temps), on peut préférer la méthode plus simple ou la méthode plus globale.

Conclusion

Le **temps CPU** et le **nombre d’itérations** jusqu’à la **stabilisation** constituent deux **indicateurs** essentiels pour analyser la **vitesse** et le coût d’un algorithme DSL. Ils déterminent la **praticabilité** d’une solution (peut-on se permettre 10 000 itérations ? combien de minutes CPU cela représente-t-il ?) et l’adéquation à un contexte temps réel ou batch :

- **Temps CPU** : dépend de la taille n et du nombre de liens (jusqu’à $O(n^2)$), ainsi que du coût supplémentaire des heuristiques (recuit, inhibition variable).
- **Itérations jusqu’à ϵ -stabilité** : un critère direct sur la **vitesse** de convergence, souvent plus abstrait mais pratique à suivre dans un code expérimental.

En comparant différentes versions (ex. DSL classique vs. DSL + recuit), on surveille à la fois la **qualité** finale (énergie, modularité) et la **vitesse** (nombre d’itérations, temps CPU). Il en découle un compromis entre **rapidité** et **probabilité** de sortir d’un puits local : une méthode plus globale et plus lente peut fournir de meilleurs résultats, mais au prix d’un nombre d’itérations supérieur.

7.8.3.3. Outils de Visualisation (Carte de Chaleur, Courbes $\omega_{i,j}(t)$)

Pour **analyser** la dynamique d’un SCN (Synergistic Connection Network), on ne se contente pas toujours des seules valeurs numériques (énergie finale, modularité, etc.) : on peut vouloir **visualiser** l’évolution des pondérations $\omega_{i,j}$ au fil des itérations, ou se faire une idée de la structure finale (réseau, clusters). Les **outils** de visualisation aident à comprendre :

- Comment se **répartissent** les valeurs $\omega_{i,j}$ dans l’espace des entités,
- Comment $\omega_{i,j}(t)$ **varie** dans le temps (croissance, décroissance, saturation),
- Où se **regroupent** les entités (clusters).

Deux approches de base se distinguent souvent : la **carte de chaleur** (heatmap) et les **courbes** temporelles des liens.

A. Carte de Chaleur (Heatmap)

La **carte de chaleur** (heatmap) consiste à représenter la matrice $\{\omega_{i,j}\}$ sous forme d’une **image**, chaque case (i,j) est colorée selon la valeur de $\omega_{i,j}$. Lorsque le réseau se stabilise, la heatmap donne un aperçu direct des **régions** (ou blocs) où les pondérations sont fortes, et des zones quasi nulles.

Visualiser ω en “carte de chaleur” permet de **repérer** facilement :

- Des **groupes** : s’il y a un bloc diagonal (ou un bloc “hors-diagonal”) de couleurs vives (valeurs élevées), cela signale un **cluster** cohérent.

- Des liens parasites : si certains $\omega_{i,j}$ sont haut perchés en dehors du bloc principal, on identifie des connexions inattendues ou transversales.
- La **progression** : en affichant la heatmap à différentes itérations ($t=0, t=10, t=50\dots$), on voit comment la matrice se densifie (certains blocs deviennent colorés), ou au contraire s'éclaircit quand la compétition ou le recuit modifie les liens.

On définit une **échelle** de couleur (ex. du bleu foncé pour $\omega \approx 0$ au rouge pour ω maximal). Si on introduit un ω_{\max} en saturation, on veille à “clipper” les valeurs supérieures. On peut opter pour un tri des entités (ordre des lignes/colonnes) en fonction d'un clustering préalable afin de faire ressortir plus clairement les blocs.

B. Courbes $\omega_{i,j}(t)$

Une autre façon de **suivre** le réseau DSL est de sélectionner quelques paires (i, j) (celles qui représentent un cluster important, ou un lien inter-groupe) et de **tracer**, au fil du temps t , la courbe $\omega_{i,j}(t)$. On peut aussi tracer le “degré” d'un nœud $\sum_j \omega_{i,j}(t)$. Ainsi, on voit si un lien s'élève rapidement, puis se tasse, ou s'il oscille avant de se stabiliser.

Cela aide à **repérer** :

- Des **oscillations** : signaux d'un paramétrage η, τ mal réglé ou d'un γ trop faible (inhibition insuffisante).
- Des **verrous** : si un lien censé être fort stagne vers 0, c'est qu'il subit un amortissement ou qu'il existe une compétition intense avec d'autres liens.
- La **phase** de recuit : on observe la variabilité plus élevée de ω au début, décroissant à mesure que la température baisse.

En couplant les **courbes** $\omega_{i,j}(t)$ à la **carte de chaleur**, on obtient une vision micro (quelques liens clés) et macro (la structure complète du réseau). Ce double point de vue est souvent essentiel à la **compréhension** fine de la dynamique DSL.

C. Autres Visualisations

Si l'on dispose d'un embedding associant chaque entité à un point (ex. PCA ou t-SNE), on peut **montrer** la force des liens $\omega_{i,j}$ comme des arêtes plus ou moins épaisses. On voit ainsi la formation de **clusters** dans l'espace projeté.

Tracer la courbe $J(\omega(t))$ permet de **suivre** la descente d'énergie, utile pour détecter plateaux, minima locaux ou transitions abruptes.

Conclusion

La **carte de chaleur** (heatmap) et les **courbes** $\omega_{i,j}(t)$ constituent deux **outils** de visualisation très utilisés pour **analyser** la dynamique DSL :

- La **heatmap** dresse un “portrait” global de la matrice ω , mettant en évidence les **blocs** ou motifs (clusters, liens parasites).
- Les **courbes** temporelles $\omega_{i,j}(t)$ ou $\sum_j \omega_{i,j}(t)$ montrent **comment** chaque lien (ou chaque nœud) évolue dans le temps, révélant parfois oscillations, instabilités ou phases de stabilisation.

Ces visualisations complètent les **métriques** (énergie, modularité, etc.) en fournissant un **résumé** intuitif et graphique du **réseau** qui se forme sous la dynamique DSL. Elles sont fréquemment mises en œuvre dans les expérimentations pour **comprendre** et **expliquer** le comportement d'un SCN face aux choix de paramètres (η, τ, γ , recuit, etc.).

7.9. Études de Cas

Dans cette section 7.9, nous passons des **principes** et **méthodes** (recuit simulé, heuristiques globales, inhibition avancée, etc.) à des **illustrations concrètes**. L'idée est de valider expérimentalement la dynamique d'**optimisation** et d'**adaptation** dans des scénarios plus ou moins réalistes, pour vérifier :

- La **capacité** du SCN (Synergistic Connection Network) à échapper à des configurations sous-optimales,
- L'**efficacité** des méthodes introduites (recuit, heuristiques) face à différents minima locaux,
- L'**impact** sur la formation de clusters, la rapidité de convergence, et la qualité globale.

7.9.1. Cas de Simulation Numérique

Afin de mieux contrôler les paramètres et de comprendre les phénomènes à l'œuvre, on commence par un **cas** relativement simple : un **petit réseau** (10 à 20 entités) conçu de façon à **présenter 2 ou 3 minima locaux connus**. Cette configuration, bien que réduite, nous permet d'observer, en laboratoire, la **dynamique DSL** et l'effet de nos algorithmes d'optimisation (recuit, heuristiques, etc.).

7.9.1.1. Petit Réseau de 10–20 Entités avec 2–3 Minima Locaux

Il est souvent **instructif**, pour tester un algorithme DSL (Deep Synergy Learning) et ses variantes (recuit, heuristiques globales, etc.), de se placer sur un **réseau** de taille modeste (d'une dizaine à une vingtaine d'entités). En outre, on peut **fabriquer** ou **configurer** la synergie $S(i, j)$ afin de mettre en évidence **plusieurs** minima locaux, illustrant la **difficulté** d'échapper à certains puits et la nécessité d'heuristiques ou de recuit.

A. Configuration du Réseau

On sélectionne un petit $n \in [10, 20]$. Chaque entité \mathcal{E}_i peut être associée à un **vecteur** \mathbf{x}_i de faible dimension (2D, 3D) ou à un label, de manière à **calculer** la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$. On veille à créer une répartition qui admette plusieurs arrangements de clusters.

Pour garantir l'existence de **plusieurs attracteurs** dans l'évolution du **SCN**, on peut structurer l'espace des connexions en définissant trois **sous-groupes** distincts A , B et C , dont la **cohérence interne** est forte. Afin de permettre l'émergence de plusieurs partitions viables, des **liaisons inter-groupes de force intermédiaire** sont ajoutées, rendant possible l'apparition de configurations alternatives.

L'initialisation des **pondérations** $\omega_{i,j}(0)$ est effectuée avec des valeurs faibles ou aléatoires sur un intervalle $[-\epsilon, \epsilon]$, afin de ne pas induire d'organisation pré-déterminée. La **dynamique DSL** est ensuite exécutée avec ou sans **recuit simulé** ou heuristique d'optimisation, et la règle d'évolution des poids suit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ est un bruit stochastique contrôlé dans le cadre du **recuit simulé**.

L'expérience est répétée sur **20 exécutions** ou plus afin d'analyser la répartition des convergences et d'évaluer **quelle proportion de runs atteint chaque attracteur**, permettant ainsi d'identifier les **configurations dominantes** au sein du SCN.

B. Exemples de Minima Locaux

Dans un SCN, plusieurs **configurations d'équilibre** peuvent exister, chacune constituant un **minimum local** où la dynamique DSL tend naturellement à converger. Ces **minima** correspondent à différentes manières de structurer le réseau en clusters, chacune ayant une **énergie associée** notée $J_{\text{mathcal}}\{J\}$.

Un premier clusterement, noté $\Omega^{(1)}$, regroupe par exemple les entités $\{\mathcal{E}_1, \dots, \mathcal{E}_k\}$ dans un groupe A , et $\{\mathcal{E}_{k+1}, \dots\}$ dans un groupe B , formant ainsi une partition stable du réseau avec une énergie associée $J^{(1)}$.

Un second clusterement, $\Omega^{(2)}$, peut être obtenu en **modifiant légèrement la structure** du réseau, par exemple en déplaçant deux entités dans un autre groupe. Ce nouvel état représente une **variation mineure**, mais qui peut suffire à influencer la dynamique d'évolution du SCN. Son énergie $J^{(2)}$ est souvent proche de $J^{(1)}$, voire légèrement inférieure si la réorganisation optimise les synergies.

Un troisième clusterement, $\Omega^{(3)}$, peut apparaître en introduisant une configuration encore différente, où la structure des liens et des groupes est légèrement modifiée. Cet état peut être plus **élevé en énergie** ou comparable aux deux premiers, rendant la descente DSL plus complexe en raison de la présence de **plusieurs puits énergétiques similaires**.

La présence de **plusieurs minima locaux** complique l'évolution du SCN, car sans heuristique additionnelle comme le **recuit simulé**, la dynamique DSL risque de rester bloquée dans un état sous-optimal, empêchant l'atteinte d'une **solution globalement plus synergétique**.

C. Indicateurs et Mesures

L'évaluation des différentes configurations atteintes par la dynamique DSL repose sur plusieurs **indicateurs quantitatifs** permettant de caractériser la convergence et la qualité des solutions obtenues.

L'**énergie finale** $J(\Omega^*)$ est l'un des principaux critères permettant de juger l'optimalité d'un état atteint. Elle peut être définie par la fonction J :

$$J(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_i \omega_{i,j}^2.$$

Lorsque plusieurs exécutions de la dynamique aboutissent à des valeurs similaires $J^{(1)}, J^{(2)}, J^{(3)}$, cela indique l'**existence de plusieurs attracteurs** énergétiques où le SCN peut se stabiliser.

Le **temps de convergence** est un autre indicateur important, mesurant le nombre d'**itérations** nécessaires avant que la norme de variation des pondérations $\|\omega(t+1) - \omega(t)\|$ atteigne un seuil δ . Cette mesure permet de comparer la rapidité des différentes approches : **DSL seule**, **DSL avec recuit simulé** et **DSL avec heuristiques additionnelles**.

Enfin, la **cohésion des clusters** est étudiée à l'aide d'indicateurs structurels tels que la **modularité du réseau** ou la **densité intra-cluster**. Ces analyses permettent de déterminer si la partition obtenue est **cohérente et lisible** ou si l'arrangement final des connexions demeure **confus et mal structuré**.

D. Observations

L'analyse des résultats obtenus dans différentes conditions révèle plusieurs comportements typiques du SCN.

Lorsque la dynamique DSL est appliquée **sans recuit ni heuristique additionnelle**, les simulations montrent une **convergence rapide**, en un nombre d'itérations relativement faible. Toutefois, cette descente rapide conduit fréquemment à un **minimum local**, tel que $\Omega^{(1)}$ ou $\Omega^{(2)}$, selon l'**initialisation aléatoire** du réseau. Le SCN se retrouve ainsi **piégé** dans une structure stable mais sous-optimale en énergie, rendant difficile la transition vers un état plus synergétique.

L'ajout d'un **recuit simulé** modéré au début de l'apprentissage permet d'autoriser l'**exploration de plusieurs structures alternatives**. Dans ce cas, certaines trajectoires dynamiques montrent un **basculement** d'un attracteur local $\Omega^{(1)}$ vers un état $\Omega^{(2)}$ présentant une **énergie plus faible**. Toutefois, cette approche a un **coût computationnel**, car la

durée nécessaire pour atteindre la stabilisation finale peut s'accroître en raison de la perturbation stochastique introduite.

Une alternative consiste à **multiplier les exécutions** de la dynamique DSL et à sélectionner, parmi plusieurs runs, la configuration Ω aboutissant à la plus faible énergie finale. Cette approche de type **sélection multiple** peut augmenter significativement la probabilité de convergence vers $\Omega^{(2)}$ ou $\Omega^{(3)}$, bien qu'elle implique un **surcoût global** lié à la répétition des exécutions.

Conclusion

Un **petit** réseau (10–20 entités) doté de **2–3 minima** locaux constitue un **terrain** d'expérimentation :

- **Vérifier** l'enfermement local avec la descente DSL basique,
- **Évaluer** l'apport du recuit ou d'heuristiques globales,
- **Quantifier** via l'énergie finale, le temps de convergence, la cohésion de clusters,
- **Observer** comment un algorithme peut (ou non) franchir la barrière d'énergie menant à un arrangement plus “globalement” optimal.

Ce **laboratoire** de petite taille facilite l'interprétation : on voit clairement si le clusterement final coïncide avec $\Omega^{(1)}$ (un arrangement local) ou $\Omega^{(2)}$ (plus global). On peut alors ajuster les **paramètres** DSL (taux η , décroissance τ , inhibition γ , planning de température) pour accroître la probabilité de trouver la “bonne” partition.

7.9.1.2. Comparaison Recuit vs. Heuristique vs. Basic DSL

Le **Deep Synergy Learning (DSL)** peut se décliner en plusieurs **variantes** lorsqu'il s'agit d'échapper à des minima locaux ou de chercher une configuration de réseaux plus globale. Il est ainsi naturel de mettre en parallèle :

- une **version basique** (sans recuit ni heuristique),
- une **version recuit simulé** (avec injection de bruit et planning de température),
- une **version heuristique globale** (par exemple un petit algorithme génétique appliqué à ω , ou un “shake” ponctuel, ou encore un protocole multi-run).

Cette comparaison se fait généralement sur un **même** problème de référence (voir § 7.9.1.1 pour un réseau de 10–20 entités et 2–3 minima locaux). Le but est de mesurer la **qualité** de la solution finale, le **temps** requis, la robustesse face à l'initialisation, et la **probabilité** d'atteindre un arrangement d'énergie plus faible.

A. Approche Basic DSL

La version dite “classique” du DSL met à jour les pondérations $\omega_{i,j}$ uniquement via des règles locales :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Cette formule converge souvent **rapidement**, et la compétition ou l'inhibition peuvent être présents mais de manière fixe ou modérée. Le principal inconvénient est la **tendance** à rester piégé dans un minimum local : dès que la descente locale s'approche d'un puits, rien ne pousse réellement le réseau à en sortir. Dans des tests comportant 2–3 minima d'énergie similaire, la simple descente DSL aboutit fréquemment à des configurations sub-optimales, même si, au regard du **temps de convergence**, elle s'avère la plus rapide.

B. Approche DSL + Recuit Simulé

Dans cette version, on ajoute un **terme stochastique** $\xi_{i,j}(t)$ dont l'amplitude est contrôlée par une **température** $T(t)$. Il se produit alors deux étapes majeures dans la dynamique :

– une **phase chaude**, où le bruit est assez fort pour permettre de franchir les barrières d'énergie et donc de sortir des vallées locales,

– une **phase de refroidissement**, où le bruit décroît, stabilisant la configuration dans une région d'énergie plus basse.

Ce recuit simulé requiert un **calibrage** délicat du planning de température (cf. 7.3.1.3). S'il est bien paramétré, on obtient souvent une **amélioration** notable de la solution finale (énergie plus basse, clustering plus cohérent), au prix d'un **plus grand** nombre d'itérations et d'un coût CPU plus élevé, car la phase chaude nécessite du temps pour que la dynamique explore suffisamment et la phase froide ne peut pas être trop abrégée.

C. Approche DSL + Heuristique Globale

Plutôt que de moduler la température, on peut se tourner vers des **heuristiques** plus globales ou plus disruptives. Un petit algorithme génétique, par exemple, va considérer plusieurs configurations ω à la fois, les “croiser” ou les “muter”, cherchant une solution combinatoire plus “haute”. On peut aussi introduire un “shake” régulier, c'est-à-dire réinjecter, toutes les 20 itérations, un bruit fort sur un sous-ensemble de liens ou perturber la matrice ω pour sortir d'un attracteur local.

De telles stratégies peuvent conduire à des **solutions** de très bonne qualité, parfois au-dessus même de ce que permet le recuit, mais leur **mise en œuvre** est souvent plus complexe et leur **coût** potentiellement plus élevé. Les heuristiques globales exigent en effet de multiples évaluations de la fonction d'énergie ou une gestion de populations (dans le cas d'un GA). On peut alors gagner en robustesse, en diminuant la dépendance à l'initialisation, mais la convergence finale peut demander encore plus de temps si on n'opère pas avec parcimonie.

D. Observations Typiques

Dans la majorité des scénarios, la **version** DSL basique atteint assez vite un **minimum local**, s'y fige et donne un niveau d'énergie intermédiaire. Les versions plus avancées (recuit, heuristique) consacrent plus d'itérations et de CPU à l'exploration, échappent aux barrières locales et finissent avec une énergie plus basse. Le paramétrage (planning de température, intensité des shakes, population de l'algorithme génétique) influe beaucoup sur la performance.

Pour un **petit réseau** de 10–20 entités, on peut observer des chiffres concrets :

- la descente simple converge en 150–200 itérations,
- le recuit peut en demander 300–400 pour se “refroidir”,
- l'heuristique globale (ex. GA) peut nécessiter 10 ou 20 générations, chacune traitant plusieurs configurations ω . Cependant, la probabilité d'aboutir à la configuration d'énergie la plus basse grimpe sensiblement avec recuit ou heuristiques (ex. 20 % pour la descente pure vs. 70–80 % pour le recuit, selon la complexité du paysage énergétique).

Conclusion

En **comparant** ces trois versions (DSL basique, DSL + recuit, DSL + heuristique globale), on distingue généralement un **choix** entre la **simplicité / rapidité** (la descente simple) et la **qualité** finale (recuit ou heuristiques globales). Le recuit, s'il est bien paramétré, constitue un **ajout** modéré à la descente DSL, conférant une exploration plus vaste. Les heuristiques globales (algorithmes génétiques, par exemple) peuvent encore mieux **couvrir** l'espace des configurations $\{\omega\}$, mais leur coût et leur mise au point sont plus élevés. Chacune de ces approches trouve sa place selon la taille du problème, la sévérité des minima locaux, et les **ressources** en temps de calcul.

7.9.1.3. Graphiques et Discussion

Lorsqu'on compare différentes variantes du **DSL** (Deep Synergy Learning) — par exemple, une version “basique”, une version “recuit simulé”, une version “heuristique globale” — il est souvent essentiel de **visualiser** les résultats et l'évolution de la dynamique, au-delà de simples chiffres (énergie finale, modularité). Les **graphiques** suivants sont très utiles pour **discuter** et **interpréter** les phénomènes :

Courbe de l'Énergie J au fil du temps

Heatmap ou **matrice** des pondérations $\omega_{i,j}$,

Clusterisation ou **réseau** final (avec éventuellement un affichage 2D/3D).

Ces représentations éclairent la **convergence**, la **structure** des clusters, et la **capacité** de l'algorithme à franchir ou non un minimum local.

A. Représentation de l'Énergie \mathcal{J} au fil des Itérations

L'évolution de l'énergie \mathcal{J} au cours du temps constitue un indicateur essentiel permettant d'analyser la **dynamique de convergence** des différentes variantes de l'algorithme DSL. Pour chaque version du modèle, incluant la **DSL basique**, la **DSL avec recuit simulé** et la **DSL couplée à une heuristique d'optimisation**, on peut représenter la fonction $\mathcal{J}(\Omega(t))$ en fonction du nombre d'**itérations** t .

Le tracé standard d'une telle courbe adopte un **axe des abscisses** représentant l'itération t , tandis que l'**axe des ordonnées** affiche la valeur de l'énergie $\mathcal{J}(t)$, soit sous sa **forme brute**, soit sous une **normalisation** facilitant la comparaison entre différentes configurations du SCN.

L'analyse de ces courbes met en évidence des **profils caractéristiques** selon la version de l'algorithme utilisée. Dans le cas de la **DSL basique**, la courbe d'énergie montre une **descente rapide** suivie d'une **stabilisation précoce** autour d'une valeur $\mathcal{J}^{(1)}$, correspondant à un **minimum local** dont le SCN ne parvient pas à s'extraire. Lorsque le **recuit simulé** est introduit, la courbe présente des **oscillations initiales** correspondant aux perturbations thermiques du système (phase chaude). Il est courant d'observer une **légère remontée temporaire** de \mathcal{J} , signe d'une exploration active, suivie d'une **descente progressive** vers un état plus optimal $\mathcal{J}^{(2)}$, inférieur à $\mathcal{J}^{(1)}$.

Lorsque des **heuristiques globales** sont mises en place, comme des méthodes inspirées de l'**optimisation génétique**, la courbe d'évolution de l'énergie adopte un **profil plus irrégulier**, avec des variations en **dents de scie** dues aux ajustements de sélection et de mutation. Cependant, ces approches peuvent aboutir à un **niveau d'énergie plus faible**, prouvant leur efficacité dans l'**éviter des minima locaux**.

L'interprétation des courbes permet de comparer les différentes variantes de l'algorithme. Si une version présente une stabilisation précoce de $\mathcal{J}(t)$ à une **valeur élevée**, cela indique qu'elle reste **bloquée dans un minimum sous-optimal**. En revanche, un algorithme capable de **réduire** \mathcal{J} de manière significative au fil des itérations, ou de **repousser** cette stabilisation à un temps plus tardif, démontre une **meilleure capacité d'exploration** et une **réduction efficace des barrières énergétiques**.

B. Visualisation de la Matrice ω (Heatmap)

La **matrice des pondérations** $\omega_{i,j}$ peut être représentée sous la forme d'une **carte de chaleur** (heatmap), où chaque cellule (i, j) est colorée en fonction de la valeur de $\omega_{i,j}$. Cette visualisation permet de suivre l'évolution des connexions du SCN et d'identifier les **zones de synergie** qui se forment au fil des itérations.

En affichant plusieurs **snapshots temporels** (ex. $t = 0, 50, 100, \dots$), on met en évidence la manière dont les blocs de **connexions fortes** se structurent et comment certaines **liaisons disparaissent progressivement**. Un SCN bien organisé généralement des **clusters distincts** sous la forme de **blocs carrés** bien marqués sur la diagonale, tandis que les liens faibles ou parasites s'effacent sous l'effet de l'inhibition ou du recuit simulé.

La **comparaison entre DSL basique et DSL avec recuit** permet d'observer des différences notables. Dans la version classique, certains liens **parasites inter-clusters** peuvent subsister, rendant la structure du réseau plus floue. Avec un recuit bien paramétré, on obtient un **contraste plus marqué** entre les liaisons fortes (zones de synergie renforcées) et les liaisons faibles (disparition progressive des connexions inutiles), ce qui traduit une meilleure organisation des groupes coopératifs.

C. Clusterisation et Réseau Final

Au-delà de la heatmap, la structure finale du SCN peut être représentée sous la forme d'un **graphe pondéré**, où les **nœuds** correspondent aux entités et les **arêtes** sont pondérées par la valeur de $\omega_{i,j}$. En appliquant un **seuil de filtrage**

sur les connexions faibles ($\omega_{i,j} < \theta$), on obtient une **vision épurée** du réseau, mettant en évidence les **sous-groupes fortement connectés** qui émergent naturellement.

Dans le cas d'un **DSL basique**, la structure obtenue peut contenir un plus grand nombre de **liaisons moyennes** qui brouillent la distinction entre clusters. En revanche, avec un **recuit simulé** ou une **heuristique d'optimisation**, la séparation entre groupes est **plus nette**, avec des **connexions intra-cluster renforcées** et une élimination efficace des liaisons inter-groupes non pertinentes. Ce type d'analyse permet aussi de vérifier si un **cluster alternatif** (comme $\Omega^{(2)}$ au lieu de $\Omega^{(1)}$) a pu émerger sous l'effet des heuristiques utilisées.

D. Discussion

L'analyse des courbes d'énergie permet souvent de mettre en évidence un **croisement des dynamiques**. Dans un scénario typique, la courbe $J(t)$ du **DSL basique** affiche une **descente rapide** puis une stabilisation précoce, illustrant l'enfermement dans un **minimum local**. À l'inverse, un **recuit simulé** ou une **heuristique plus aggressive** prolonge la phase d'exploration, permettant parfois de **franchir une barrière énergétique** et d'atteindre un niveau d'énergie plus bas.

Cette amélioration se traduit concrètement par une **meilleure organisation des clusters**, ce qui est visible à travers les heatmaps et les graphes obtenus. Une meilleure séparation des sous-groupes, avec des connexions bien définies et une **élimination efficace des liens parasites**, démontre la capacité du recuit ou de l'heuristique utilisée à structurer le SCN de manière plus optimale.

Ces visualisations permettent également d'affiner le **calibrage des paramètres**, notamment en ajustant la **température du recuit**, la **fréquence des perturbations stochastiques**, ou encore l'**intensité des règles d'inhibition**. Si les **clusters finaux restent flous**, cela peut signifier que la température est restée trop élevée, ou que le filtrage des liaisons n'a pas été suffisamment strict. À l'inverse, une segmentation trop brutale peut être le signe d'un recuit trop rapide ou d'une heuristique trop contraignante, empêchant certaines connexions pertinentes de se stabiliser.

En combinant ces analyses graphiques avec les **mesures quantitatives d'énergie et de convergence**, on obtient ainsi une compréhension plus approfondie de la dynamique du SCN et de la manière dont les différentes méthodes influencent la qualité des regroupements finaux.

Conclusion

Les **graphes** d'évolution de J et les **représentations** (heatmap, graphe) de la structure ω constituent des **instruments** majeurs pour interpréter et **discuter** les résultats comparatifs (DSL basique vs. recuit vs. heuristique). Ils révèlent :

- **La vitesse** à laquelle chaque méthode baisse J (plateaux, stagnations),
- **La structure** de clusters plus ou moins marquée (via heatmap ou graphes filtrés),
- **La dynamique** : si le recuit "rebondit" pour franchir un puits, si l'heuristique restructure le réseau soudainement, etc.

Au final, ces visualisations éclairent l'impact qualitatif et quantitatif des différentes approches, facilitant ainsi la **discussion** et la **conclusion** quant à la pertinence des paramètres (température, heuristique) et à la performance globale de chaque variante DSL.

7.9.2. Mise en Application Robotique ou Multi-Agent

Les principes d'**optimisation** et d'**adaptation** présentés dans les sections précédentes (recuit, heuristiques, inhibition avancée) trouvent une application directe dans les scénarios **robotique** et **multi-agent**. En effet, un essaim de robots, ou un ensemble d'agents coopératifs, peut faire face à des **défis** de configuration et de répartition des ressources, où la **dynamique** DSL (Deep Synergy Learning) doit se réorganiser en temps réel. Avant de plonger dans les détails (7.9.2.2, 7.9.2.3), rappelons la **configuration** générale de l'essaim de robots décrite en chapitres 2.5.3 et 4.7, et comment elle s'intègre dans la logique du SCN.

7.9.2.1. Rappel Essaim de Robots (Chap. 2.5.3, 4.7)

L'**essaim** de robots constitue un champ d'application privilégié pour la dynamique **DSL** (Deep Synergy Learning), déjà évoqué dans plusieurs sections antérieures (en particulier Chap. 2.5.3 à propos de la synergie en robotique, et Chap. 4.7 dans le cadre d'exemples concrets). L'idée sous-jacente est de considérer un ensemble de robots coopérant pour une mission donnée, et de leur permettre de **reconfigurer** leurs liens de communication ou de collaboration $\{\omega_{i,j}\}$ au cours du temps de manière **auto-organisée**. Les robots sont modélisés comme des **nœuds** d'un SCN (Synergistic Connection Network), et la dynamique DSL offre un cadre pour renforcer ou affaiblir les liaisons selon la **synergie** perçue $S(R_i, R_j)$. Les sections qui suivent (7.9.2.2, 7.9.2.3) exploiteront les principes d'optimisation (recuit, heuristiques) à ce contexte robotique.

A. Configuration Générale d'un Essaim

Un **essaim** (ou **swarm**) regroupe un ensemble $\{R_1, \dots, R_N\}$ de robots, souvent de conception modulaire et capables de se disperser pour couvrir un terrain ou de se regrouper pour réaliser une action collective. Chaque robot R_i embarque un ensemble de **capteurs** (caméra, LIDAR, ultrasons, GPS, etc.), des **actionneurs** (roues, moteurs articulés, bras, etc.) et une **interface** de communication (radio, wifi, Bluetooth) pour échanger des données avec d'autres robots à proximité. Les **missions** associées à l'essaim couvrent de multiples domaines : exploration (cartographie d'une zone inconnue), surveillance (détection d'intrus), transport en convoi, formation (réaliser une figure géométrique), etc.

Dans ce cadre, la **synergie** $S(R_i, R_j)$ exprime la **compatibilité** ou la **complémentarité** entre deux robots. Cette complémentarité peut s'appuyer sur la **distance** (deux robots proches peuvent s'entraider plus facilement) ou sur les **rôles** (un robot de type "capteur" et un robot de type "actionneur" forment un binôme productif). Il peut aussi s'agir de corrélation entre signaux, d'une co-activation dans l'exécution de la tâche ou de la capacité à relayer les informations. L'apport fondamental du **DSL** est de maintenir, en **temps réel**, une **matrice** $\omega(t)$ où $\omega_{i,j}(t)$ détermine la force de la liaison entre robots R_i et R_j . Quand deux robots collaborent efficacement, $\omega_{i,j}$ s'accroît ; s'ils se gênent mutuellement ou n'interagissent pas, la pondération se réduit.

B. SCN pour l'Essaim de Robots

Les **chapitres** 2.5.3 et 4.7 ont montré l'intérêt du **SCN** pour modéliser l'**évolution** des liens de collaboration dans un essaim robotique. Chaque robot R_i peut constituer un **nœud** dans la matrice ω . On applique ainsi la règle **DSL** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(R_i, R_j) - \tau \omega_{i,j}(t)] + \Delta_{(\text{inhibition ou bruit})}$$

où η correspond au taux d'apprentissage, τ reflète la décroissance basique, et $\Delta_{...}$ inclut éventuellement l'**inhibition** (chap. 7.4) ou le **bruit** (recuit, chap. 7.3).

L'intérêt majeur réside dans l'**adaptation** continue : si l'environnement change ou si la mission se modifie (passer d'une phase d'exploration à une phase de rendez-vous centralisé), les **mesures** $S(R_i, R_j)$ se recalculent (les robots perçoivent différemment leur synergie), et la **dynamique** $\omega_{i,j}(t)$ s'actualise en conséquence. Chaque robot i "ressent" localement l'efficacité (ou inefficacité) de sa relation avec j et adapte sa pondération $\omega_{i,j}$.

Cette mise à jour locale peut cependant conduire à des **minima locaux** dans la configuration de l'essaim : un sous-groupe s'organise pour l'ancienne mission et tarde à se reconfigurer, ou un robot reste isolé, alors qu'une distribution plus globale de tâches pourrait améliorer la performance générale. C'est à ce **problème** que les méthodes d'optimisation (recuit, heuristiques globales) apportent une solution, en autorisant des **sauts** ou des **disruptions** dans la dynamique DSL.

C. Défis d'Optimisation dans l'Essaim

L'**espace** des configurations $\{\omega_{i,j}\}$ étant rapidement énorme pour un nombre N de robots, la **descente** locale du **DSL** peut se figer, surtout si ω est mis à jour de façon déterministe et monotone. Pour un **grand** essaim, on se heurte de plus à la **complexité** $O(N^2)$ (chap. 7.2.3). On cherche alors à incorporer des **modifications** comme la **sparsification** (ex. chaque robot ne gère que k voisins) et des **mécanismes** plus globaux :

- **Recuit simulé** : on injecte un bruit contrôlé par une température $T(t)$ permettant de casser une organisation figée et d'explorer d'autres topologies.
- **Heuristiques** globales (ex. un micro-algorithme génétique sur la matrice ω), combinant ou mutant diverses configurations.
- **Inhibition dynamique** : on fait varier $\gamma(t)$ pour accroître la compétition quand la densité de liens devient trop forte, ou la diminuer si le réseau se retrouve fragmenté.

Ces approches **enrichissent** la logique DSL de base, permettant à l'essaim de quitter des arrangements que l'on soupçonne sous-optimaux (difficulté à inclure un robot tardivement arrivé, ou inefficacité d'une répartition de rôles qui persiste sans raison).

D. Exemple Mathématique Minimal

Pour illustrer ce qui précède, imaginons un **essaim** de 5 robots $\{R_1, \dots, R_5\}$. La **synergie** $S(R_i, R_j)$ dépend par exemple de la **distance** spatiale (deux robots proches obtiennent un S élevé) ou de la **complémentarité** de rôles (un robot capteur, un robot actionneur). La dynamique DSL se traduit par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(R_i, R_j) - \tau \omega_{i,j}(t)].$$

Si quatre robots se rapprochent, isolant le cinquième, la descente locale peut s'en tenir à cette configuration stable, même si le cinquième robot propose un capteur-clé pour la mission. C'est dans ce cas qu'un **recuit** modéré ou un **shake** heuristique, en rehaussant ponctuellement des liaisons "en sommeil", peut modifier la "topologie" ω et amener l'essaim à réintégrer le robot isolé dans une structure plus performante.

Conclusion

La **notion** d'essaim robotique (référence Chap. 2.5.3 et 4.7) illustre parfaitement l'**intérêt** du DSL : un ensemble de robots, chacun équipé de capteurs et d'actionneurs, se coordonnent via un SCN dont les pondérations $\omega_{i,j}$ varient en fonction de la **synergie** $S(R_i, R_j)$. Le **défi** réside dans le fait que la descente locale du DSL peut se figer dans un **minimum** local, ignorant des configurations plus globalement optimales. Les **outils** d'optimisation (recuit, heuristiques, inhibition dynamique) détaillés en chapitre 7 permettent de dépasser cette limite, en **améliorant** la capacité de l'essaim à se **reconfigurer** quand la mission ou le contexte évolue, et en **maximisant** la performance de la collaboration multi-robots. Les sections suivantes (7.9.2.2, 7.9.2.3) approfondiront ces approches et **discuteront** des résultats de simulations en environnement robotique.

7.9.2.2. Optimisation Adaptative : Recuit pour Reconfigurer l'Essaim, ou Inhibition Avancée Modulée

Dans des scénarios de **robotique multi-agent**, la capacité d'un **essaim** à se **reconfigurer** rapidement et efficacement est fondamentale : l'environnement évolue, la mission change, et le SCN (Synergistic Connection Network) doit ajuster ses pondérations ω pour maintenir une **coopération** optimale. Les méthodes d'**optimisation adaptative** décrites dans ce chapitre (recuit simulé, inhibition modulée) offrent des moyens de sortir d'une configuration figée, de redistribuer les rôles et de reconstituer des **clusters** plus performants.

A. Recuit Simulé pour Reconfigurer l'Essaim

Les **robots** de l'essaim échangent des informations ou des tâches via des liens $\omega_{i,j}$. Au fil du temps, la descente locale du DSL peut se **bloquer** dans un minimum local, où certains robots se sont rassemblés en un cluster moins optimal que d'autres configurations possibles. Injecter un **terme** aléatoire $\sigma(t) \xi_{i,j}(t)$ (cf. chap. 7.3) dans la mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t)$$

apporte la **possibilité** de rompre la structure locale et de "sauter" vers d'autres vallées d'énergie.

Chaque robot R_i forme ou défait des liens $\omega_{i,j}$ avec ses voisins. En l'absence de bruit, l'ajustement suit la **descente** de la fonction $-\sum \omega_{i,j} S(i,j) + \dots$. Avec du **bruit** modulé par une température $\sigma(t)$, on peut sortir d'une organisation trop stable (mais sous-optimale) pour en rejoindre une autre plus pertinente à la mission (déclenchement d'une **phase** de reconfiguration).

Si, par exemple, deux robots se trouvent isolés parce que leur liaison $\omega_{i,j}$ n'a jamais pu surmonter la compétition d'autres liens, un bruit aléatoire pourrait rehausser $\omega_{i,j}$ lors d'un tirage. Si la synergie $S(i,j)$ est bonne, ce nouveau lien se consolide et **réorganise** l'essaim.

Le **schéma** de décroissance $\sigma(t)$ (exponentiel, logarithmique, etc.) décide de la période de "phase chaude" (exploration). Trop bref, le recuit n'exploré pas assez ; trop long, la convergence se prolonge. Dans l'essaim robotique, on peut aussi lancer un **recuit ponctuel** lors de changements majeurs (robot tombé en panne, mission modifiée).

B. Inhibition Avancée Modulée pour la Coordination

L'**inhibition dynamique** constitue une alternative ou un complément au **recuit simulé** dans la gestion de la **compétition** entre liens au sein du SCN. Cette méthode repose sur la **réduction progressive des connexions faibles**, incitant chaque **robot** à focaliser ses interactions sur un **ensemble réduit de partenaires** présentant la meilleure synergie. Dans ce cadre, la **pénalisation** des liaisons excessives est introduite sous la forme d'un **terme d'inhibition** appliqué à la mise à jour des pondérations :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau\omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Ce terme d'inhibition agit comme un **mécanisme de régulation** empêchant un robot i de maintenir un trop grand nombre de connexions en parallèle, **forçant ainsi une sélection plus rigoureuse** des liens pertinents.

L'ajustement de l'**intensité de l'inhibition** γ peut être réalisé **dynamiquement** au fil du temps afin de s'adapter à la structure émergente du réseau. Une valeur croissante de γ favorise une **réduction progressive de la densité de connexions**, en particulier lorsque l'essaim de robots tend à devenir **trop centralisé** autour d'un unique cluster. À l'inverse, une diminution de γ **relâche la compétition**, autorisant une **communication plus étendue** entre les entités lorsque cela est nécessaire, notamment lors des phases de recherche et d'exploration.

Un cas pratique illustre ce mécanisme lorsque l'essaim doit passer d'une **organisation centralisée** à une **division en sous-groupes autonomes**. Supposons qu'un groupe de robots se coordonne autour d'un nœud pivot formant un **gros cluster connecté**. Si la mission impose désormais la **surveillance de plusieurs zones distinctes**, une **augmentation locale de γ** entraîne une **rupture des connexions moyennes**, forçant le **fractionnement du réseau** en sous-ensembles optimisés. Ce processus s'opère de manière **décentralisée**, chaque robot adaptant ses liaisons en fonction de l'**intensité locale de la compétition**, ce qui conduit à une **réorganisation progressive du SCN** selon les besoins de la mission.

C. Avantages et Cas d'Usage

Les mécanismes de **recuit simulé** et d'**inhibition modulée** peuvent être combinés pour exploiter à la fois leurs **effets exploratoires et sélectifs**, permettant ainsi une **optimisation dynamique** du SCN.

Le **recuit** joue un rôle clé dans la **phase d'exploration**, en autorisant la réorganisation des liens existants et la modification des structures émergentes. Il évite l'**enfermement prématué** du réseau dans un minimum local, en maintenant une **plasticité temporaire** des connexions. Cet effet est particulièrement utile dans les situations où un robot doit **réviser ses alliances stratégiques**, par exemple lorsqu'un cluster est sous-optimal et nécessite un **remaniement structurel**.

L'**inhibition modulée**, en revanche, introduit un **filtrage progressif**, permettant au réseau de **stabiliser des sous-groupes** en sélectionnant uniquement les liaisons les plus pertinentes. Une fois que la phase d'exploration a permis d'identifier de nouvelles interactions efficaces, l'inhibition assure une **convergence plus robuste**, en éliminant les connexions de faible synergie et en **solidifiant la structure des clusters**.

Dans un **environnement robotique évolutif**, où les conditions peuvent changer (déplacement d'obstacles, arrivée ou départ de robots, modification des objectifs de mission), cette combinaison assure une **adaptabilité continue** du SCN.

Lorsqu'un nouveau défi ou un changement d'environnement survient, une **brève phase de recuit** peut être déclenchée afin de **secouer** la structure actuelle et autoriser de nouvelles configurations. Ensuite, l'**inhibition progressive** intervient pour **stabiliser les ajustements**, consolidant ainsi de **nouvelles structures coopératives** plus adaptées aux nouvelles conditions de la mission.

Cette dynamique d'**alternance entre exploration et consolidation** permet à l'essaim de robots de **réorganiser intelligemment ses connexions**, en assurant un équilibre entre **réactivité et stabilité**, garantissant ainsi une **optimisation continue du SCN** face aux perturbations externes.

Conclusion

L'**essaim** robotique (chap. 2.5.3 et 4.7) exige une reconfiguration continue du **SCN** pour faire face aux changements de mission. Les **approches d'optimisation adaptative** (recuit simulé, inhibition modulée) fournissent les mécanismes nécessaires :

Recuit : injecte un bruit au réseau, brisant la stabilité d'un attracteur local et autorisant une reconfiguration plus profonde,

Inhibition variable : contrôle la “parcimonie” des liaisons, aidant à constituer des sous-groupes robustes ou, au contraire, à encourager plus de connectivité.

Cette **alliance** — recuit pour la **remise en question** de la topologie, inhibition pour la **sélection** stricte — permet à l'essaim de se **réarranger** de façon optimale ou quasi-optimale à chaque phase de la mission, en évitant les piègeages dans une configuration obsolète. Les sections suivantes (7.9.2.3) offriront des retours d'expériences et **discussions** sur la mise en œuvre et l'efficacité pratique de ces méthodes dans des simulations robotiques.

7.9.2.3. Résultats : Meilleure Coordination, Robustesse

Dans un **essaim** robotique ou un ensemble **multi-agent**, l'application des **approches d'optimisation adaptative** (recuit, inhibition dynamique, etc.) révèle généralement un **gain** notable sur deux axes : la **coordination** entre les agents et la **robustesse** du système face aux perturbations. Afin d'illustrer ces résultats, on considère, par exemple, un essaim de m robots, chacun doté d'un **vecteur d'actions** $\mathbf{a}_i(t)$ ou d'une politique locale, et relié aux autres via des pondérations $\omega_{i,j}(t)$ décrivant la **synergie** $S(R_i, R_j)$.

A. Coordination Accrue

Les mécanismes d'inhibition (chap. 7.4) et la **sparsification** (chap. 7.5.1) visent à **éviter** qu'un agent entretienne trop de liaisons simultanément, ce qui disperserait l'effort collectif. Sur le plan **mathématique**, on applique un terme $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ qui pénalise la somme des liens d'un même robot, ou on impose un $k\text{-NN local}$, incitant chaque agent à **sélectionner** seulement quelques synergies fortes.

Une conséquence directe est la formation de **sous-groupes** plus clairement définis. Dans un **essaim** de robots, un agent R_i ne cherche plus à établir des connexions avec l'ensemble du réseau, mais se **focalise sur deux ou trois partenaires** présentant la plus forte **complémentarité** ou **proximité**.

Cette dynamique entraîne une **auto-structuration** du réseau en **clusters**. Les robots impliqués dans un **sous-objectif commun**, tel qu'un **déplacement coordonné** ou des **manipulations synchronisées**, voient leurs connexions $\omega_{i,j}$ se **renforcer mutuellement**. Simultanément, les **liens moins pertinents**, soit parce qu'ils sont faiblement exploités, soit parce qu'ils n'apportent pas de contribution significative, sont progressivement **inhibés** jusqu'à leur extinction.

En fin de compte, l'**émergence** de groupements internes accroît la **coordination**. Les courbes $\omega_{i,j}(t)$ montrent que, dans ces sous-groupes, les liens finissent nettement plus élevés qu'en mode DSL basique, où la convergence locale peut être moins sélective.

B. Robustesse et Résilience

Dans un environnement **robotique**, les perturbations sont fréquentes, qu'il s'agisse de la panne d'un robot, de la dégradation d'un capteur ou d'un changement de priorité dans une zone de surveillance. Les mécanismes de **recuit simulé** (chap. 7.3) et d'**inhibition adaptative** (chap. 7.4) permettent une **reconfiguration dynamique** du SCN, assurant une meilleure adaptation aux imprévus.

Lorsqu'un **robot tombe en panne**, la mise à jour du SCN, influencée par l'inhibition et les perturbations stochastiques, redistribue progressivement la synergie vers d'autres partenaires, réduisant ainsi l'impact de la défaillance sur la performance globale du système. Si un **changement de mission** intervient, par exemple l'apparition d'une nouvelle cible ou une modification des priorités de couverture, un recuit même modéré peut **dissoudre les liaisons obsolètes**, favorisant la création de **nouvelles interactions** entre agents, mieux adaptées à la situation.

Par ailleurs, l'inhibition adaptative empêche qu'un **robot devienne un goulot d'étranglement**, en concentrant excessivement les interactions sur un unique acteur central. Lorsque la pondération d'un lien $\omega_{i,j} / \omega_{i,j}$ devient trop élevée, l'*augmentation de la somme $\sum k \omega_{i,k} \sum_k \omega_{i,k}$* entraîne une **régulation automatique**, limitant ce déséquilibre. Cette dynamique stabilise ainsi le réseau, rendant la structure plus résistante aux dérèglements potentiels d'un agent central.

C. Indicateurs de Performances Améliorées

L'impact des mécanismes d'optimisation peut être évalué à l'aide de plusieurs indicateurs mesurant les **gains en robustesse et en efficacité**.

Le **taux de réussite** ou le **temps d'accomplissement de la mission** sont des métriques essentielles dans les scénarios où un essaim de robots doit coordonner des tâches complexes, comme le transport coopératif ou la couverture d'une zone. Une organisation optimisée par recuit ou inhibition dynamique accélère souvent l'exécution de la mission et améliore le **taux de succès**, en réduisant les erreurs d'assignation ou les pertes de communication.

Les **mesures de robustesse** peuvent être obtenues en simulant la **panne aléatoire** d'un robot et en observant la capacité du SCN à se **réorganiser spontanément**. Dans un réseau bien optimisé, les liens synergiques se réallouent plus rapidement, garantissant une continuité des interactions et une meilleure résilience face aux défaillances locales.

Enfin, l'**analyse de la répartition des liens** dans le SCN final permet d'évaluer la polarisation des connexions. Un réseau optimisé affiche généralement une **structure plus contrastée**, où certains liens sont fortement consolidés tandis que d'autres disparaissent presque totalement. Cette répartition accentue la formation de **clusters bien définis**, rendant l'architecture du réseau plus lisible et plus facile à maintenir face aux évolutions dynamiques de l'environnement.

Conclusion

Lorsqu'on applique les **techniques** d'optimisation adaptative (recuit, inhibition dynamique, heuristiques globales) à un **essaim** de robots ou un système **multi-agent**, on obtient :

Une **meilleure coordination** : les robots s'organisent en sous-groupes cohérents, renforçant les liaisons vraiment utiles et éliminant celles qui n'apportent pas de plus-value.

Une **robustesse accrue** : l'adaptation continue permet de réagir rapidement aux pannes, aux changements de contexte ou à l'arrivée de nouveaux agents, évitant la stagnation dans des configurations obsolètes.

Ces résultats s'observent dans la pratique par des **performances** plus élevées (temps de mission réduit, moins de collisions) et une **distribution** des liaisons plus nette (mieux clusterisée). Ainsi, l'alliance de recuit (pour l'exploration globale) et d'inhibition modulée (pour la compétition locale) renforce la **résilience** de l'essaim, tout en améliorant sensiblement sa **coopération** et son **adaptation** face aux situations imprévues.

7.9.3. Scénario Symbolique + Sub-Symbolique

Dans de nombreux contextes, le **DSL** (Deep Synergy Learning) se déploie au contact d'entités hétérogènes : certaines sont **symboliques** (règles logiques, axiomes, concepts catégoriels), d'autres **sub-symboliques** (vecteurs d'embedding, représentations neuronales, etc.). Cette **hybridation** pose la question de savoir comment optimiser et faire évoluer un **SCN** (Synergistic Connection Network) qui doit gérer simultanément ces deux types d'entités, dont les mesures de synergie $S(i,j)$ peuvent se calculer selon des métriques très différentes.

7.9.3.1. DSL Hybride : Entités Logiques et Embeddings

Un **SCN** (Synergistic Connection Network) peut être entièrement sub-symbolique, c'est-à-dire manipuler des entités représentées par des **vecteurs** (embeddings), et définir une **synergie** $S(i,j)$ fondée sur une mesure de **similarité** (cosinus, noyau RBF, distance, etc.). À l'inverse, on peut envisager un **SCN** plus "symbolique" ou "logique", où les entités sont des **règles** ou des **concept**s reliés par une forme de compatibilité ou de contradiction.

Un **DSL** (Deep Synergy Learning) dit "hybride" combine ces deux aspects : on y trouve à la fois des **entités sub-symboliques** (embeddings) et des **entités symboliques** (règles logiques, axiomes sémantiques). Cela implique de concevoir une synergie S_{hyb} capable de gérer deux sources de "proximité" ou de "compatibilité" : l'une provenant d'un espace vectoriel, l'autre d'une structure logique. Ce mélange accroît la **richesse** du SCN, mais soulève des **défis** concernant la définition du S_{hyb} , la gestion des contradictions symboliques et la complexité globale.

A. Principe d'un SCN Hybride

Les entités $\{\mathcal{E}_i\}$ se répartissent en deux (ou plusieurs) **familles** :

- **Famille sub-symbolique** \mathcal{E}_{sub} , par exemple un ensemble de **vecteurs** (embeddings). Chaque $\mathbf{x}_i \in \mathbb{R}^d$ représente des données (images, sons, tokens de texte, etc.). La **mesure** $S_{\text{sub}}(i,j)$ repose typiquement sur une **similarité** vectorielle (cosinus, distance Gaussienne).
- **Famille symbolique** \mathcal{E}_{sym} , par exemple un ensemble de **règles**, **axiomes** ou **concept**s décrits sous forme logique, ou un ensemble de **labels** (catégories sémantiques). La **mesure** $S_{\text{sym}}(i,j)$ exprime la **compatibilité** logique (absence de contradiction), la co-occurrence, ou une évaluation de cohérence (ex. "ces règles se complètent").
- **Possibilité** d'entités "mixtes" \mathcal{E}_{mix} , liant des attributs symboliques et sub-symboliques.

Dans un **DSL** hybride, le **SCN** comporte donc des liens $\omega_{i,j}$ entre **tous** les types d'entités (vectorielles, logiques, mixtes). La **synergie** $S_{\text{hyb}}(i,j)$ doit unifier la partie sub-symbolique (S_{sub}) et la partie symbolique (S_{sym}) :

$$S_{\text{hyb}}(i,j) = \alpha S_{\text{sub}}(i,j) + \beta S_{\text{sym}}(i,j),$$

avec $\alpha, \beta \geq 0$. Ainsi, si deux entités sont cohérentes à la fois dans l'espace vectoriel (proches embeddings) et dans le plan logique (compatibilité de règles), leur synergie hybride est élevée. Inversement, si elles sont sémantiquement contradictoires, $S_{\text{sym}}(i,j)$ pourra être négatif, faisant baisser S_{hyb} .

B. Calcul de la Synergie et Organisation

La **mise à jour** DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hyb}}(i,j) - \tau \omega_{i,j}(t)] + \Delta_{(\text{inhibition, recuit, etc.})}$$

s'applique de la même manière qu'en DSL standard, mais la **valeur** de $S_{\text{hyb}}(i,j)$ provient d'une évaluation hybride. Il existe plusieurs **cas** :

- Deux **vecteurs** $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}_{\text{sub}}$: on évalue la similarité sub-symbolique (cosinus, RBF, etc.) comme d'habitude.
- Deux **règles/logiques** $\mathcal{E}_i, \mathcal{E}_j \in \mathcal{E}_{\text{sym}}$: on vérifie la cohérence $S_{\text{sym}}(i, j)$ (par exemple, absence de contradiction).
- Un **mélange** : l'un est vectoriel, l'autre symbolique ; on peut recourir à des modules sémantiques reliant un concept logique "Chat" à un embedding "cat" en traitant de la correspondance sémantique (des étiquettes ou une ontologie reliant le vecteur "chat" au concept "Chat").

Les **clusters** finaux résultent d'une auto-organisation dans laquelle des **blocs** sub-symboliques peuvent s'aligner avec des blocs symboliques, formant des "macro-nœuds" plus consistants (règles soutenues par exemples concrets, embeddings justifiés par axiomes). La dynamique DSL hybride clarifie alors les regroupements pertinents malgré la **nature hétérogène** des entités.

C. Avantages du DSL Hybride

Le **DSL hybride** combine les approches **symbolique** et **sub-symbolique**, permettant d'exploiter à la fois la **rigueur des règles logiques** et la **souplesse des modèles statistiques**. Cette hybridation assure une **cohérence conceptuelle**, tout en intégrant les **nuances floues** et les **tendances statistiques** détectées dans les données.

L'un des atouts majeurs du DSL hybride réside dans son **richesse sémantique**. D'un côté, la **logique symbolique** garantit la validité des relations entre entités, préservant la non-contradiction et l'alignement des règles avec la structure conceptuelle du domaine. De l'autre, le **sub-symbolique** (ex. embeddings) capture les **propriétés continues** et les **corrélations implicites**, facilitant l'intégration de connaissances plus complexes et moins strictement définies.

L'interaction entre les deux systèmes permet un **recouvrement efficace entre symbolique et statistique**. Un concept logique tel que *Oiseau* peut être associé à un ensemble de **vecteurs d'embedding** issus d'images ou de textes, où les entités sont regroupées par **similitude de forme ou de sémantique statistique**. L'association entre une **règle logique** et un **modèle sub-symbolique** renforce ainsi la reconnaissance : si une image est identifiée comme un oiseau sur la base de ses caractéristiques statistiques, cette classification est validée par la présence d'une **description symbolique cohérente** issue de la biologie. Cela favorise la formation de **macro-clusters**, où les **concepts logiques** viennent stabiliser et structurer les regroupements produits par l'apprentissage profond.

L'**interprétabilité** est un autre avantage essentiel du DSL hybride. Dans un modèle purement sub-symbolique, il est souvent difficile d'expliquer pourquoi certains vecteurs sont groupés ensemble. En intégrant des **entités logiques explicites**, il devient possible d'établir des liens clairs entre les **règles symboliques** et les regroupements statistiques. Un ensemble de vecteurs est alors identifié non seulement par sa proximité numérique, mais aussi par sa **compatibilité avec des relations symboliques connues**. Cette synergie offre ainsi une **meilleure transparence du SCN**, en fournissant une explication rationnelle à la structure du réseau et aux décisions prises par le modèle.

D. Points de Vigilance

Il convient de porter une attention particulière à plusieurs aspects lorsqu'on envisage d'intégrer, dans un même **SCN**, des composantes sub-symboliques et des composantes symboliques afin de calculer une **synergie globale**. Le premier point concerne la **complexité** résultant du passage d'un **espace vectoriel** à des **structures logiques** plus formelles. Dans la continuité des principes de la **section 2.2.1** sur la définition de $S(\mathcal{E}_i, \mathcal{E}_j)$, l'introduction d'une composante symbolique suppose une fusion entre un score vectoriel, par exemple $S_{\text{sub}}(\mathcal{E}_i, \mathcal{E}_j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|)$, et une compatibilité logique $S_{\text{sym}}(\mathcal{E}_i, \mathcal{E}_j)$. On peut alors définir

$$S(\mathcal{E}_i, \mathcal{E}_j) = \beta S_{\text{sub}}(\mathcal{E}_i, \mathcal{E}_j) + (1 - \beta) S_{\text{sym}}(\mathcal{E}_i, \mathcal{E}_j),$$

où la pondération $\beta \in [0,1]$ exige un **calibrage** rigoureux pour éviter une situation où une simple **contradiction** symbolique annihilerait des similarités vectorielles importantes, ou inversement. Cela illustre l'importance de gérer les **ordres de grandeur** respectifs : une négligence sur la valeur de β peut mener à un déséquilibre, compromettant la logique décrite en **section 2.2.3** (règles de parsimonie) ou la stabilité de la descente.

Un second point réside dans l'**évaluation** de la composante symbolique $S_{\text{sym}}(\mathcal{E}_i, \mathcal{E}_j)$. Les entités $\mathcal{E}_i, \mathcal{E}_j$ peuvent contenir des **représentations logiques**, telles que des formules, des clauses, ou des graphes de connaissances, dont la compatibilité n'est pas triviale à estimer. Ainsi, la mise à jour des pondérations $\omega_{i,j}(t)$ requiert à chaque itération un calcul complet de $S_{\text{sym}}(i, j)$. Lorsque le réseau compte un grand nombre d'entités n , la dynamique itérative sur $O(n^2)$ paires pourrait devenir coûteuse, surtout si la compatibilité symbolique exige un **moteur d'inférence** sophistiqué. Cette charge computationnelle accroît la complexité globale et doit être anticipée, en particulier si l'on veut mettre en place les heuristiques de la **section 7.3** (recuit simulé) ou de la **section 7.4** (inhibition variable) pour franchir les minima locaux.

Enfin, on observe que l'**introduction** d'un double mécanisme (sub-symbolique et symbolique) peut **multiplier** le nombre de minima locaux à l'intérieur de la fonction d'énergie globale $J(\omega)$. La dynamique DSL (décrise en **section 2.2.2**) peut alors rencontrer des difficultés accrues pour trouver un état globalement optimal, car le paysage d'énergie se trouve enrichi de nouvelles « vallées ». Pour atténuer ce phénomène et capitaliser pleinement sur la puissance du **DSL hybride**, il demeure nécessaire de mobiliser des **techniques** d'optimisation plus robustes, telles que le recuit simulé ou des heuristiques globales inspirées de la recherche tabou. Ces procédés, en ajoutant du **bruit** contrôlé ou des règles d'**inhibition** adaptative, élargissent l'exploration de l'espace de configuration et augmentent la probabilité de localiser un **minimum** plus satisfaisant.

Les **avantages** de cette combinaison résident dans la possibilité de tirer profit de la **cohérence logique** et de la **similarité** vectorielle : cela enrichit la capacité du réseau à former des **clusters** qui tiennent compte tant de la proximité sub-symbolique (ou sémantique) que des compatibilités formelles ou syntaxiques. Les **limites** se situent dans la complexité du calcul et dans la sensibilité au réglage des coefficients, qui peuvent nécessiter de nombreux essais pour parvenir à un équilibre. L'expérience empirique et l'analyse théorique, appuyées par les principes du recuit (chapitre 7.3) ou de l'inhibition ajustable (chapitre 7.4), demeurent souvent indispensables pour garantir que le **SCN hybride** se stabilise dans un état informatif et cohérent.

Conclusion

Le **DSL Hybride**, qui traite à la fois des **entités logiques** (règles, concepts) et des **embeddings** sub-symboliques, constitue une extension naturelle et ambitieuse du **SCN**. Cette approche :

Unifie la similarité vectorielle (sub-symbolique) et la compatibilité symbolique (logique) au sein d'une **même** fonction de synergie S_{hyb} .

Permet des **clusters** ou **macro-nœuds** combinant du raisonnement symbolique et de l'apprentissage statistique,

Nécessite des précautions (calibrage α, β , gestion de la complexité logicielle) et peut **multiplier** le nombre de minima locaux, renforçant l'importance des **mécanismes** d'optimisation (recuit, inhibition dynamique).

Les sections suivantes (7.9.3.2–7.9.3.3) proposeront des exemples ou **discussions** illustrant cette intégration et les **résultats** qu'on peut en attendre dans des tâches combinant données vectorielles (embeddings d'images ou de texte) et **axiomes** logiques (ontologies, règles, etc.).

7.9.3.2. Algorithmes d'Optimisation : si l'on Repère un Bloc de Règles Contradictoire, on Injecte du Bruit ou de l'Inhibition

Lorsque le **SCN** (Synergistic Connection Network) s'applique à des **entités logiques**, il peut survenir des **contradictions** ou conflits au sein d'un sous-ensemble de règles (ou de concepts) qui, prises ensemble, engendrent un bloc incohérent. Dans un DSL (Deep Synergy Learning) purement sub-symbolique, la synergie $S(i, j)$ repose sur la similarité vectorielle et se contente de minimiser J localement. Mais dans un **contexte hybride** intégrant des entités symboliques (cf. section 7.9.3.1), l'**incohérence** d'un bloc de règles peut empêcher la formation d'un cluster stable ou engendrer un attracteur local contradictoire. Pour sortir de ce **verrou**, on peut mobiliser les méthodes d'**optimisation** (recuit simulé, inhibition) déjà introduites au chapitre 7, en les **focalisant** sur la portion contradictoire du réseau.

A. Détection d'un Bloc Contradictoire

Dans la partie **logique** du SCN, certaines **règles** ou **concepts** peuvent se contredire. D'un point de vue symbolique, on détecte par exemple qu'un ensemble $\mathcal{B} \subseteq \{\mathcal{E}_{\text{sym}}\}$ ne peut être satisfaisant simultanément (ex. \mathcal{R}_α dit "si A alors B", \mathcal{R}_β dit "si A alors non-B"). Sur le plan **mathématique**, on peut formaliser un score $S_{\text{sym}}(\mathcal{R}_\alpha, \mathcal{R}_\beta)$ qui devient négatif ou nul quand \mathcal{R}_α et \mathcal{R}_β sont incompatibles. S'il se trouve que la **dynamique** DSL tente de renforcer $\omega_{\alpha,\beta}$ malgré un $S_{\text{sym}}(\alpha, \beta) \leq 0$, on se confronte à un bloc contradictoire.

Dans un **SCN** purement sub-symbolique, ce problème n'apparaît pas. Mais dans un DSL "hybride" (7.9.3.1), la dimension symbolique peut **entrer** en conflit avec certaines associations sub-symboliques, d'où la **besoin** d'une stratégie pour "**casser**" ou "**reconfigurer**" la portion contradictoire.

B. Injection de Bruit (Recuit) en Zone Contradictoire

On sait (cf. 7.3) que le **recuit simulé** se traduit par l'ajout d'un **bruit** $\xi_{i,j}(t)$ modulé par une température $\sigma(t)$. Au lieu de l'appliquer globalement à tout le SCN, on peut **focaliser** ce recuit sur la zone \mathcal{B} identifiée comme contradictoire. Formulons la mise à jour, pour $(i, j) \in \mathcal{B} \times \mathcal{B}$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{hyb}}(i, j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t).$$

L'introduction d'un **bruit ponctuel** dans la dynamique du SCN agit comme un mécanisme de **chaotisation contrôlée**, permettant de **rompre** des liaisons contradictoires et d'**encourager** la formation de connexions plus cohérentes. Cette perturbation locale permet d'**explorer** des configurations alternatives et d'éliminer des **structures incohérentes** qui auraient pu se figer dans un état sous-optimal.

L'**effet attendu** de cette injection de bruit est qu'après quelques **itérations de recuit** appliquées à la zone problématique \mathcal{B} , les liaisons ω impliquées dans la contradiction se réajustent progressivement, réduisant ou inversant certaines connexions de manière à produire un **état plus stable et plus logique**. Cette phase exploite la **plasticité temporaire** du SCN induite par le bruit pour permettre une **réorganisation adaptative** des relations entre entités.

Une fois cette **réorganisation initiée**, une **phase de refroidissement** est introduite pour stabiliser la nouvelle structure. La température σ , initialement élevée pour faciliter les **réarrangements**, est progressivement **diminuée**, réduisant ainsi l'amplitude des fluctuations stochastiques. En conséquence, les liaisons ω affectées cessent de subir des variations importantes et convergent vers une **configuration stable**, garantissant un réseau **logiquement plus cohérent** et débarrassé de ses contradictions initiales.

C. Inhibition Dynamique pour Contradiction

Souvent, un **bloc contradictoire** se manifeste parce que plusieurs règles/logiques se **renforcent** mutuellement malgré leur incohérence globale. On peut alors user de l'**inhibition** (chap. 7.4) pour limiter la somme des liens sortants d'une entité symbolique. Concrètement, la mise à jour inclut :

$$\omega_{\alpha,\beta}(t+1) = \omega_{\alpha,\beta}(t) + \eta [S_{\text{sym}}(\alpha, \beta) - \tau \omega_{\alpha,\beta}(t)] - \gamma \sum_{\beta' \neq \beta} \omega_{\alpha,\beta'}(t).$$

Ici, $\gamma > 0$ force la règle α à "choisir" un nombre limité d'associations fortes. Si deux règles β et β' sont logiquement contradictoires, la compétition fait que α ne pourra pas maintenir $\omega_{\alpha,\beta}$ et $\omega_{\alpha,\beta'}$, élevées simultanément. L'un des liens finira par chuter, levant la contradiction.

On peut localement **hausser** γ pour la portion contradictoire, intensifiant la sélection. Dès que la contradiction se résorbe, γ revient à un niveau normal. Cela agit comme un "**mode d'alarme**" dans la zone \mathcal{B} .

À l'issue, le SCN parvient à un état où chaque bloc de règles a dû opérer des compromis : un lien contradictoire est **réduit**, un autre est renforcé, etc. La partie symbolique obtient un certain **équilibre** (pas de contradiction interne) sans exiger de brutale coupure, mais via la compétition naturelle du DSL.

D. Complémentarité : Recuit + Inhibition

Il est souvent **pertinent** de combiner l'injection de **bruit** (recuit) avec l'**inhibition**. D'un côté, le recuit introduit une possibilité de "sortir" d'un attracteur local. De l'autre, l'inhibition structure la compétition pour éviter de multiples liens contradictoires soutenus en parallèle. Dans un **SCN** hybride, la zone contradictoire peut être **chauffée** (recuit), tandis que l'on **augmente** γ dans cette même zone, assurant que la "reconstruction" post-bruit ne tombe pas dans une nouvelle incohérence.

Conclusion

Lorsque des **règles** ou entités symboliques s'avèrent **contradictoires**, il est crucial de "déverrouiller" la portion de réseau concernée. Deux grandes approches d'**optimisation** peuvent être invoquées :

Recuit local : injecter un bruit sous contrôle d'une température $\sigma(t)$, pour permettre à la dynamique DSL de rompre les liens contradictoires et de s'essayer à d'autres configurations logiques,

Inhibition dynamique : accroître la compétition pour empêcher une entité de maintenir simultanément des relations élevées envers des règles incompatibles.

Cette combinaison — ou l'usage de l'un ou l'autre mécanisme — offre au **SCN** la **flexibilité** nécessaire pour gérer un **bloc** logique incohérent, et ramène l'ensemble du réseau vers une **cohérence** symbolique-sub-symbolique plus stable, en évitant d'avoir à redémarrer l'apprentissage de zéro ou à supprimer brutalement des entités.

7.9.3.3. Conclusion : Synergies Clarifiées, Clusters plus Pertinents

Le **scénario** hybride décrit en section 7.9.3, où cohabitent des **entités logiques** (symboliques) et des **entités** sub-symboliques (embeddings), soulève un intérêt particulier pour les **méthodes d'optimisation** (recuit simulé, heuristiques globales, inhibition) présentées tout au long du chapitre. L'objectif principal est de composer un **SCN** (Synergistic Connection Network) capable de faire émerger des **clusters** au sein desquels la cohérence symbolique et la proximité sub-symbolique se renforcent mutuellement. L'enjeu est de garantir que les **règles** logiques ne contredisent pas la structure statistique portée par les embeddings, tout en évitant des liaisons ambiguës ou mal justifiées.

Ci-après, on résume les **bénéfices** constatés lorsque ces mécanismes (recuit, inhibition modulée, etc.) sont appliqués pour clarifier les synergies entre blocs logiques et embeddings, et former des **clusters** plus pertinents.

A. Clarification des Synergies

Dans un cadre **hybride**, la fonction de synergie

$$S_{\text{hyb}}(i, j) = \alpha S_{\text{sub}}(i, j) + \beta S_{\text{sym}}(i, j)$$

peut engendrer des **ambiguités** si, par exemple, la composante symbolique S_{sym} s'avère contradictoire (plusieurs règles incompatibles) alors que S_{sub} (similitude vectorielle) encourage un rapprochement. Ce problème se matérialise lorsque le **DSL** pur cherche à renforcer des liens $\omega_{i,j}$ soutenus par la partie sub-symbolique, malgré l'existence d'une incohérence symbolique parallèle.

Un **recuit simulé** avec un bruit $\xi_{i,j}(t)$ modulé par $\sigma(t)$ amène la dynamique DSL à **revoir** certains liens. Même si localement un lien $\omega_{i,j}$ semblait avantageux pour la partie sub-symbolique, le recuit peut le rendre instable, permettant à la composante symbolique d'exprimer un éventuel conflit et, le cas échéant, de **faire décroître** ce lien. On "casse" ainsi des *liaisons ambiguës* contradictoires, puis on laisse la partie sub-symbolique consolider d'autres liaisons plus consensuelles.

En augmentant la **compétition** (cf. chap. 7.4), on contraint chaque entité (qu'elle soit un concept logique ou un vecteur embedding) à **sélectionner** un nombre limité d'associations fortes. Cela évite que deux entités symboliques incompatibles parviennent à subsister simultanément avec des $\omega_{i,j}$ élevés. La compétition réduit la possibilité de soutenir plusieurs règles contradictoires, clarifiant du même coup la structure globale.

De manière générale, l'injection de bruit ou l'accroissement de la compétition agit comme un “**filtre**” : on écarte petit à petit les liens dont la **double cohérence** (symbolique + sub-symbolique) ne peut s'accorder, et on renforce les liens soutenus par **les deux** composantes. Il en résulte une **synergie** $S_{\text{hyb}}(i,j)$ plus nettement exprimée dans la matrice ω .

B. Clusters plus Pertinents

La clarification progressive des **liaisons** et l'élimination des liens contradictoires ou trop faibles favorisent la formation de **clusters** nettement plus pertinents au sein du **SCN** (Synergistic Connection Network). Ce phénomène de regroupement émerge directement de la dynamique de mise à jour des pondérations présentée à la **section 2.2.2**, où chaque $\omega_{i,j}(t)$ évolue sous l'influence de la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ et de la **décroissance** $\tau \omega_{i,j}(t)$. Lorsque s'y ajoutent des mécanismes de **compétition** ou d'**inhibition** (cf. **chap. 7.4**), on assiste à une “binarisation” progressive qui pousse les liaisons porteuses de contradictions ou de synergie insuffisante vers zéro et renforce en parallèle celles jugées plus fiables ou plus cohérentes.

Dans un tel contexte, la distribution des poids $\omega_{i,j}$ se polarise : de nombreux liens finissent proches de zéro, tandis qu'un ensemble plus restreint de connexions s'accroît jusqu'à atteindre des valeurs nettement supérieures à la moyenne. Cette dynamique traduit la tendance naturelle du **SCN** à “trier” les entités : les paires (i,j) présentant à la fois une cohérence logique et une similarité sub-symbolique élevée voient leurs pondérations culminer, alors que les couples inappropriés ou contradictoires convergent vers des liens quasi nuls. L'**inhibition modulée** et la **saturation** (décrites au **chapitre 7.4** et parfois à la **section 7.4.2** en particulier) consolident encore cet effet de **sélection**.

En conséquence, les **clusters** émergent sous une forme plus lisible. Chacun d'eux réunit des **règles** (concepts logiques) et des **embeddings** (vecteurs d'images ou de textes) qui se renforcent mutuellement. Ce processus est conforme aux principes de la **section 2.2.5** sur la formation de micro-clusters, lesquels s'amplifient ici grâce à la synergie symbolique-sub-symbolique. Les entités qui ne parviennent pas à concilier leurs aspects logiques et leur proximité vectorielle avec un bloc donné sont alors poussées à la périphérie, voire rattachées à un cluster secondaire plus adapté à leurs caractéristiques.

La **portée pratique** de cette organisation apparaît dans les tâches d'exploitation en aval, puisque chaque bloc cohérent devient un groupe à la fois solide sur le plan logique (pas d'auto-contradiction interne) et homogène sur le plan sub-symbolique (embeddings fortement similaires). Les clusters peuvent ainsi recevoir une **interprétation** aisée, par exemple lorsque l'on constate que certains blocs associent un concept comme « AnimalVolant » à une collection d'images ou de représentations textuelles directement reliées aux oiseaux ou aux insectes. Cette correspondance symbolique-sub-symbolique augmente la **valeur** du SCN pour des applications variées, puisque les liens internes élevés témoignent d'une synergie équilibrée et exploitable pour l'inférence, la classification ou la recommandation de nouveaux contenus.

C. Conclusion Globale

Dans un **DSL** hybride intégrant **entités logiques** et **embeddings**, l'emploi des **mécanismes** d'optimisation (chap. 7) se révèle décisif pour :

- **Clarifier** les synergies ambiguës : le recuit simulé introduit un “grain de folie” qui rompt les attracteurs contradictoires, l'inhibition modulée évite la persistance de multiples liens incohérents simultanément.
- **Converger** vers des **clusters** où la **cohérence** symbolique (absence de contradictions) et la **proximité** sub-symbolique (similarité vectorielle) s'articulent en synergie.

Cette dynamique, une fois stabilisée, conduit à des **blocs** plus explicites et plus utiles, dans lesquels la **partie** logique est soutenue par des exemples sub-symboliques appropriés, et la **partie** sub-symbolique gagne en sémantique via l'ancre symbolique. Le SCN final obtient alors des **clusters** plus robustes, plus **interprétables**, et plus conformes à la double exigence du **DSL** : gérer la variabilité statistique tout en respectant des règles logiques.

7.10. Conclusion et Ouverture

7.10.1. Récapitulatif du Chapitre

7.10.1.1. On a présenté diverses approches d'optimisation (recuit, heuristiques) et d'adaptation (inhibition dynamique, apprentissage continu).

Le **Chapitre 7** a mis en exergue les difficultés liées à la **dynamique** du Deep Synergy Learning (DSL) lorsqu'on cherche à optimiser la structure d'un Synergistic Connection Network (SCN). Au-delà de la simple mise à jour locale de chaque pondération $\omega_{i,j}$, il est apparu nécessaire d'introduire des **mécanismes** plus élaborés pour échapper aux minima locaux, maintenir un certain degré de **compétition** entre les liens et gérer l'arrivée **incrémentale** de nouvelles entités ou de nouveaux flux de données.

Dans une première partie, le **recuit simulé** (section 7.3) a été examiné en détail. L'idée directrice est de laisser la matrice ω évoluer selon la règle standard de la dynamique auto-organisée (cf. Chapitre 4), mais en y superposant un **terme stochastique** qui dépend d'une température $T(t)$. Cette température décroît généralement au fil des itérations :

$$T(t+1) = \alpha T(t), \quad \text{pour un facteur } 0 < \alpha < 1.$$

Le rôle de ce terme est de permettre des "sauts" aléatoires dans l'espace des pondérations, si bien que le réseau n'est pas prisonnier d'un **bassin d'attraction** trop étroit. Lorsque $T(t)$ est encore assez élevée, on autorise des modifications plus audacieuses de ω , lesquelles pourraient augmenter temporairement l'"énergie" du système, mais lui éviter de se figer dans un **minimum local**. À mesure que la température baisse, ces écarts deviennent plus rares et la dynamique finit par se stabiliser, idéalement dans une configuration de meilleure qualité que celle obtenue par une descente purement déterministe.

Au-delà du recuit, plusieurs **heuristiques** ont été envisagées pour renforcer l'**optimisation globale** (section 7.5). La **sparsification contrôlée** consiste à supprimer régulièrement les liaisons faiblement pondérées (en deçà d'un certain seuil adaptatif), ce qui clarifie la structure et limite l'influence de multiples liens « moyens » qui, pris collectivement, pourraient piéger le réseau dans un état sous-optimal. Les **algorithmes génétiques**, pour leur part, considèrent la matrice ω comme un "génome" dont on peut faire varier certains éléments (mutations, croisements) pour explorer des configurations plus éloignées. Enfin, la stratégie de "shake" périodique introduit un **bruit** collectif sur ω à intervalles réguliers, obligeant le SCN à "ressasser" sa structure et éventuellement découvrir de meilleurs attracteurs.

Un second axe a porté sur les **mécanismes d'adaptation dynamique**, et notamment sur l'**inhibition avancée** (section 7.4). De base, le DSL sait déjà couper ou réduire les liens de faible synergie, et imposer une décroissance via $\tau \omega_{i,j}$. Mais pour éviter que trop de liaisons "moyennes" ne persistent, on peut restreindre la capacité de chaque entité \mathcal{E}_i à entretenir un grand nombre (ou un trop grand cumul) de liaisons. Cela se formule par un terme de pénalisation supplémentaire $\gamma \sum_k \omega_{i,k}$, qui agit comme un "budget" limitant la somme des pondérations sortantes. De cette façon, seules les connexions vraiment profitables (au sens de la synergie S) sont conservées à long terme.

La notion d'**apprentissage continu** (section 7.6) fait quant à elle référence au fait que le SCN n'est pas conçu uniquement pour un entraînement "hors-ligne" sur un ensemble fixe d'entités. Au contraire, on peut recevoir de nouveaux flux (par exemple, de nouvelles données capteurs ou de nouvelles entités symboliques) au cours du temps, et on souhaite que l'**auto-organisation** se mette à jour **incrémentalement**. Les formules de mise à jour locales, conjuguées à des ajustements d'inhibition et à une surveillance de la **densité** globale du réseau, permettent alors d'intégrer ces nouveaux éléments sans réapprendre depuis zéro. Le SCN s'adapte donc en douceur, préservant les clusters déjà formés lorsque ceux-ci restent pertinents, et créant ou modulant de nouveaux sous-groupes s'il détecte des synergies naissantes entre les entités fraîchement arrivées et celles déjà en place.

Plusieurs **constats** se dégagent de l'ensemble de ces approches. D'abord, il est crucial de disposer de **stratégies stochastiques** et de **mécanismes** d'exploration pour éviter le **piège** des minima locaux, car la simple descente d'énergie (guidée par la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

) peut conduire à des configurations sous-optimales si la fonction de synergie S comporte plusieurs vallées disjointes. Par ailleurs, la **compétition** doit être gérée de manière avancée, via l'inhibition ou la saturation, pour éviter une sur-dispersion des pondérations : il est souvent plus sain, aussi bien pour la clarté des clusters que pour l'efficacité du réseau, de maintenir des liens vraiment forts dans un nombre restreint de connexions, plutôt que de se retrouver avec une matrice ω dense en modeste synergie. Enfin, le **mode incrémental** proposé par l'apprentissage continu s'avère essentiel dans de nombreux scénarios réels, où les **données** arrivent de manière permanente ou par gros paquets successifs, et où l'on veut que le SCN s'adapte en "temps réel" pour capter les **nouvelles** synergies sans oublier les anciennes.

En agrégeant tous ces outils, le DSL acquiert une **robustesse** et une **flexibilité** accrues. Le recuit, les heuristiques globales et les mécanismes de compétition favorisent une **exploration** profonde de l'espace des poids, assurant une qualité d'organisation supérieure ; l'apprentissage continu ouvre la voie à des **applications** durables, capables de gérer le **flux** et la **variabilité** des données. À la fin du Chapitre 7, la conjonction de ces méthodes est présentée comme un **cadre uniifié**, transcendé par la philosophie générale de l'auto-organisation : le DSL n'est pas simplement une règle locale, mais un écosystème algorithmique dans lequel des ajustements stochastiques, des inhibitions modulées et une intégration incrémentale des données se combinent pour aboutir à un SCN performant et évolutif.

Ces conclusions préparent la voie aux applications plus spécifiques qui seront abordées dans les chapitres suivants (Chap. 8 et Chap. 9), notamment en **multimodalité** et en **apprentissage temps réel**, domaines où les entités et les flux de données sont particulièrement nombreux et variés, requérant toute la panoplie d'optimisations et d'adaptations décrites dans ce Chapitre 7.

7.10.1.2. Couplage du DSL avec un Signal de Récompense (RL) ou la Gestion Incrémentale (Flux)

Les développements précédents, tout au long du **chapitre 7**, ont décrit plusieurs **méthodes d'optimisation adaptatives** dans un **SCN (Synergistic Connection Network)**, visant à **éviter les minima locaux** et à **clarifier la structure des liaisons**. En complément des mécanismes existants tels que le **recuit**, les **heuristiques globales** ou encore l'**inhibition modulée**, deux prolongements permettent d'élargir les capacités d'adaptation du SCN.

Le premier prolongement consiste à **coupler la dynamique DSL avec un signal de récompense**, inspiré des approches en **apprentissage par renforcement (RL)**. Ce signal extrinsèque oriente la formation des liaisons $\omega_{i,j}$ et influence l'évolution du réseau au-delà de sa seule dynamique auto-organisée.

Le second prolongement repose sur une **gestion incrémentale des entités**, permettant de traiter en **flux** l'apparition ou la disparition d'entités dans le SCN. Contrairement à un modèle statique, cette approche assure une adaptation continue des liaisons et favorise une meilleure flexibilité du réseau face aux changements structurels.

Ces deux approches prolongent la **logique auto-organisée** du DSL en y intégrant soit des **retours externes** (via le signal de récompense), soit une **évolution dynamique** (ajout et retrait d'entités), garantissant ainsi une **meilleure adaptabilité** du réseau.

A. Couplage DSL-RL : Injection d'un Signal de Récompense

Le **DSL** (Deep Synergy Learning) repose sur la mise à jour des pondérations $\omega_{i,j}(t)$ en fonction d'une **synergie** $S(i,j)$. Pour donner un **objectif** global supplémentaire, on peut introduire un **signal de récompense** $\mathcal{R}(t)$. Cette récompense peut être un **score** mesurant la réussite d'une tâche (p. ex. un taux de détection, une performance collective). On complète alors la mise à jour classique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \kappa \mathcal{R}(t) \Phi(i,j,t).$$

- $\mathcal{R}(t)$: un **signal** extrinsèque, pouvant être global (même valeur pour tous) ou local (une portion de la récompense associée à certains noeuds).

- $\Phi(i, j, t)$: éventuel *masque* ou *poids* qui détermine la sensibilité du lien (i, j) à cette récompense (ex. si le lien est pertinent pour la tâche).
- κ : un coefficient réglant l'impact de $\mathcal{R}(t)$ dans la mise à jour.

L'introduction d'un **signal de récompense** dans un **SCN (Synergistic Connection Network)** permet d'aligner la formation des liaisons $\omega_{i,j}$ sur un **objectif fonctionnel**, dépassant ainsi la simple logique de synergie **interne** $S(i, j)$. Un lien ne se renforce plus uniquement en fonction des interactions locales, mais également selon son **impact sur la performance finale**. Lorsqu'un lien $\omega_{i,j}$ contribue à améliorer la **récompense globale**, il bénéficie d'un **renforcement préférentiel**, tandis que les liens peu pertinents ou nuisibles sont progressivement réduits.

Cette approche peut être vue comme une **analogie avec l'apprentissage par renforcement (RL)** dans un cadre **multi-agent** ou **semi-supervisé**. Les liaisons $\omega_{i,j}$ fonctionnent alors comme des **associations dynamiques**, renforcées ou affaiblies en fonction du signal $\mathcal{R}(t)$, et les entités $\{\mathcal{E}_i\}$ peuvent apprendre à **coopérer** pour **maximiser un score global**.

Plusieurs **implémentations** sont possibles pour intégrer cette dynamique de récompense dans un SCN. Une première approche repose sur une **récompense globale**, où un unique **signal** $\mathcal{R}(t)$ est attribué à tout le réseau. Ce schéma implique un **feedback partagé**, renforçant uniformément les liens impliqués dans les interactions ayant conduit à une bonne performance globale. A l'inverse, une **récompense locale** peut être définie pour chaque entité i sous la forme $\mathcal{R}_i(t)$, permettant de moduler l'impact des liens $\omega_{i,j}$ en fonction de la **contribution individuelle** des entités connectées. Dans ce cas, une pondération comme $\Phi(i, j, t) \approx [\mathcal{R}_i(t) + \mathcal{R}_j(t)]$ permet d'attribuer une **récompense distribuée**, chaque lien recevant un retour en fonction des performances combinées des **nœuds associés**.

B. Gestion Incrémentale du Flux (Apprentissage Continu)

Dans un **DSL** standard, on considère souvent un ensemble fixe d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Or, de nombreux contextes requièrent une **mise à jour en flux** : nouvelles entités apparaissent, entités obsolètes disparaissent. Dans ce cas, redémarrer la dynamique DSL *ab initio* (calcul complet) après chaque modification serait trop onéreux.

Supposons qu'au temps t , une **entité** \mathcal{E}_{n+1} survienne. On crée alors de nouvelles pondérations $\omega_{(n+1),j}$ pour tous $j \in \{1, \dots, n\}$. Plutôt que de les calculer exhaustivement, on peut :

$$\omega_{(n+1),j}(0) = \delta \times S(\mathcal{E}_{n+1}, \mathcal{E}_j)$$

avec δ petit (ou 0 pour la plupart des j , sauf un échantillon restreint). Puis on **laisse** la dynamique DSL s'exécuter, mais *seulement* autour d'un voisinage restreint $\mathcal{V}_{(n+1)}$. Cela évite un recalcul global. Au bout de T_{local} itérations, $\omega_{(n+1),j}$ atteint un état stable ou cohérent au SCN existant.

De même, lorsqu'une entité se révèle obsolète, on peut procéder à un **isolement progressif** : diminuer $\omega_{i,\text{old}}$ de façon linéaire ou exponentielle (chap. 7.6.2.2) jusqu'à les annuler. On retire alors \mathcal{E}_{old} du SCN. Cela assure que la disparition d'une entité se fasse **en douceur**, laissant les autres réallouer leurs synergies.

Cette **gestion incrémentale** (ou en flux) épargne le coût d'une **reconstruction** complète du réseau. Le SCN demeure un **processus continu** : chaque nouvelle entité \mathcal{E}_{n+1} prend sa place en **quelques** mises à jour, se connectant aux entités déjà en place. La dynamique DSL, éventuellement assistée de recuit ou d'inhibition locale (chap. 7.6, 7.4), assure la fluidité de cette transition.

Synthèse

En sus des mécanismes d'**optimisation** centrés sur l'évitement des minima locaux ou la compétition entre liens, deux **axes** étendent la **portée** et la **flexibilité** du **DSL** :

Couplage à un Signal de Récompense :

Permet d'intégrer un **objectif** global. Les liens $\omega_{i,j}$ se forment alors à la fois sur la base de la synergie locale $S(i,j)$ et de la **performance** globale $\mathcal{R}(t)$. L'exemple type est l'apprentissage par renforcement (RL), où le SCN devient un vecteur de coordination entre agents souhaitant maximiser un **score**.

Gestion Incrémentelle (Apprentissage Continu) :

Les entités $\{\mathcal{E}_i\}$ ne sont plus figées : on ajoute ou on retire des nœuds et leurs liaisons en cours de route, sans recourir à un recalcul global. Les **pondérations** ω s'adaptent localement via la même **règle** DSL, maintenant le SCN **vivant** et toujours à jour face à de nouvelles entités ou missions.

Grâce à ces **extensions**, le **DSL** dépasse le stade d'une auto-organisation statique : il peut **s'aligner** sur des finalités externes (via $\mathcal{R}(t)$) et **s'adapter** en flux continu (apparition ou retrait d'entités). Cela le rend pertinent pour des environnements ou systèmes dynamiques (multi-robots, sources de données arrivant en streaming), tout en conservant sa **philosophie** d'auto-organisation locale et compétitive.

7.10.2. Liens vers Chapitres 8 et 9

Dans cette section de conclusion, nous souhaitons replacer les **méthodes d'optimisation** et **d'adaptation** (présentées tout au long de ce chapitre 7) dans la trajectoire du livre. Après avoir exposé comment perfectionner la dynamique du SCN (Synergistic Connection Network) — via recuit simulé, heuristiques globales, inhibition avancée, apprentissage continu, etc. — nous verrons comment ces techniques s'appliquent et s'étendent dans les **chapitres** suivants, en particulier le **Chap. 8** (DSL Multimodal) et le **Chap. 9** (Évolutions Temps Réel).

7.10.2.1. Chap. 8 (DSL Multimodal) : Recours aux Algorithmes d'Optimisation pour Gérer des Canaux Multiples (image, audio, texte)

Les analyses menées jusqu'à présent dans ce chapitre (chap. 7) proposent divers **mécanismes** d'optimisation et d'adaptation (recuit simulé, heuristiques globales, inhibition modulée, etc.) appliqués à un **SCN** (Synergistic Connection Network) de taille moyenne ou dans des contextes “monomodaux”. Le **Chapitre 8** envisage un **scénario multimodal**, où coexistent différents **canaux** (visions, audio, texte, ou autres capteurs). Cette extension multimodale intensifie plusieurs **défis** déjà entrevus :

- **Explosion** du nombre d'entités $\{\mathcal{E}_i\}$ si chaque canal apporte un large ensemble de features ou d'objets (patchs d'images, embeddings audio, tokens textuels, etc.).
- **Nécessité** de maintenir une **fusion** ou une **cohérence** entre plusieurs modalités, ce qui peut imposer des synergies cross-modales complexes.
- **Multiplication** de potentiels minima locaux, car chaque modalité structure ses propres clusters, et la rencontre entre clusters d'images, de sons ou de textes peut engendrer des attracteurs hybrides.

Pour répondre à ce foisonnement, on recourt aux **approches** d'optimisation élaborées dans ce chapitre (sparsification, recuit, heuristiques compétitives), permettant de **piloter** la dynamique DSL vers une intégration cohérente et non surchargée de ces multiples canaux.

A. Fusion Multimodale et Complexité Accrue

Lorsque plusieurs **flux d'information** coexistent dans un **SCN** (Synergistic Connection Network), le **nombre total d'entités** n peut croître considérablement. Un **flux vision** peut générer des centaines de **descripteurs** ou **patchs**, chacun correspondant à une région d'image ou à une frame vidéo. Un **flux audio** peut décomposer un signal en plusieurs **frames** ou **embeddings spectro-temporels**, tandis qu'un **flux texte** contient un grand nombre de **tokens** transformés en représentations vectorielles. Le SCN doit alors gérer un ensemble **massif et hétérogène** de nœuds, comprenant des entités issues de domaines très différents : **image, audio, texte**, chacun portant une structure et des propriétés distinctes.

Le premier défi est celui du **coût computationnel**. Le traitement exhaustif de toutes les connexions $\omega_{i,j}$ entre n entités entraîne un **coût quadratique** $O(n^2)$, ce qui devient rapidement prohibitif (chap. 7.2.3). Cette explosion du nombre de liaisons justifie l'utilisation de **stratégies d'optimisation**, parmi lesquelles figurent les **mécanismes d'inhibition** (chap. 7.4) et de **sparsification** (chap. 7.5) pour restreindre le nombre de connexions réellement **actives**. En complément, les **heuristiques globales** comme le **recuit simulé** ou les **algorithmes d'optimisation combinatoire** permettent d'**éviter un balayage exhaustif** et de structurer le réseau de manière plus efficace.

Un second défi réside dans le **croisement inter-modal**. La définition de la **synergie** $S(i,j)$ entre deux entités peut nécessiter des comparaisons entre des **données de nature très différente**, comme un **embedding audio** et un **mot textuel**. Le **Chapitre 8** explore ces questions en profondeur, notamment sur les méthodes permettant de mesurer et d'aligner la synergie entre flux multimodaux. Dans ce contexte, les algorithmes d'optimisation décrits dans le **chapitre 7** prennent toute leur importance pour dépasser les **barrières locales**, souvent induites par l'absence de **repères communs** entre les différentes modalités du SCN.

B. Inhibition et Saturation Multimodale

Dans un cadre **multimodal**, il est fréquent qu'une modalité (ex. vision) domine en quantité ou en expressivité. Sans mécanisme de contrôle, de nombreux liens “vision–vision” peuvent saturer la matrice ω . Les techniques **d'inhibition** avancée (7.4) permettent de **restreindre** la prolifération de liaisons intra-canal, ouvrant la voie à des connexions inter-canal (audio–texte, vision–texte) plus équilibrées.

Si on détecte qu'un flux “X” occupe 80 % des liens actifs, on peut localement **augmenter** la concurrence (inhibition) au sein des entités de la modalité X, pour qu'elles ne conservent que les associations vraiment cruciales. Cela **libère** le SCN pour d'autres canaux, évitant l'écrasement d'une modalité par une autre.

En complément, on peut définir une **valeur max** ω_{\max} pour un lien, ou appliquer un **hard threshold** (7.4.3.1) pour forcer la coupure des liaisons en dessous d'un certain niveau. Cela clarifie la structure dans un cadre hétérogène, où certaines synergies inter-modales risquent d'être confondues avec d'innombrables synergies intra-modales légèrement élevées.

Dès qu'on veut **re-agencer** la fusion multimodale (par exemple, le SCN se bloque sur un regroupement purement visuel, ignorant l'audio), un recuit simulé (7.3) redonne de la **plasticité** à la structure, permettant aux liens “audio–texte” d'émerger et aux liens strictement “vision–vision” de se tempérer.

C. Heuristiques pour Gérer les Flux Variés

Les chapitres **7.5 (Heuristiques globales)** et **7.6 (Apprentissage continu)** introduisent plusieurs **outils d'optimisation** permettant de traiter des **réseaux vastes et évolutifs**. Dans un **environnement multimodal**, où les entités peuvent provenir de sources hétérogènes comme l'image, l'audio ou le texte, plusieurs stratégies permettent de limiter la complexité et d'assurer une **structuration efficace** du SCN (**Synergistic Connection Network**).

Une première approche consiste à utiliser des **méthodes d'échantillonnage**, permettant de réduire le **coût quadratique** $O(n^2)$ en limitant le calcul des synergies aux **liaisons les plus pertinentes**. Une solution efficace consiste à appliquer un **k-NN local** au sein de chaque modalité, puis à réaliser un **k-NN inter-modal** afin de fusionner les sous-réseaux obtenus. Ce procédé permet d'identifier des relations importantes sans devoir traiter **exhaustivement toutes les paires** (i,j) , évitant ainsi une explosion combinatoire.

Les **heuristiques d'exploration** constituent une seconde approche permettant d'optimiser la **répartition des clusters** dans un SCN multimodal. Des algorithmes globaux tels que les méthodes **génétiques**, le **recuit simulé**, ou les **random restarts** permettent de **dépasser les barrières énergétiques** qui peuvent isoler les entités en fonction de leur modalité d'origine. Par exemple, un protocole de **shake dynamique** (chap. 7.2.2.3) peut être déclenché si l'on observe une **partition trop stricte** entre entités **vision–vision** et **texte–texte**, alors que l'objectif est de favoriser une **fusion inter-modal** plus complète.

Enfin, dans le cas de flux multimodaux **temporels** (ex. vidéo en streaming, audio en temps réel), l'**apprentissage continu** joue un rôle fondamental. Les mécanismes **d'insertion incrémentale d'entités** (chap. 7.6) permettent d'intégrer progressivement de **nouveaux échantillons**, en les connectant localement à leurs plus proches voisins avant

d'étendre leur influence dans le réseau. Cette approche assure une **croissance progressive** du SCN, en évitant un recalcul complet des connexions à chaque mise à jour du système.

D. Exemples d'Application Annoncés pour Chap. 8

Le **Chapitre 8** décrira dans le détail :

- **Comment** on définit la synergie S pour comparer un embedding d'image vs. un embedding de texte, ou un segment audio vs. un concept sémantique.
- **Comment** s'organise la dynamique DSL si l'on dispose de multiples familles d'entités ($\mathcal{E}_{\text{vision}}, \mathcal{E}_{\text{audio}}, \mathcal{E}_{\text{text}}$), et si l'on applique l'inhibition pour éviter une prolifération intra-canal.
- **Comment** l'on recourt aux **mêmes** algorithmes d'optimisation (recuit, heuristiques, etc.) afin de "mixer" au sein d'un cluster des entités d'origines différentes, franchissant les barrières entre flux.

On verra notamment l'intérêt pour des **tâches** de correspondance image-légende, alignement audio-texte (transcription), ou fusion vision-audio dans un robot multi-capteur. Ces cas mettent en évidence la **puissance** de la structure DSL, couplée aux **techniques** d'optimisation du chapitre 7, pour **unifier** des flux hétérogènes en un seul réseau auto-organisé.

Conclusion

Le **Chapitre 8** (DSL Multimodal) s'appuiera massivement sur les **algorithmes** d'optimisation et d'adaptation développés dans le **Chapitre 7**, pour gérer la **fusion** de différents canaux (image, audio, texte, etc.) :

- **Croissance** rapide du nombre d'entités → on applique **sparsification** (chap. 7.5) et **inhibition** (chap. 7.4) pour maintenir un réseau gérable,
- **Potentialité** de minima locaux "mono-modaux" → on emploie **recuit** (chap. 7.3) ou heuristiques globales afin de "casser" ces attracteurs et autoriser une véritable fusion inter-modale,
- **Flux évolutif** → on introduit les mécanismes d'**apprentissage continu** (chap. 7.6) pour insérer ou retirer des entités issues de différents canaux, sans reconduire un recalcul complet.

Ainsi, la **dynamique** DSL, combinée à l'inhibition, au recuit, ou encore au couplage $\mathcal{R}(t)$ (RL), fournira un **cadre** solide pour **agréger** et **auto-organiser** des données variées, promesse de nombreuses applications multimodales décrites plus en détail dans le **Chapitre 8**.

7.10.2.2. Chapitre 9 (Évolutions Temps Réel) : Approfondissement du Concept d'Adaptation en Flux et Robustesse à plus Grande Échelle

Un **SCN** (Synergistic Connection Network) appliqué à un environnement dynamique se heurte, à mesure que croissent le volume et la variabilité des données, à des défis de **réactivité** et de **robustesse**. Le **Chapitre 9** mettra l'accent sur la façon d'appliquer en **temps réel** les mécanismes décrits dans la présente section (chap. 7), en insistant sur la capacité à gérer un **flux** continu d'entités (ou de modifications) et à demeurer stable malgré les perturbations incessantes. Les sections précédentes (7.6) ont présenté quelques outils d'**apprentissage continu**, tandis que 7.3, 7.4 et 7.5 ont fourni un répertoire d'**optimisations** (recuit, heuristiques globales, inhibition adaptative) dont la pertinence croît encore dès lors que l'on veut préserver un **SCN vivant** et ne pas se retrouver submergé par de trop grands volumes de données.

A. Adaptation en Flux et Insertion Incrémentale d'Entités

Le **Chapitre 9** explorera le scénario où de **nouvelles entités** $\{\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \dots\}$ apparaissent au fil du temps. Ces entités peuvent correspondre à de nouveaux capteurs, à des documents arrivant en streaming, ou à des robots supplémentaires rejoignant un essaim (voir 7.9.2.1). Dans de telles circonstances, relancer de zéro la dynamique DSL et recalculer toute la matrice $\{\omega_{i,j}\}$ serait prohibitif en temps de calcul, surtout si l'on se situe dans un contexte de grande échelle ou de flux continu.

Afin de répondre à ce besoin, la logique d'**insertion** incrémentale se met en place. Une entité \mathcal{E}_{n+1} se voit allouer un **voisinage** ou un échantillon $\mathcal{V} \subseteq \{1, \dots, n\}$ pour lesquelles on calcule $\omega_{(n+1),j}$ en partant d'une petite valeur initiale δ . On utilise alors la **même** règle DSL :

$$\omega_{(n+1),j}(t+1) = \omega_{(n+1),j}(t) + \eta [S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau \omega_{(n+1),j}(t)].$$

Cette formule peut inclure un **terme** d'inhibition si l'on souhaite limiter la densité de liaisons que \mathcal{E}_{n+1} entretient simultanément. Les itérations se concentrent localement, évitant un recalcul global. Le **Chapitre 9** décrira en détail ce genre de protocoles d'**adaptation en flux**, où l'on doit insérer (ou retirer) des entités de manière continue et stabiliser leurs liens ω par quelques itérations partielles seulement.

B. Robustesse et Perturbations dans de Grands Volumes

Un **SCN** de taille importante se confronte au risque de **minima** locaux, de structures figées, ou d'une sur-connexion aboutissant à un coût en $O(n^2)$ (voir 7.2.3). Le **Chapitre 9** analysera également la manière dont les algorithmes d'**optimisation** (recuit, heuristiques globales) décrits au chapitre 7 contribuent à maintenir une **robustesse** en présence de modifications incessantes. Les idées clés sont les suivantes.

Dans un flux de données massives, le recuit prend la forme d'injections stochastiques régulières : toutes les K itérations ou à chaque palier de volumes, on applique un "coup de chaleur" localisé (chap. 7.3) qui secoue la configuration de ω . Les liens qui se trouvaient sur le point de s'installer dans un attracteur local obsolète peuvent se réorienter, permettant une reconfiguration si le flux de nouvelles entités l'exige.

Les mécanismes d'inhibition (chap. 7.4) ou de saturation (chap. 7.4.3) sont également ajustés pour conserver un graphe plus ou moins "sparse" : plus le flux est imposant, plus la compétition est nécessaire pour empêcher l'explosion du nombre de liaisons moyennes. Cela confère une **résilience** notable face aux fluctuations : même si de nouvelles entités se joignent en nombre, le SCN ne se "pollue" pas de milliers de liens faibles, car la compétition en élimine la plupart, ne laissant qu'un sous-ensemble plus pertinent.

Des **changements soudains** (par exemple, la disparition d'un cluster d'entités, ou l'arrêt de capteurs critiques) se gèrent aussi via l'inhibition et le recuit, qui aident à rompre certains liens qui n'ont plus de sens dans le nouveau contexte. Le **Chapitre 9** exposera des exemples concrets où l'on voit la **dynamique DSL** se "redresser" après un tel changement de structure, reconquérir un état stable où les clusters sont reconfigurés pour la nouvelle situation.

C. Distribution et Parallélisation

Dans une optique temps réel, on peut vouloir **distribuer** les calculs sur plusieurs nœuds ou threads. Déjà au chapitre 5-6, la notion d'échelle (scalabilité) et de multi-niveaux fut abordée. Le **Chapitre 9** prolongera cette vision pour l'apprentissage continu : fractionner le réseau en zones traitées parallèlement. Les méthodes d'optimisation présentées ici (chap. 7) — recuit, heuristiques, inhibition — peuvent alors s'exécuter de manière partiellement distribuée, moyennant une synchronisation épisodique pour éviter les incohérences globales. On retrouve des idées d'"îlots évolutionnaires" ou de "zones d'inhibition" reliant plusieurs blocs concurrentiels de ω .

Le paramétrage (taux η , coefficient τ , loi du recuit, γ pour l'inhibition) acquiert une dimension plus complexe, car la dynamique en flux, l'éventuel multi-threading, et la taille n imposent un calibrage fin. Le chap. 7 a déjà montré la sensibilité aux paramètres ; en temps réel sur de grands volumes, on doit souvent employer des **stratégies** adaptatives (7.1.2.3) afin d'éviter tant la stagnation que l'explosion des calculs.

Conclusion

Le **Chapitre 9** poursuivra la démarche amorcée en chap. 7 en l'appliquant à un **environnement temps réel** et à un **volume** potentiellement important d'entités :

- L'**adaptation en flux** (7.6) sera reprise, détaillant comment on insère de nouvelles entités ou on en retire, sans recomputations totales.
- Les **mécanismes** d'optimisation (recuit, heuristiques, inhibition modulée) trouveront un usage intensif pour préserver la **robustesse** lorsque les données évoluent.

- Les **questionnements** de distribution (multi-threads, zones distinctes) deviennent critiques dès lors que la dimension n gonfle et que le SCN doit rester “vivant” sans se figer dans un minima local.

Cette transition prépare à la mise en œuvre d'un **DSL** réellement **opérationnel** en contexte dynamique, consolidant les acquis de ce chapitre 7 pour un usage sur de **plus grands volumes** ou sous **perturbations** plus intenses, qui seront au cœur du propos de **Chapitre 9**.

7.10.3. Perspectives pour l'Optimisation Avancée

Dans les sections précédentes, nous avons principalement abordé le **recuit simulé**, les **heuristiques** (inhibition, clip, sparsification) et l'**apprentissage continu** comme leviers d'**optimisation** pour le **SCN** (Synergistic Connection Network). Cependant, d'autres **familles** d'algorithmes, souvent employées en métahéuristiques ou en apprentissage par renforcement, peuvent encore enrichir la **dynamique** du DSL. Leur intégration est facilitée par la **flexibilité** de la mise à jour $\omega_{i,j}$, laquelle autorise l'ajout (ou la substitution) de modules dans l'architecture SCN (Chap. 5).

7.10.3.1. Possibilités de Mélanger d'Autres Heuristiques Globales (PSO, Colony, etc.)

Les méthodes d'**optimisation** globales décrites tout au long du chapitre 7 (recuit simulé, heuristiques génétiques, multi-run) ne sont pas les seules à pouvoir s'intégrer à la dynamique du **DSL**. D'autres approches évolutives ou à base de “colonies d'agents” sont également envisageables pour **explorer** l'espace des pondérations $\{\omega_{i,j}\}$ et **franchir** les minima locaux. Les **algorithmes** tels que la **Particle Swarm Optimization (PSO)** ou les **Colonies de Fourmis (ACO)**, initialement conçus pour des problèmes de chemin minimal ou de fonctions continues, peuvent être adaptés à un **SCN** (Synergistic Connection Network), avec toutefois un soin particulier quant à la dimensionnalité potentiellement élevée et aux opérations de fusion avec la **règle** DSL de base. Les paragraphes suivants détaillent ces possibilités et les précautions associées.

A. Particle Swarm Optimization (PSO) et Espace ω

Il est possible de considérer la **matrice** $\Omega = \{\omega_{i,j}\}$ comme un unique **vecteur** de dimension $d \approx n(n - 1)/2$ (si le SCN est non orienté) ou n^2 (orienté). Un algorithme de **PSO** manipule alors plusieurs “particules”, chacune représentant une configuration $\Omega \in \mathbb{R}^d$. À chaque itération :

$$\Omega^{(p)}(t + 1) = \Omega^{(p)}(t) + v^{(p)}(t + 1),$$

où $v^{(p)}(t + 1)$ dépend de la meilleure solution locale et de la meilleure solution globale de la population. Les liens $\omega_{i,j}$ se voient réinterprétés comme les “coordonnées” du vecteur Ω . On définit une **fonction d'évaluation** :

$$\text{Fitness}(\Omega) = -\mathcal{J}(\Omega),$$

ou une transformation équivalente, de sorte que plus la fonction Fitness est élevée, plus l'énergie \mathcal{J} est faible. Les **particules** se déplacent donc dans cet espace à forte dimension en se calquant sur la meilleure solution rencontrée par l'essaim. Cette **métaphore** de la nuée de particules (PSO) peut rompre des attracteurs locaux en modifiant simultanément plusieurs composantes $\omega_{i,j}$.

Il convient cependant de noter que, pour un **grand** SCN (n important), la dimension $d \approx n^2$ peut rendre le **PSO** très coûteux. On préfère alors limiter l'espace (par ex. ne considérer qu'un sous-ensemble de liens ou un vecteur plus réduit décrivant la structure). Néanmoins, l'avantage réside dans la **répartition** de la recherche sur plusieurs particules (exploration globale) et dans la possibilité de **combine** la dynamique DSL locale (mis à jour itératif “descente ou ascension synergie”) avec un **régime PSO** sur un certain nombre d’itérations, afin d'éviter un minimum local.

B. Algorithmes de Colonies (Fourmis, Abeilles...)

Les **Colonies de Fourmis** (Ant Colony Optimization, ACO) furent historiquement proposées pour la résolution de chemin minimal (TSP, graphes divers). Leur principe repose sur le **dépôt** et l'**évaporation** de “phéromone” le long des chemins empruntés par des fourmis virtuelles. On peut transposer ce concept :

$$\omega_{i,j}(t) \leftrightarrow \text{Taux de phéromone sur le lien } (i,j),$$

et introduire une **évaporation** $\rho \omega_{i,j}(t)$ tout en renforçant $\omega_{i,j}$ quand des “fourmis” franchissent la liaison (i,j) dans un parcours jugé fructueux (basé sur \mathcal{J} ou un critère de cohésion). Chaque “fourmi” simule un **trajet** dans le SCN, choisissant les liens proportionnellement à $\omega_{i,j}$. Si la cohérence d’un tel trajet s’avère bonne (faible énergie, forte similarité), on “dépose” une **phéromone** additionnelle sur (i,j) , c’est-à-dire :

$$\omega_{i,j}(t+1) = \rho \omega_{i,j}(t) + \Delta_{\text{fourmis}}(i,j,t).$$

Cette dynamique peut coexister avec la **règle** DSL, ou la remplacer par périodes. Ainsi, au lieu d’ajuster $\omega_{i,j}$ via $\eta[S(i,j) - \tau \omega_{i,j}]$, on exécute un cycle “colonie” (fourmis virtuelles), puis on applique la “mise à jour phéromone” sur les liens traversés. Là encore, le but est de **parcourir** l’espace potentiel du SCN sous un angle global, contournant les minima locaux.

Les **algorithmes** de colonies d’abeilles, de chauves-souris ou d’autres heuristiques “nature-inspirées” se conforment à des principes analogues : explorer un **vaste** espace de solutions, mettre en concurrence ou en coopération divers “agents artificiels”, et renforcer dans ω les chemins ou configurations jugées plus “viables”.

C. Hybrides et Scénarios d’Usage

Les heuristiques globales (PSO, Fourmis, etc.) peuvent se **mélanger** avec :

Recuit simulé : on peut insérer un *terme stochastique* supplémentaire dans la dynamique ACO ou PSO. Les fourmis, par exemple, pourraient subir un bruit sur leur choix de chemin. De même, dans le PSO, on peut injecter un “petit recuit” sur la meilleure solution globale pour déverrouiller des stagnations.

Inhibition : on peut décider qu’un agent (une particule, une fourmi) ne peut “déposer” de phéromone au-delà d’un certain nombre de liaisons, ou que l’on constraint un noeud i à limiter son degré dans le contexte ACO. Cela revient à conjuguer un *terme* d’inhibition (7.4) avec la mécanique colonie (dépôt/évaporation).

DSL Basique : on peut opérer un “cycle DSL” local (descente selon $\eta[S(i,j) - \tau \omega_{i,j}]$) puis, toutes les K itérations, on lance un cycle “PSO global” (ou “Fourmis”) manipulant Ω de manière plus radicale. Ainsi, la partie DSL gère la convergence micro-locale, tandis que la partie heuristique agit en “macro-étapes”.

Conclusion

En plus du recuit simulé et des méthodes génétiques déjà mentionnées, il existe tout un **panel** d’heuristiques globales (PSO, Fourmis, colonies diverses) qu’il est possible de **transplanter** dans le cadre DSL, en considérant ω comme la “configuration” d’un espace à optimiser. Cette **métaphore** élargit l’ensemble des outils disponibles pour **sortir** des minima locaux et structurer un SCN de manière plus globale. Toutefois, la **dimension** potentiellement élevée (n^2 liens) et la cohabitation avec la **règle** DSL imposent des paramétrages soigneux, sous peine d’un coût numérique prohibitif. Pour autant, sur des cas ciblés, la synergy heuristique–DSL peut apporter un **gain** significatif en capacité d’exploration et en robustesse face aux structures complexes.

7.10.3.2. Intégration de Méthodes “Online” plus Sophistiquées (Bandits Contextuels, RL Avancé)

Les stratégies d’**optimisation** abordées dans ce chapitre 7 (recuit, heuristiques globales, inhibition, etc.) s’articulent naturellement avec des approches **en flux** (online learning) où l’on dispose d’un **signal** de récompense ou d’efficacité. Dans un cadre plus formel, on peut enrichir la mise à jour DSL au moyen de **méthodes** issues de la théorie des bandits

ou du renforcement (RL), conférant au **SCN** (Synergistic Connection Network) une capacité à **explorer** et **exploiter** en continu les liens ω .

Cette approche combine les **bandits contextuels**, qui optimisent le dilemme **exploration-exploitation** dans des environnements évolutifs, avec des **méthodes RL avancées** telles que le **Q-learning profond** et les modèles **actor-critic**. Ces techniques permettent d'ajuster dynamiquement les **liens, clusters et parties du réseau**, assurant ainsi une adaptation continue du **SCN (Synergistic Connection Network)** en fonction des interactions et de la récompense reçue.

A. Bandits Contextuels : Extension du Principe Multi-Armed

La théorie des **bandits multi-bras** (Multi-Armed Bandit) consiste à sélectionner une action parmi plusieurs, dans le but de **maximiser** la récompense espérée à long terme. Cette formulation initiale, souvent écrite sous la forme $\max_{a \in \mathcal{A}} \mathbb{E}[R(a)]$, suppose que la distribution de récompense $\mathbb{P}(R | a)$ est inconnue et doit être explorée au fil des itérations. Les **bandits contextuels** étendent ce cadre en tenant compte d'un **contexte** $\mathbf{c}(t)$ qui, à chaque étape t , influe sur la probabilité d'obtenir une récompense élevée pour un choix a .

Dans un **SCN** (Synergistic Connection Network), il est envisageable de transposer cette logique pour **piloter** la formation et le renforcement de liaisons $\omega_{i,j}$ de façon semi-supervisée. Plus précisément, la variable $\omega_{i,j}(t)$ reflète la tendance actuelle du réseau à maintenir le lien (i,j) ; son évolution est traditionnellement régie par la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

comme indiqué en **section 2.2.2**. En incorporant un **bandit contextuel**, on ajoute un mécanisme de **sélection d'action** et de **récompense** : à chaque itération, on choisit explicitement un certain lien (i,j) à renforcer (ou à affaiblir), puis on observe un signal extrinsèque évaluant la pertinence de ce choix.

Le **contexte** dans cette version hybride peut inclure des informations sur l'état local d'un nœud \mathcal{E}_i , par exemple son degré moyen $\sum_k \omega_{i,k}(t)$, sa synergie globale, ou encore la configuration partielle du sous-cluster dans lequel il évolue. Le **bandit** interprète ces variables contextuelles comme autant d'indicateurs susceptibles de prédire l'efficacité d'activer ou de désactiver un lien (i,j) . Le **chapitre 7.10.1.2** fournit un exemple d'une récompense extrinsèque R_t calculée à partir de la satisfaction d'une contrainte ou d'un objectif opérationnel, qui peut provenir d'une source extérieure au SCN (par exemple, le succès d'une mission robotique ou la performance d'un module de reconnaissance).

Dans ce cadre, on définit un ensemble d'**actions** reliées aux pondérations : choisir de renforcer (ou de diminuer) $\omega_{i,j}$ pour un certain couple (i,j) correspond à "tirer un bras" du bandit. La stratégie du bandit contextuel repose alors sur la mise à jour d'estimateurs de la récompense associée à chaque action $a \in \mathcal{A}$, conditionnellement au contexte $\mathbf{c}(t)$. Lorsque le SCN constate ex post que renforcer $\omega_{i,j}$ a rapporté un gain, il accroît la probabilité de reprendre cette action dans un contexte similaire.

La **dynamique DSL** s'enrichit ainsi d'un **pilotage global** : au lieu de se contenter de l'évolution locale décrite par $\eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}]$, on introduit un algorithme de sélection (de type ε -greedy, UCB, ou Thompson Sampling) qui désigne le lien (i,j) à mettre à jour en fonction du contexte. Ce lien reçoit alors un **renforcement** pondéré par l'observation de la récompense, illustrant la combinaison d'un mécanisme statistique (estimation des gains pour chaque action et contexte) et d'un mécanisme d'**auto-organisation** (la dynamique DSL) dans le même réseau.

Du point de vue des **avantages**, ce couplage permet au SCN de tenir compte d'indices externes, censés mesurer la qualité d'une liaison dans une situation particulière. Lorsque le contexte et la récompense sont bien définis, on oriente l'apprentissage des connexions vers les plus prometteuses pour la tâche globale, plutôt que de s'appuyer exclusivement sur la synergie interne S . Les **limites** tiennent à la complexité additionnelle : il faut un **module** de bandit capable de gérer un espace d'actions potentiellement grand (toutes les paires (i,j)) et un contexte multivarié représentant l'état du réseau. Le compromis entre exploration et exploitation se transpose également sur le plan de la formation des liaisons $\omega_{i,j}$.

Dans l'ensemble, l'**intégration** de bandits contextuels dans un SCN prolonge la démarche du DSL en introduisant un **signal** semi-supervisé ciblé sur chaque liaison. La conclusion de chaque "tirage de bras" avec feedback renforce les pondérations associées aux décisions les plus fructueuses dans le contexte courant, tout en maintenant la structure

localement adaptée par la règle de descente standard. On obtient dès lors un **équilibre** entre l'auto-organisation purement non supervisée (chapitres 2.2.2 et 2.2.3) et une approche plus dirigée, apte à optimiser une fonction de performance externe.

B. Approches de Renforcement (RL) Avancées

Il est possible de dépasser le cadre des bandits contextuels en intégrant, au sein d'un **SCN** (Synergistic Connection Network), des techniques de **renforcement** complètes, telles que le **Q-learning profond** ou des méthodes **actor-critic**. Contrairement au simple choix d'actions isolées (comme dans un bandit), ces algorithmes opèrent sur des **états** et **politiques** plus riches, en tenant compte de l'évolution temporelle du réseau et de la récompense cumulée.

Pour mettre en place un tel dispositif, on définit tout d'abord un **état** $\mathcal{E}(t)$ qui décrit la configuration du réseau à l'instant t . Une manière naturelle consiste à inclure la **matrice** $\{\omega_{i,j}(t)\}$ ou un **résumé** de celle-ci, par exemple le degré moyen de certains nœuds, la répartition cluster-wise, ou un indicateur global de synergie. L'ensemble des **actions** \mathcal{A} regroupe des modifications possibles sur la structure des liens, comme la mise à jour ponctuelle de $\omega_{i,j}$, le déclenchement d'un **recuit** partiel, l'ajustement du paramètre d'**inhibition** γ , ou encore le **redispatching** d'un sous-ensemble de connexions pour tester une configuration alternative. Une **récompense** $\mathcal{R}(t)$ est ensuite calculée en fonction d'objectifs plus larges que la simple minimisation de l'énergie interne : on peut, par exemple, prendre en compte la **cohésion** d'un cluster, la **réussite** d'une mission robotique ou tout autre critère externe.

L'agent de **renforcement** (ex. Q-learning, actor-critic, etc.) détermine alors, pour chaque étape temporelle, l'action à entreprendre selon la politique $\pi[\mathcal{E}(t)]$. Cette action agit directement ou indirectement sur la règle DSL locale, ce qui se formalise par la réécriture de l'équation de mise à jour,

$$\omega_{i,j}(t+1) = \underbrace{\omega_{i,j}(t)}_{\text{DSL local}} + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)] + \underbrace{\Omega_{\text{RL}}(\omega_{i,j}(t), \mathcal{R}(t))}_{\text{action RL}}$$

dans laquelle Ω_{RL} est un **terme correctif** dicté par l'algorithme de RL en fonction de la récompense courante $\mathcal{R}(t)$, de la configuration $\omega_{i,j}(t)$ et de la politique en vigueur. L'idée générale est de **mélanger** la dynamique auto-organisée du DSL, qui tend à descendre un potentiel d'énergie interne, avec un **feedback** extrinsèque plus global, piloté par un agent cherchant à optimiser un objectif à long terme.

La convergence peut alors être plus performante pour des tâches qui ne se résument pas à la minimisation locale d'une énergie $J(\omega)$. Par exemple, si la **section 7.3** discute la nécessité d'un recuit simulé pour franchir certains puits, cette forme de bruit peut être modulée automatiquement par l'agent RL, lequel détecte si la récompense stagne et augmente la dose de recuit ou bien modifie la force d'inhibition γ . Les **avantages** d'une telle hybridation résident dans la capacité à prendre en compte un but extrinsèque et à orienter la formation des clusters ou la stabilisation des liens dans une direction avantageuse pour ce but. Les **limites** ou **difficultés** proviennent de la haute dimension de l'espace des états (si chaque $\omega_{i,j}$ est considéré comme une variable indépendante) et du fait que la sélection d'actions "structurelles" peut changer radicalement la dynamique interne du réseau, ce qui rend l'apprentissage instable si l'on ne régule pas suffisamment la prise de décision.

En somme, l'ajout d'un **agent de Renforcement** complète la logique de **section 2.2.2** (descente DSL) et **section 7.4** (inhibition, compétition) pour englober des stratégies de contrôle plus élaborées. La politique RL devient un "**chef d'orchestre**" qui infléchit la descente locale en lui imposant des corrections liées à la récompense globale. Un tel système peut mieux s'adapter à des environnements changeants ou à des objectifs complexes, là où la simple auto-organisation par synergie ne suffirait pas à garantir l'optimisation d'un critère externe.

C. Enjeux et Intégration

L'un des points clés pour combiner un **SCN** (Synergistic Connection Network) avec des algorithmes de **bandits** ou de **renforcement** concerne la manière d'interfacer concrètement la dynamique DSL (Deep Synergy Learning) et le traitement du **signal** de récompense. Une possibilité est de dissocier l'évolution de la descente locale, régie par l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

d'une étape ultérieure où l'on évalue la performance $\mathcal{R}(t)$ ou la récompense associée. Dans cette seconde phase, on applique un algorithme de **RL** ou de **bandit** qui, en s'appuyant sur l'observation de $\mathcal{R}(t)$, modifie certaines pondérations ou contrôle un paramètre global (par exemple l'inhibition γ). Ce découplage temporel clarifie la contribution de la synergie **interne** $S(\mathcal{E}_i, \mathcal{E}_j)$ et du **feedback externe** $\mathcal{R}(t)$.

Un enjeu majeur tient à la **complexité** et à la **convergence** lorsque le SCN est de grande taille, puisque le nombre de liaisons $\omega_{i,j}$ peut atteindre l'ordre de n^2 . Un algorithme de Q-learning ou un bandit contextuel opère alors sur un espace d'états ou d'actions potentiellement gigantesque. Pour rendre le problème tractable, il est fréquent de limiter l'action de la **méthode RL** à un sous-ensemble de paramètres : plutôt que de manipuler chaque liaison $\omega_{i,j}$, l'agent peut se contenter de régler l'**inhibition** γ (voir **section 7.4**) ou le niveau de recuit (cf. **section 7.3**) à chaque étape, voire d'infléchir la mise à jour sur un sous-bloc de liens spécifiques. On obtient ainsi une structure hybride, où la descente DSL demeure en grande partie locale et auto-organisée, tandis qu'un module de renforcement pilote seulement quelques variables globales, ce qui diminue la charge computationnelle et facilite la convergence.

Un autre aspect crucial concerne le **compromis** entre l'auto-organisation intrinsèque et le **signal extrinsèque** fourni par le RL ou le bandit. La logique DSL valorise une cohérence locale fondée sur la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$, d'après les principes de la **section 2.2.1**. En introduisant un ajustement Δ_{RL} ou Δ_{bandit} susceptible de contrer la dynamique $\eta[S - \tau \omega_{i,j}]$, on peut compromettre la cohérence de clusters qui s'étaient formés spontanément selon la synergie interne. À l'inverse, si l'on ne permet pas à la récompense externe d'agir suffisamment, on risque de rester bloqué dans des configurations localement stables mais peu adaptées à un objectif plus global. Il faut alors calibrer la part de Δ_{RL} dans la mise à jour totale,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}(t) + \Delta_{\text{RL}}(t),$$

de manière à préserver une part d'**auto-organisation** tout en laissant à l'algorithme de renforcement (ou de bandit) la possibilité d'orienter la configuration finale selon la **récompense** $\mathcal{R}(t)$. Trouver cet équilibre reste délicat et peut exiger des expérimentations ou une analyse plus fine des courbes de convergence.

Conclusion (7.10.3.2)

Des **méthodes** plus sophistiquées “online” — bandits contextuels, RL avancés — peuvent s'**imbriquer** dans la mise à jour du **SCN**, renforçant ou modulant la dynamique DSL en fonction d'un **feedback** extrinsèque. Les bandits contextuels offrent un cadre *exploration–exploitation* pour choisir quels liens $\omega_{i,j}$ privilégier, tandis qu'un RL de type Q-learning permet à un “agent superviseur” d'**apprendre** une politique d'ajustement des pondérations. Ce couplage hybride ouvre la voie à des scénarios d'**apprentissage continu** encore plus interactifs, où le SCN ne se contente pas de l'auto-organisation, mais l'inscrit dans un **environnement** produisant des signaux de récompense. Cela peut accroître :

- La **flexibilité** : adaptation plus rapide aux changements,
- La **focalisation** sur l'objectif extrinsèque : car la synergie interne est *pondérée* par la performance mesurée,
- La **puissance** de la recherche de configurations, en combinant le local (DSL) et le global (RL ou bandit).

Comme pour les heuristiques globales (PSO, colonies), on doit toutefois maîtriser la **complexité** (n^2 liens) et définir un protocole clair d'interaction **DSL–RL** (ou DSL–bandit) pour assurer une convergence stable et un coût numérique soutenable.

7.10.3.3. Conclusion : La Flexibilité du DSL s'Adapte à Ces Méthodes pour Perfectionner la Formation Auto-Organisée de Clusters

Le **Deep Synergy Learning (DSL)**, tel que développé dans l'ensemble de ce chapitre (7.1 à 7.10), offre déjà une structure de mise à jour des pondérations $\omega_{i,j}$ s'appuyant sur la **synergie** et la **régulation** locale (inhibition, saturation, etc.). Cependant, cette dynamique « descente locale » peut se retrouver confrontée à des **minima** d'énergie partiels ou à des configurations sous-optimales qui ne se résolvent pas spontanément. Les **métaheuristiques** globales (PSO,

colonies de fourmis), les **méthodes** issues du **RL** (bandits, Q-learning), ou encore les **heuristiques** génétiques décrites tout au long de ce chapitre viennent se **greffer** à la mécanique DSL pour dépasser ces blocages et consolider la formation auto-organisée de *clusters* plus nets.

A. Rôle Central de la Modularity du DSL

La **structure** modulaire du DSL (voir chapitres précédents, autour de la compétition, de la localité, et de l'injection possible d'informations externes) facilite l'intégration d'**algorithmes** d'optimisation externes. Au lieu de rompre la logique interne de synergie $S(i,j)$, il est envisageable de :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \Psi(\mathcal{H}, \Omega, t),$$

composante DSL terme heuristique global

où $\Psi(\cdot)$ représente une “correction” apportée par l'**algorithme** global, qu’il s’agisse d’une **variante** de recuit simulé, d’une **mutation** génétique, ou d’un **ajustement** inspiré d’un bandit contextuel. Cette cohabitation entre la mise à jour **locale** (descente sur S) et la *métacorrection* (composante globale) confère au SCN la capacité de parcourir plus largement l'espace des configurations, sans se priver de la **plasticité** intrinsèque du DSL.

B. Flexibilité et Complémentarité des Méthodes

La multiplicité des **méthodes** évoquées dans la section 7.10.3 (PSO, colonies, bandits) souligne que le DSL se prête à des **stratégies** d'optimisation très différentes : on peut instancier un *swarm* de particules dans l'espace des ω , ou recourir à une *colonie* de fourmis qui “déposent” de la phéromone sur certains liens, ou encore utiliser un bandit contextuel pour piloter un sous-ensemble de liaisons. Chacune de ces approches propose un **mode** d'exploration global distinct, venant compléter la synergie locale définie par S .

La **combinaison** recuit + heuristique, ou heuristique + inhibition, apporte encore d'autres degrés de liberté. Sur le plan **mathématique**, on écrit souvent la mise à jour de la pondération $\omega_{i,j}$ comme :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}(\omega_{i,j}(t)) + \Delta_{\text{Global}}(\omega_{i,j}(t), \Omega_{\text{all}}),$$

où Δ_{Global} tient compte d'une **démarche** PSO, ACO, ou RL, assurant un “sursaut” possible hors du bassin local. Cette synergie entre la dynamique DSL locale et les interventions globales maintient un **équilibre** entre la *compétition* (inhibition, filtrage local) et l'**exploration** (recherche globale, recuit, heuristiques) dans un large espace $\{\omega_{i,j}\}$.

C. Poursuite du DSL dans les Chapitres Suivants

Le **Chapitre 8** (DSL Multimodal) et le **Chapitre 9** (Évolutions Temps Réel) vont s'appuyer directement sur ces enseignements :

- La **multimodalité** (chap. 8) peut nécessiter une exploration plus ample de la configuration, car le risque de minima locaux “mono-canal” est fort : la recutilisation, l'inhibition, ou les heuristiques globales apportent alors la “secousse” ou la “filtration” indispensable.
- L'**apprentissage continu** (chap. 9) requiert une robustesse face aux flux et changements : on mobilise l'**inhibition** (gestion de la densité quand beaucoup d'entités nouvelles apparaissent), le **recuit** partiel (pour surmonter un blocage local), ou la **sparification** par heuristiques (réduire le nombre effectif de liaisons manipulées au fil du temps).

Cette **flexibilité**, qui accueille et articule des **méthodes** (PSO, bandits, colonie) au sein d'un cadre DSL déjà compétitif et localement auto-organisé, est un atout majeur pour perfectionner la **formation auto-organisée de clusters** et consolider la convergence globale.

Conclusion (7.10.3.3)

La **souplesse** du DSL (Deep Synergy Learning), fondée sur une mise à jour localement compétitive ($\eta[S(i,j) - \tau \omega_{i,j}]$) et ouverte à l'ajout de termes d'**inhibition**, de **recuit**, ou de **feedback** RL, lui permet de s'adapter harmonieusement à

d'autres **méthodes** globales telles que la **PSO** (Particle Swarm), les **colonies** de fourmis ou les **bandits contextuels**. L'ensemble de ces outils offrent la capacité de :

- **Passer outre** les minima locaux via des approches exploratoires (PSO, ACO, recuit, bandits),
- **Canaliser** la densité et la compétition grâce à l'inhibition dynamique,
- **Combiner** la descente DSL locale (qui préserve la synergie S) avec un module global plus "intelligent" ou stochastique,
- **Renforcer** la pertinence des clusters émergents dans des scénarios complexes (multimodalité, flux, apprentissage en ligne).

Ce **mélange** d'approches se présente ainsi comme un moyen de **perfectionner** la formation auto-organisée de clusters, permettant au SCN d'aboutir à des configurations plus "globalement" optimales, sans sacrifier la rapidité et la localité intrinsèque au DSL. Cette flexibilité constitue l'un des fils conducteurs pour les chapitres ultérieurs (Chap. 8 sur la **multimodalité**, Chap. 9 sur le **temps réel**), où ces méthodologies se concrétisent dans des situations encore plus exigeantes.

Chapitre 8 : DSL Multimodal : Fusion de la Vision, du Langage et des Sons

Chapitre 8 : DSL Multimodal : Fusion de la Vision, du Langage et des Sons	1119
8.1. Introduction	1121
8.1.1. Contexte.....	1121
8.1.2. Objectifs et Intérêt	1123
8.1.3. Plan du Chapitre	1138
8.2. Principes Généraux de la Fusion Multimodale dans le DSL.....	1142
8.2.1. Notion de Synergie Inter-Modale	1142
8.2.2. Avantages du DSL pour la Fusion.....	1146
8.2.3. Défis et Écueils	1151
8.3. Modélisation des Entités Visuelles.....	1157
8.3.1. Représentations d'Images	1157
8.3.2. Synergie Visuelle-Visuelle	1161
8.3.3. Liens avec d'Autres Modalités.....	1166
8.4. Modélisation des Entités Linguistiques.....	1171
8.4.1. Représentations Textuelles.....	1171
8.4.2. Synergie Langage-Langage	1176
8.4.3. Liens Texte–Autres Modalités	1180
8.5. Modélisation des Entités Sonores	1186
8.5.1. Représentations Audio.....	1186
8.5.2. Synergie Audio-Audio	1191
8.5.3. Fusion Audio–Vision ou Audio–Texte.....	1196
8.6. Construction d'un SCN Multimodal Unique	1203
8.6.1. Entités Hétérogènes dans un Même Réseau	1203
8.6.2. Densité et Parcimonie	1209
8.6.3. Synergie Tri-Modal (Vision–Langage–Audio)	1215
8.7. Dynamique d'Auto-Organisation en Contexte Multimodal.....	1223
8.7.1. Mise à Jour des Pondérations	1223
8.7.2. Émergence de Clusters Multimodaux	1230
8.7.3. Oscillations ou Confusions possibles	1240
8.8. Apports en Applications Concrètes.....	1246
8.8.1. Annotation d'Images par le Langage	1246
8.8.2. Reconnaissance Audio–Visuelle	1250
8.8.3. IA Symbolique–Sub-Symbolique en Multimodal	1255
8.9. Aspects Évolutifs et Temps Réel.....	1259
8.9.1. Flux Multimodal en Continu.....	1259
8.9.2. Convergence et Stabilisation	1262

8.9.3. Suivi et Visualisation	1266
8.10. Limites, Défis et Pistes de Recherche	1271
8.10.1. Complexité	1271
8.10.2. Qualité de la Synergie	1274
8.10.3. Sécurité et Biais	1277
8.10.4. Recherche Future	1280
8.11. Conclusion et Ouverture	1285
8.11.1. Récapitulatif du Chapitre	1285
8.11.2. Liens vers Chapitres Suivants	1288
8.11.3. Synthèse sur la Valeur du DSL Multimodal	1292

8.1. Introduction

L'un des **intérêts majeurs** du Deep Synergy Learning (DSL) réside dans sa capacité à traiter **simultanément** plusieurs types de données (visuelles, textuelles, sonores) et à faire émerger des **synergies** entre ces différentes modalités. Dans les chapitres antérieurs :

- Le **Chapitre 3** a posé les bases de la **représentation** d'entités (vecteurs, symboles, hybrides).
- Le **Chapitre 4** a explicité la **dynamique** auto-organisée (mise à jour ω), tandis que le **Chapitre 5** a présenté l'**architecture SCN** et ses modules.
- Le **Chapitre 6** s'est concentré sur l'**apprentissage multi-échelle** et la fractalité, et le **Chapitre 7** a introduit diverses **méthodes d'optimisation** (recuit, inhibition avancée, etc.).

À présent, nous allons **montrer** comment le DSL peut intégrer et **fusionner** plusieurs canaux de données (images, textes, sons), permettant de dégager une structure **multimodale** riche et auto-organisée.

8.1.1. Contexte

8.1.1.1. Rappel

Dans les chapitres précédents, plusieurs **fondements** ont été exposés afin de poser les bases du **Deep Synergy Learning** (DSL) et de son architecture en **Synergistic Connection Network** (SCN). En **Chapitre 3**, la **représentation** des **entités** a été abordée ; il s'agit de décrire chaque entité \mathcal{E}_i au moyen soit d'un **vecteur** (par exemple un *embedding* neuronal) soit d'un **formalisme** symbolique (par exemple un ensemble de règles logiques), voire d'une structure **hybride** combinant aspects numériquement continus et sémantique symbolique. Un **DSL multimodal** mobilise alors divers espaces, notamment un **espace visuel**, comprenant des extraits issus de réseaux de neurones de type **ResNet** ou **VGG**, ainsi que des points clés **SIFT**. Il intègre également un **espace textuel**, basé sur des embeddings tels que **Word2Vec**, **BERT** ou **GPT**, et un **espace audio**, utilisant des représentations comme les **spectrogrammes**, les **MFCC** ou des **représentations profondes**.

Dans le **Chapitre 4**, la **dynamique auto-organisée** a été analysée. Les pondérations $\omega_{i,j}$ reliant deux entités \mathcal{E}_i et \mathcal{E}_j évoluent selon la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

où $\eta > 0$ est le **taux d'apprentissage** et $\tau > 0$ est un **facteur de décroissance**. Cette **mise à jour** exploite la **fonction de synergie** $S(i,j)$. Dans un **contexte multimodal**, la **similarité** n'est plus exclusivement un score intra-canal (image–image, texte–texte), mais peut devenir un **score cross-modal** (image–texte, audio–image, etc.), ce qui entraîne une **auto-organisation** plus riche.

Le **Chapitre 5** a introduit l'**architecture SCN**. Le cœur de ce dispositif gère les **pondérations** $\omega_{i,j}$ et collabore avec différents **modules** (par exemple un **Module Synergie** ou un **Module Inhibition**) pour ajuster les connexions. En **multimodal**, les **modules** se spécialisent en fonction des types de données traitées. Un **Module Synergie** dédié à la **vision-langage** calcule la fonction de similarité $S(\text{image}, \text{texte})$, tandis qu'un autre module prend en charge la correspondance entre **audio** et **image** en évaluant $S(\text{audio}, \text{texte})$. Chaque module applique alors une **mesure de similarité** spécifique, optimisée pour son propre canal d'information.

En **Chapitre 6**, le concept d'**apprentissage multi-échelle** a été exploré. Les **entités** peuvent se **regrouper** en **clusters** locaux, puis se rassembler en **macro-clusters** représentant des thèmes plus vastes. Dans un **DSL multimodal**, cette **hiérarchie** sert à structurer des **images**, des **textes** et des **signaux audio** autour de **concept**s ou d'événements partagés.

Enfin, le **Chapitre 7** a présenté différentes **méthodes d'optimisation** visant à éviter les **minima locaux**, à gérer l'**inhibition** ou la **saturation** et à permettre un **apprentissage continu**. Une fois le caractère **multimodal** intégré, la **dimension** du réseau s'accroît ; il devient plus fréquent que le SCN contienne de multiples **entités** réparties dans plusieurs **canaux** (visions, textes, sons), ce qui requiert d'autant plus de **régulation** et de **compétition** entre liaisons afin de maintenir un **niveau cohérent de pondérations** $\omega_{i,j}$.

8.1.1.2. Ici, on se focalise sur la fusion multimodale : comment le DSL peut agréger des données provenant de la vision, du langage et de l'audio pour en extraire des synergies plus riches.

Le **Deep Synergy Learning (DSL)** n'est pas circonscrit à un seul type de données ; il se prête, au contraire, à un traitement **multimodal**, dans lequel plusieurs canaux (tels que la vision, l'audio et le texte) sont combinés à l'intérieur d'un **Synergistic Connection Network (SCN)** unique. Cette capacité s'avère particulièrement pertinente dans les applications qui requièrent de reconnaître et de corrélérer des informations **hétérogènes** : on pense, par exemple, à l'analyse audio-visuelle d'une scène (où des signaux image et son doivent être reliés), à la recherche de correspondances entre une description textuelle et des images (systèmes de vision-langage), ou encore à la constitution d'assistants conversationnels capables d'exploiter des flux **vocaux**, **visuels** et **textuels** simultanément. Dans tous ces contextes, le DSL se démarque par la possibilité de **fusionner** plusieurs **modalités** de façon naturelle, en tirant parti de sa **dynamique auto-organisée** et de sa **fonction de synergie**.

Dans cette optique, il est utile de décrire comment un SCN peut accueillir un **ensemble** d'entités $\{\mathcal{E}_i\}_{i=1}^n$ issues de canaux distincts. Chaque entité \mathcal{E}_i peut correspondre à un **extrait visuel**, un **descripteur audio** ou une **unité textuelle** (mot, token, phrase). Pour tenir compte de cette hétérogénéité, on introduit des représentations adéquates : un objet visuel $\mathcal{E}_i^{(\text{visuel})}$ peut-être encodé sous la forme d'un vecteur $\mathbf{x}_i^{(\text{visuel})} \in \mathbb{R}^{d_v}$ (par exemple, un embedding extrait d'un réseau neuronal convolutif tel que ResNet ou VGG) ; un objet audio $\mathcal{E}_j^{(\text{audio})}$ peut-être décrit par un vecteur $\mathbf{x}_j^{(\text{audio})} \in \mathbb{R}^{d_a}$ (typiquement, des MFCC ou un embedding profond) ; un segment textuel $\mathcal{E}_k^{(\text{texte})}$ peut être associé à un embedding $\mathbf{x}_k^{(\text{texte})} \in \mathbb{R}^{d_t}$ (du type Word2Vec, GloVe, BERT, etc.). Même si ces espaces ont des dimensions différentes, le **DSL** autorise leur "collage" en un **SCN** unique, grâce aux propriétés d'extension et de construction de la fonction de synergie (voir Chapitres 2 et 3).

La **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ revêt alors plusieurs formes selon que \mathcal{E}_i et \mathcal{E}_j appartiennent ou non au même **canal**. On distingue les **similarités intra-modales**, comme la distance euclidienne inversée ou la similarité cosinus entre deux entités visuelles (image–image) ou deux entités audio (spectrogramme–spectrogramme). On distingue aussi les **corrélations inter-modales** (image–texte, audio–image, texte–audio), qui nécessitent un mécanisme ou un module spécifique pour calculer un score de compatibilité. Sur le plan mathématique, on peut définir une fonction globale

$$S(i, j) = \begin{cases} S_{\text{intra}}(\mathbf{x}_i^{(\text{visuel})}, \mathbf{x}_j^{(\text{visuel})}), & \text{si } i, j \text{ sont tous deux du canal visuel,} \\ S_{\text{cross}}(\mathbf{x}_i^{(\text{visuel})}, \mathbf{x}_j^{(\text{audio})}), & \text{si } i, j \text{ appartiennent à des canaux différents,} \\ \dots & (\text{autres cas : texte–image, texte–audio, etc.}). \end{cases}$$

Chacun de ces termes peut être implémenté par une **distance** ou une **similarité** adaptée (ex. distance cosinus, corrélation basée sur un espace latent partagé, kernel RBF, etc.), d'où l'on dérive un score entre [0,1] ou \mathbb{R}^+ .

Dans un **SCN** multimodal, la règle de mise à jour,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j)\tau \omega_{i,j}(t)],$$

reste inchangée. Toutefois, la présence de synergies cross-modales multiplie la variété des liens susceptibles d'émerger ou de se renforcer. Si une image $\mathcal{E}_i^{(\text{visuel})}$ montre une scène de conversation et qu'un segment $\mathcal{E}_k^{(\text{audio})}$ capture des voix humaines, on peut observer un **renforcement** progressif du poids $\omega_{i,k}$ dès lors que la corrélation audio–visuelle est élevée (temporellement et sémantiquement). De même, si un **mot** $\mathcal{E}_\ell^{(\text{texte})}$ décrit exactement l'objet visible dans $\mathcal{E}_i^{(\text{visuel})}$, un score de similarité "texte–image" élevé conduira à un $\omega_{i,\ell}$ plus fort. C'est la co-occurrence ou la **cohérence** entre modalités qui guide la consolidation ou l'extinction des liaisons.

Le principal **bénéfice** de cette fusion multimodale consiste à **capturer** des correspondances plus richement sémantiques. Un seul canal, par exemple le canal visuel, ne peut pas nécessairement lever toutes les ambiguïtés (un même type d'objet peut apparaître dans des contextes différents). Mais si, en parallèle, la piste audio ou la piste textuelle donne des indices compatibles (bruit caractéristique de l'objet, ou champ lexical qui décrit la scène), la **dynamique** du DSL va “coller” ces entités ensemble et former des **clusters** multimodaux qui représentent des objets ou des événements plus finement définis. Au niveau mathématique, ce principe correspond à la superposition de scores de synergie provenant de différents canaux, laquelle amplifie les liens réellement fiables et écarte les couplages purement fortuits dans un canal donné.

Un autre aspect majeur est la **constitution** de “macro-clusters” hiérarchiques dans un **DSL** à plusieurs échelles (voir Chapitre 6). Au niveau le plus fin, on observe des associations ponctuelles (une région d'image, un motif audio, un mot). Progressivement, l'auto-organisation agrège ces sous-ensembles en **sous-réseaux** qui encapsulent un concept ou un événement multimodal (par exemple, “conversation téléphonique dans un véhicule” peut être un macro-cluster regroupant : un extrait audio de voix, le bruit d'un moteur, des images de passagers, et des mots-clés associés). La **formation** de tels macro-clusters tire parti du fait que chaque entité **hérite** des liens accumulés dans ses canaux respectifs, favorisant l'émergence d'une sémantique globale.

Sur le plan **algorithmique**, la fusion multimodale dans le DSL ne requiert pas de surcoût conceptuel considérable : on conserve la même règle locale de mise à jour $\omega_{i,j}$, mais on y ajoute la capacité de **calculer** $S(i,j)$ pour des paires (i,j) appartenant à des canaux différents. Dès lors, la structure du réseau évolue pour faire **apparaître** des liaisons cross-modales, parfois renforcées par une règle de parsimonie ou d'inhibition (voir Chapitre 7) qui écarte les liaisons trop faibles. Il suffit de paramétrier les modules de **similarité** ou de **corrélation** adaptés à chaque couple de modalités (image–image, audio–audio, texte–texte, image–audio, image–texte, audio–texte, etc.), éventuellement en ponderant leur influence respective (α_{image} , α_{audio} , α_{texte}) selon l'importance souhaitée.

La **conclusion** est que le **DSL**, grâce à la plasticité de ses pondérations $\omega_{i,j}$ et à la souplesse de sa fonction de synergie, fournit un **cadre** naturel pour la **fusion multimodale**. L'intégration de canaux multiples enrichit notamment la sémantique des clusters émergents, ceux-ci devenant capables de capturer non seulement la proximité intra-canal, mais aussi les **liens** croisés (image–audio, texte–image, audio–texte, etc.). Les sections suivantes de ce chapitre (Chap. 8) détailleront divers **exemples** où le DSL est appliqué à des tâches multimodales concrètes, notamment la **reconnaissance de scène audio-visuelle**, l'**association texte–image** ou encore la **classification** de séquences vidéo par l'analyse simultanée du flux visuel, du flux sonore et des éventuelles annotations textuelles. Ces scénarios illustreront comment la **dynamique** auto-organisée, couplée à un mécanisme de **synergie** multisource, permet d'extraire des **représentations** plus robustes et plus expressives qu'un traitement unicanal isolé.

8.1.2. Objectifs et Intérêt

Dans le contexte du **DSL** (Deep Synergy Learning) appliqué aux données multimodales (chap. 8), l'objectif est de **fusionner** plusieurs sources d'information — images, textes, sons, capteurs divers — dans un **même SCN** (Synergistic Connection Network). On souhaite non seulement traiter ces modalités séparément, mais surtout **exploiter** la dynamique du DSL pour créer des **liaisons synergiques** entre elles, permettant l'émergence de clusters ou de structures qui reflètent les **corrélations** (ou complémentarités) entre objets visuels, mots-clés, signaux audios, etc. Les **sections** suivantes (8.1.2.2, 8.1.2.3) détailleront comment cette approche peut déboucher sur des **applications** concrètes (annotation automatique, recherche audio-visuelle, etc.).

8.1.2.1. Gérer simultanément plusieurs modalités (image, texte, sons) dans un même SCN

L'un des enjeux majeurs dans les approches multimédias actuelles est de parvenir à **combiner** et à **corréler** plusieurs types de données (images, textes, sons, etc.) au sein d'un même cadre d'apprentissage. Le **Deep Synergy Learning (DSL)**, via sa conception de **Synergistic Connection Network (SCN)**, fournit une structure adaptée à cette fusion, en

permettant d'intégrer dans un seul et même réseau des entités issues de canaux hétérogènes. Les pondérations $\omega_{i,j}$ au sein d'un SCN peuvent en effet relier des entités visuelles, des segments audios ou des tokens textuels, s'il s'avère que les **synergies** calculées entre elles sont élevées.

Notion de Multi-Modalité et Enjeux

Dans de nombreux scénarios concrets (classification d'images avec annotation textuelle, analyse audio-visuelle, systèmes de recommandation contextuelle, etc.), on dispose d'entités \mathcal{E}_i provenant de canaux très variés : un ensemble $\{\mathcal{E}_i^{(\text{image})}\}$ décrit les caractéristiques visuelles d'un objet ou d'une scène, un ensemble $\{\mathcal{E}_i^{(\text{texte})}\}$ encode des segments linguistiques ou des mots-clés, et un autre ensemble $\{\mathcal{E}_i^{(\text{audio})}\}$ représente des descripteurs sonores (spectrogrammes, MFCC, embeddings profonds, etc.). Sur le plan mathématique, chaque modalité peut être associée à son propre espace vectoriel : on parle ainsi de $\mathbf{x}_i^{(\text{image})} \in \mathbb{R}^{d_{\text{img}}}$ pour la composante visuelle, $\mathbf{x}_i^{(\text{texte})} \in \mathbb{R}^{d_{\text{txt}}}$ pour la composante linguistique, ou encore $\mathbf{x}_i^{(\text{audio})} \in \mathbb{R}^{d_{\text{aud}}}$ pour la partie sonore. Le défi, pour le SCN, est de **coller** ces espaces au sein d'un unique graphe $\{\omega_{i,j}\}$, dans lequel chaque nœud \mathcal{E}_i peut entretenir des liaisons avec des entités issues de la même modalité ou d'une autre.

Cette fusion multimodale se justifie par le fait que de nombreux **concepts** ou **événements** ont une existence multi-canal (par exemple, une vidéo avec son commentaire textuel et sa piste audio). En traitant l'image, le texte et l'audio dans des systèmes totalement séparés, on risquerait de rater les corrélations qui apparaissent pourtant de manière évidente (le son du piano et l'image d'un clavier, le mot "chien" et la photo d'un chien, etc.). Le **DSL** permet, au contraire, d'**auto-organiser** un réseau unique où chaque lien $\omega_{i,j}$ représente la pertinence d'associer deux entités, même si elles proviennent de modalités différentes.

Construction de la Synergie Inter-Modal

Le point délicat réside dans la **définition** de la fonction de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ lorsqu'on compare des entités issues d'espaces différents (image vs. texte, audio vs. image, texte vs. audio, etc.). Au sein du Chapitre 2 (sections 2.2.1.2 et 2.2.1.3), on a déjà vu comment "coller" des mesures de similarité ou d'information mutuelle sur des ensembles hétérogènes. Concrètement, on introduit souvent un **espace latent partagé** où l'on projette chacune des entités. On définit alors, par exemple,

$$S(\mathcal{E}_i^{(\text{image})}, \mathcal{E}_j^{(\text{texte})}) = \text{similarité}\left(\phi_{\text{img}}(\mathbf{x}_i^{(\text{image})}), \phi_{\text{txt}}(\mathbf{x}_j^{(\text{texte})})\right),$$

où ϕ_{img} et ϕ_{txt} sont des fonctions (le plus souvent neuronales) qui permettent de projeter respectivement l'image et le texte dans un espace vectoriel commun, comme \mathbb{R}^d . Une fois ce **score** défini, la synergie $S(i, j)$ alimente la mise à jour des pondérations $\omega_{i,j}$. Chaque couple d'entités $(\mathcal{E}_i^{(\text{image})}, \mathcal{E}_j^{(\text{texte})})$ voit ainsi sa liaison renforcée si les descripteurs latents sont jugés suffisamment proches ou corrélés.

De la même façon, pour l'audio et l'image, on peut concevoir un score de correspondance temporelle ou sémantique, en appuyant la similarité sur des méthodes de co-occurrence (scènes vidéo–audio) ou d'embeddings partagés. Le SCN évolue alors simultanément selon la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)],$$

ce qui autorise la naissance de liens forts entre canaux différents lorsque la corrélation est considérée comme robuste.

Intégration dans un SCN Unique et Émergence de Clusters Multimodaux

Une fois la synergie inter-modale (et intra-modale) spécifiée, l'ensemble des entités visuelles, textuelles et audio est **indexé** dans un **même** graphe $\{\omega_{i,j}\}$. Les nœuds \mathcal{E}_i n'ont plus besoin d'être circonscrits à un seul type de feature : on peut donc se retrouver avec, par exemple, $\mathcal{E}_1^{(\text{img})}, \mathcal{E}_2^{(\text{txt})}, \mathcal{E}_3^{(\text{audio})}$ et ainsi de suite, tous intégrés dans la même matrice de poids $\{\omega_{i,j}\}$. La dynamique du DSL fait que les pondérations $\omega_{i,j}$ se renforcent s'il existe une grande similarité (intra-modale) ou une grande compatibilité (inter-modale). Progressivement, on observe la **formation** de clusters dans lesquels se mêlent des entités de différentes natures mais renvoyant à un même concept ou à un même événement.

Ainsi, un “thème” audio-visuel-texte peut émerger, où figurent : une image d’un chien, un segment audio d’abolement, et le mot “chien”/“dog”.

Cette **émergence de clusters multimodaux** profite à de multiples applications. D’abord, elle fournit un outil puissant pour la **recherche d’information** : un mot-clé textuel peut se retrouver dans le même groupe qu’un extrait sonore et une image thématiquement liée. Ensuite, elle facilite la **classification** ou l'**annotation** automatiques, puisqu’on identifie rapidement qu’un ensemble “image + mot + son” correspond à un même concept. Enfin, cela ouvre la voie à des **scénarios cognitifs** plus évolués (voir Chapitre 9), où un agent conversationnel exploite simultanément des signaux visuels et des indices textuels pour mieux répondre, le SCN assurant la correspondance et la consolidation des liens entre ces flux.

Problèmes de Complexité et Normalisation

Naturellement, l’intégration de plusieurs modalités fait croître la **taille** du réseau (nombre de nœuds) et, potentiellement, la complexité $O(n^2)$ des connexions $\omega_{i,j}$. Sur le plan algorithmique, il peut donc être nécessaire de **filtrer** en amont les couples d’entités trop éloignés, ou de restreindre la mise à jour aux plus proches voisins, pour éviter la saturation en ressources (voir Chapitre 7, section sur l’équilibre entre densité et parsimonie). De plus, il convient de **normaliser** la fonction de synergie S afin qu’aucune modalité ne domine trop les autres. Dans certains cas, on introduit des coefficients α_{img} , α_{txt} , α_{aud} pour pondérer l’influence de chaque canal ; on peut également calibrer les échelles de similarité (entraînement d’un réseau contrastif, usage de kernel RBF, etc.) pour harmoniser le traitement de chaque couple modal.

Conclusion

L’intégration simultanée de plusieurs modalités (image, texte, audio) à l’intérieur d’un **même SCN** constitue l’une des forces du **DSL**. Cette approche dépasse les limites d’un apprentissage unicanal, en laissant la **dynamique auto-organisée** détecter et consolider, localement, les **corrélations** apparaissant entre différents types d’informations. Les liens $\omega_{i,j}$ se mettent à jour en se fondant sur la similarité (ou la compatibilité) **intra-modale** et **inter-modale**, si bien que des **clusters** plus riches surgissent naturellement, intégrant plusieurs canaux au service d’une sémantique plus globale. Dans la suite (sections 8.1.2.2 et 8.1.2.3), on détaillera davantage la façon dont la dynamique DSL, couplée à des embeddings ou à des méthodes de projection latente, consolide ces **clusters** multimodaux et les exploite pour différentes **applications** (annotation d’images par le texte, reconnaissance audio-visuelle, etc.).

8.1.2.2. Utiliser la dynamique DSL pour faire émerger des clusters ou liens synergiques entre modalités (objets visuels, mots, sons)

Lorsque l’on traite simultanément des données **visuelles**, **textuelles** et **sonores**, l’enjeu consiste à **faire ressortir** les correspondances et les regroupements les plus pertinents à travers ces trois canaux. Le **Deep Synergy Learning (DSL)**, grâce à sa mécanique d’**auto-organisation** et à sa **fonction de synergie** appliquée à un **Synergistic Connection Network (SCN)** unique, offre une **méthode** systématique pour encourager la **création** de liens cohérents entre des éléments multimodaux et, de ce fait, **faire émerger** des **clusters** sémantiquement riches (ex. relier l’image d’un chat, le mot “chat” et un son de miaulement).

A. Principes Généraux : Dynamique DSL en Contexte Multimodal

Dans un environnement **multimodal**, on définit un ensemble $\{\mathcal{E}_i\}_{i=1}^n$ rassemblant toutes les entités possibles :

- Des **objets visuels** (ou régions d’images) annotés ou détectés,
- Des **tokens** ou **segments textuels** (mots, chunks de phrases, etc.),
- Des **caractéristiques** ou **extraits** sonores (spectrogrammes, MFCC, événements acoustiques, etc.).

Ces entités cohabitent dans un **même SCN**, où chaque nœud \mathcal{E}_i possède une pondération $\omega_{i,j}$ avec chaque autre nœud \mathcal{E}_j . Le nombre total d’entités n peut être élevé, rendant critique la question de la **complexité** (voir Chapitre 7 pour les techniques de parsimonie et de filtrage).

Pour calculer la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$, on prend en compte la **nature** de \mathcal{E}_i et \mathcal{E}_j . Quand on compare deux entités **visuelles**, on utilise une similarité intra-canal (distance cosinus, distance euclidienne inversée sur les embeddings visuels, etc.). Quand on compare un objet visuel à un **mot**, ou un segment audio à un **mot**, on recourt à des méthodes “cross-modales” (ex. projection dans un espace latent partagé, ou évaluation de co-occurrences temporelles).

$$S(\mathcal{E}_i, \mathcal{E}_j) = \begin{cases} \text{similarité_visuelle}(\mathbf{x}_i^{(\text{visuel})}, \mathbf{x}_j^{(\text{visuel})}), & (\text{intra-image}) \\ \text{similarité_cross}(\mathbf{x}_i^{(\text{visuel})}, \mathbf{x}_j^{(\text{texte})}), & (\text{image-texte}) \\ \text{co-occurrence_audio}(\mathbf{x}_i^{(\text{son})}, \mathbf{x}_j^{(\text{texte})}), & (\text{audio-texte}) \\ \dots \end{cases}$$

Une fois $S(i, j)$ défini, on l’emploie pour la mise à jour de $\omega_{i,j}$.

La **dynamique** auto-organisée du DSL s’exprime par la règle habituelle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Ainsi, si la **cohérence** (ou co-occurrence) entre \mathcal{E}_i et \mathcal{E}_j est élevée (i.e. $S(i, j)$ grand), leur liaison $\omega_{i,j}$ aura tendance à **se renforcer** et, au fil des itérations, à devenir un lien stable du réseau. À l’inverse, si la corrélation inter- ou intra-modale reste faible, la liaison $\omega_{i,j}$ **décroît** progressivement.

B. Formalisation Mathématique Simplifiée

La présente section propose une extension du **Deep Synergy Learning (DSL)** au cas de données multimodales, en regroupant trois catégories d’entités : des entités **visuelles**, des entités **textuelles** et des entités **sonores**. Cette modélisation prolonge les principes de la **section 2.2.1** (définition des entités et de la fonction de synergie) et de la **section 2.2.2** (mise à jour des pondérations ω) en précisant comment un **SCN** (Synergistic Connection Network) peut traiter de multiples modalités de données dans un même espace d’apprentissage auto-organisé.

Pour introduire la pluralité des modalités, on se donne trois ensembles :

$$\begin{aligned} \mathcal{V} &= \{\mathcal{V}_1, \dots, \mathcal{V}_p\} \quad (\text{entités visuelles}), & \mathcal{T} &= \{\mathcal{T}_1, \dots, \mathcal{T}_q\} \quad (\text{entités textuelles}), & \mathcal{S} \\ &= \{\mathcal{S}_1, \dots, \mathcal{S}_r\} \quad (\text{entités sonores}). \end{aligned}$$

Le **réseau** complet de n entités, noté $\mathcal{E} = \mathcal{V} \cup \mathcal{T} \cup \mathcal{S}$, satisfait ainsi $n = p + q + r$. Chaque entité \mathcal{E}_i appartient à l’une de ces trois modalités. On représente ensuite les **pondérations** par une matrice ω de taille $n \times n$, dont chaque terme $\omega_{i,j}$ indique la **force** de la liaison entre \mathcal{E}_i et \mathcal{E}_j . La mise à jour de $\omega_{i,j}$ suit la **règle locale** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

conformément aux principes du **DSL** décrits en **section 2.2.2**. La **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ doit alors être définie pour chaque couple (i, j) selon la nature (modalité) des entités.

Lorsque $\mathcal{E}_i = \mathcal{V}_a$ est une entité visuelle et $\mathcal{E}_j = \mathcal{T}_b$ est une entité textuelle, on peut s’appuyer sur un **embedding** commun (par exemple ϕ_{img} pour les images et ϕ_{txt} pour les textes) afin de mesurer la similarité :

$$S(\mathcal{V}_a, \mathcal{T}_b) = \cos(\phi_{\text{img}}(\mathbf{v}_a), \phi_{\text{txt}}(\mathbf{t}_b)).$$

Cette formule calcule le **cosinus** de l’angle entre deux vecteurs latents, estimant ainsi la correspondance sémantique entre l’image \mathbf{v}_a et le texte \mathbf{t}_b . Il est également possible de recourir à une **fonction de coïncidence** : si l’on dispose, dans un grand corpus, de la fréquence conjointe d’apparition d’un objet visuel \mathcal{V}_a et d’un mot \mathcal{T}_b , on peut définir

$$S(\mathcal{V}_a, \mathcal{T}_b) = \text{freq_cooc}(\mathcal{V}_a, \mathcal{T}_b),$$

ou un score dérivé de cette cooccurrence (par exemple une probabilité normalisée).

Dans le cas d'une **entité sonore** \mathcal{S}_c , l'évaluation de $S(\mathcal{S}_c, \mathcal{T}_b)$ ou $S(\mathcal{S}_c, \mathcal{V}_a)$ peut faire appel à des **corrélations temporelles** ou à des modèles d'alignement entre audio et texte ou audio et image. Par exemple, si un mot « cat » est identifié à l'instant t dans un flux textuel associé à la vidéo ou à la scène, l'extrait audio \mathcal{S}_c couvrant l'intervalle $[t - \delta, t + \delta]$ est davantage susceptible de présenter une synergie élevée avec ce mot, ce qui se reflète dans $\omega_{i,j}$ si \mathcal{S}_c et \mathcal{T}_b coïncident fréquemment sur ce type de segments.

À mesure que l'on incrémente t , la pondération $\omega_{i,j}(t)$ évolue jusqu'à se rapprocher d'un état stable $\omega_{i,j}^*$. Les entités $(\mathcal{E}_i, \mathcal{E}_j)$ qui présentent une corrélation sémantique (ou statistique) élevée et répétée voient leurs liens se renforcer ; les paires moins cohérentes ou rarement associées aboutissent à des pondérations proches de zéro, sous l'effet de la **décroissance** $\tau \omega_{i,j}(t)$ et de la **parsimonie** (voir **section 2.2.3**).

Dans ce **SCN** multimodal, on observe in fine l'émergence de **clusters** où se regroupent plusieurs **objets visuels** \mathcal{V}_a , **concepts textuels** \mathcal{T}_b et éventuellement **segments audio** \mathcal{S}_c , tous reliés par des liens $\omega_{i,j}$ élevés. Ces **clusters** peuvent être interprétés comme des “thèmes” ou “concepts” reliant plusieurs modalités de données, formant ainsi des sous-ensembles d'entités fortement synergiques, analogues à ceux décrits en **section 2.2.5** pour le cas purement unimodal.

C. Avantages : Émergence de Clusters Inter-Modalités

L'un des points forts de cette approche réside dans son **auto-organisation**. Contrairement à un modèle nécessitant des **labels externes** ou un alignement imposé de type “cette image illustre ce mot”, le SCN multimodal laisse la dynamique DSL révéler les **associations** latentes à partir des co-occurrences ou des similarités vectorielles. Ainsi, si un objet visuel \mathcal{V}_a apparaît systématiquement dans un corpus avec un mot \mathcal{T}_b , la règle de mise à jour fera grimper $\omega_{a,b}$. Dès lors que l'on ajoute la composante audio \mathcal{S}_c et que celle-ci se révèle régulièrement liée à la même scène ou au même mot, un “triplet” $(\mathcal{V}_a, \mathcal{T}_b, \mathcal{S}_c)$ peut se renforcer, signant la mise en place d'un **cluster** tri-modal.

En permettant à chaque entité \mathcal{E}_i de se connecter librement à toutes les autres (qu'elles soient images, textes ou sons), le **SCN** favorise la formation de **micro-clusters** reliant quelques images spécifiques, un groupe de termes pertinents et certains signaux audio spécifiques. À plus grande échelle (voir **chapitre 6** si la référence renvoie à un traitement plus approfondi), on voit aussi des **macro-clusters** se constituer en agrégeant plusieurs de ces micro-clusters autour d'un thème élargi.

Les **conséquences pratiques** sont multiples. Une fois cette structure auto-organisée établie, on peut réaliser :

- de la **recherche** ou de la **recommandation** cross-modale, en identifiant les images “proches” d'un mot ou les mots les plus fortement reliés à un extrait audio ;
- de l'**annotation** automatique en sélectionnant les termes textuels qui apparaissent dans le cluster d'une image donnée ;
- un **système cognitif** multimodal pouvant rapidement basculer d'une modalité à l'autre (texte, image, audio) pour décrire, identifier ou comparer un contenu.

L'émergence de clusters enrichis sur le plan sémantique peut ainsi servir de base à des applications d'**indexation**, de **filtrage** ou de **fusion** de données, répondant au besoin de manipuler des informations hétérogènes sans imposer a priori un alignement rigide ou des labels manuels sur chaque entité. Cette adaptabilité vient directement de la **dynamique** du DSL, qui combine les notions de synergie, de compétition (inhibition) et de recuit potentiel (chap. 7.3), maintenant dans un même cadre la plasticité nécessaire à la mise en correspondance de modalités distinctes.

D. Conclusion

La **dynamique** du DSL, fondée sur la mise à jour itérative des liaisons $\omega_{i,j}$ sous l'influence de la **synergie** $S(i,j)$, s'adapte naturellement à un **univers** multimodal (visuel, textuel, sonore). Les entités qui **co-opèrent** (c'est-à-dire celles dont les indices d'appariement sont forts) voient leurs liaisons se **renforcer** au fil des itérations, tandis que les liens peu pertinents disparaissent ou s'affaiblissent. Il en résulte un **SCN** où des **clusters** ou **groupes** d'entités multimodales s'assemblent, reflétant des concepts ou des événements communs (ex. le concept “chat” unissant une image d'un chat, le mot “cat” et un extrait audio de miaulement).

Cet aspect **auto-organisé** facilite la **découverte** et la **structuration** de corrélations complexes entre des flux hétérogènes. Il ne repose pas uniquement sur des étiquettes (labels) fixes, mais sur la **dynamique** même des pondérations ω . La suite (section 8.1.2.3) prolongera cette discussion en examinant comment ces **clusters** multimodaux peuvent être exploités pour diverses applications (indexation, annotation, réponse conversationnelle, etc.), et comment la mise à jour peut être affinée par des **mécanismes** d'inhibition ou de compétition afin de maîtriser la croissance du réseau.

8.1.2.3. Montrer des applications (ex. annotation d'images, reconnaissance audio-visuelle, systèmes multimédias)

Une fois que l'on a posé les bases théoriques (sections 8.1.2.1 et 8.1.2.2) et qu'on comprend comment des entités de diverses modalités (visuel, audio, texte) peuvent interagir dans un **Synergistic Connection Network (SCN)** commun, il devient naturel d'explorer des **applications** mettant à profit cette **dynamique**. Le **Deep Synergy Learning (DSL)** se prête en effet à plusieurs cas d'usage, allant de la **classification** et **annotation** d'images jusqu'aux **systèmes multimédias** plus complexes, en passant par la **reconnaissance audio-visuelle**. Nous illustrerons ci-après trois domaines dans lesquels la synergie **inter-modale** et la **mise à jour** automatique des liaisons $\omega_{i,j}$ font la différence.

A. Annotation d'Images

L'annotation d'images vise à associer une **image** ou une **région d'image** à un ou plusieurs **mots-clés** décrivant son contenu. Contrairement aux approches classiques basées sur des classificateurs ou des modèles de détection d'objets spécifiques (reconnaissance de chat, voiture, bâtiment, etc.), la dynamique **DSL** permet une structuration plus **générale et auto-organisée** des relations entre les entités visuelles et textuelles.

Dans ce cadre, les **entités visuelles** $\{\mathcal{E}_i^{(\text{img})}\}$ correspondent aux images globales ou à des "patches"/"bounding boxes" spécifiques, tandis que les **entités textuelles** $\{\mathcal{E}_j^{(\text{txt})}\}$ représentent des mots, phrases ou concepts associés (ex. "chat", "ciel bleu", "bâtiment"). L'ensemble des interactions est encapsulé dans un **SCN (Synergistic Connection Network)**, où la **synergie**

$$S(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_j^{(\text{txt})})$$

exprime la **compatibilité** entre une représentation visuelle (issue d'un CNN) et une représentation sémantique (obtenue par Word2Vec, GloVe, BERT, etc.). Une formulation typique est la similarité cosinus entre leurs **embeddings** respectifs :

$$S(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_j^{(\text{txt})}) = \cos(\phi_{\text{vis}}(\mathbf{x}_i), \phi_{\text{txt}}(\mathbf{x}_j)),$$

où ϕ_{vis} et ϕ_{txt} sont des projections dans un espace latent partagé.

L'évolution des **liaisons** $\omega_{i,j}$ est gouvernée par la mise à jour **DSL** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Au fil des itérations, les **associations pertinentes** se renforcent, favorisant l'émergence de clusters reliant des images à des mots-clés. Une image contenant un chat développera progressivement une liaison forte avec le mot "chat", tandis qu'une autre, représentant un chien, établira une connexion robuste avec "chien".

Plutôt que de s'appuyer uniquement sur des annotations manuelles, le SCN **auto-organise** ces relations, permettant d'identifier des **groupements plus larges** :

- Une collection d'images de félin pourrait être associée aux mots "cat", "lion", "tiger", créant un cluster cohérent.
- Un **macro-cluster** plus large (voir Chapitre 6) peut émerger, regroupant toutes les images liées à la faune et leurs descriptions textuelles associées.

Cette approche favorise une **adaptabilité** accrue, facilitant l'**insertion dynamique** de nouvelles images ou mots sans nécessiter d'apprentissage supervisé supplémentaire. De nouvelles entités s'intègrent naturellement, ajustant leurs connexions ω avec les structures existantes, en cohérence avec la logique d'**apprentissage continu** (voir Chapitre 7.6).

L'annotation d'images basée sur le SCN présente plusieurs **bénéfices** majeurs :

- **Robustesse et Correction d'Erreurs** : L'**auto-organisation** du réseau compense les imperfections d'un modèle de reconnaissance. Si un mot-clé est **pertinent** pour un sous-ensemble d'images, cette influence structurelle **attire** progressivement les connexions ω , même si l'algorithme de détection initial a omis certaines instances.
- **Recherche Cross-Modale** : L'interrogation du SCN devient triviale. On peut demander : “*Quelles sont les images les plus liées au mot ‘chat’ ?*” en se basant directement sur les poids $\omega_{i,j}$, sans avoir à entraîner un modèle spécifique pour chaque tâche.
- **Scalabilité et Adaptabilité** : Le réseau croît de manière **incrémentale** au fur et à mesure que de nouvelles entités sont ajoutées. Grâce aux mécanismes de **parsimonie** (filtrage des liens faibles, régularisation par seuil ω_{\min}), la taille du SCN reste maîtrisée, garantissant une **structure efficace** et optimisée pour la recherche et l'interprétation des relations image-texte.

B. Reconnaissance Audio-Visuelle

La **reconnaissance** audio-visuelle constitue un cas particulier de **Deep Synergy Learning (DSL)** appliqué à des flux multimodaux où coexistent des données vidéo et des données audio. L'objectif, par exemple, est de détecter et de caractériser des **événements** dans une séquence vidéo, en faisant correspondre chaque **entité** visuelle (souvent un ensemble de frames ou de features extraites) à des **entités** audio (segments sonores ou embeddings).

Pour formaliser cette approche, on suppose l'existence d'un SCN (Synergistic Connection Network) comprenant deux ensembles d'entités : $\{\mathcal{E}_i^{(\text{vis})}\}$ pour la partie **visuelle** et $\{\mathcal{E}_j^{(\text{aud})}\}$ pour la partie **audio**. Chaque lien $\omega_{i,j}$ symbolise la force de la liaison entre $\mathcal{E}_i^{(\text{vis})}$ et $\mathcal{E}_j^{(\text{aud})}$. Conformément à la **section 2.2.2**, la mise à jour de $\omega_{i,j}$ se décrit par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i^{(\text{vis})}, \mathcal{E}_j^{(\text{aud})}) - \tau \omega_{i,j}(t)],$$

où η est le taux d'apprentissage, τ un coefficient de décroissance, et $S(\cdot, \cdot)$ la **synergie** reliant l'entité visuelle $\mathcal{E}_i^{(\text{vis})}$ à l'entité audio $\mathcal{E}_j^{(\text{aud})}$.

La **fonction** S peut reposer sur une **coïncidence temporelle**, par exemple lorsqu'on souhaite aligner un segment audio $[t - \delta, t + \delta]$ à la frame vidéo correspondant à l'instant t . On peut également définir un **embedding** cross-modal, ϕ_{vis} et ϕ_{aud} , chacun projetant les données visuelles et sonores dans un espace latent partagé. Dans ce dernier cas, la synergie peut s'écrire sous forme de cosinus :

$$S(\mathcal{E}_i^{(\text{vis})}, \mathcal{E}_j^{(\text{aud})}) = \cos(\phi_{\text{vis}}(\mathbf{v}_i), \phi_{\text{aud}}(\mathbf{a}_j)).$$

Ces deux types de définitions (temporelle ou sémantique) ne sont d'ailleurs pas exclusifs et peuvent être combinés pour une robustesse accrue.

Au fil des itérations, la **descente** des pondérations $\omega_{i,j}$ tend à renforcer les liens qui révèlent une correspondance répétée ou stable, et à faire décroître ceux qui n'apportent pas de synergie suffisante. En s'appuyant sur la **parsimonie** décrite en **section 2.2.3**, on élimine les liaisons trop faibles qui ne contribuent guère à la structure globale, ce qui met en évidence des **clusters** audio-visuels d'autant plus nets.

Ces **clusters** représentent des événements ou des scènes dans la séquence vidéo, où un ensemble de frames $\mathcal{E}_i^{(\text{vis})}$ s'avère relié à un ensemble de segments audio $\mathcal{E}_j^{(\text{aud})}$. Par exemple, lors d'un match de football, l'instant du but associe une vue du stade et des joueurs célébrant (frames vidéo), ainsi qu'une montée de la clameur de la foule (segment audio)

ou le commentaire spécifique du présentateur. De même, un “cluster concert” peut regrouper des images de musiciens sur scène et des extraits sonores contenant guitare ou batterie.

Les **applications** pratiques sont diverses. En **indexation** de vidéos, on cherche souvent à localiser automatiquement le début et la fin d'un événement particulier. Les liens $\omega_{i,j}$ élevés dans le **SCN** signalent la cooccurrence forte de frames et de segments audio, ce qui permet de pointer rapidement où se situe l'action. En **robotique**, un robot muni de capteurs auditifs et visuels peut, grâce à la dynamique DSL, associer un son spécifique à une scène visuelle, localisant par exemple un objet qui émet un signal sonore. Par ailleurs, sur le plan de l'**accessibilité**, détecter la synchronisation audio-visuelle (par exemple, la parole correspond à telle personne filmée) ouvre la porte à la **création** de légendes ou de sous-titres automatiques en temps réel.

Pour éviter le risque de stagner dans des **minima locaux**, on peut employer un recuit simulé (voir chap. 7.3) qui injecte du bruit dans la mise à jour des pondérations, augmentant la probabilité de franchir des barrières d'énergie. L'**inhibition adaptative** (chap. 7.4) offre un mécanisme complémentaire en limitant la prolifération de liens au sein d'un nœud donné, ce qui favorise une représentation plus épars et des clusters plus clairement séparés. L'aboutissement est une **organisation** audio-visuelle où chaque événement, scène ou action est identifié par un regroupement de frames et de segments sonores, rendant la reconnaissance des temps forts (but, concert, interview, etc.) plus aisée et plus explicite au sein du **réseau**.

Programme Python 1

Voici ci-dessous un programme Python complet qui simule, de manière concrète, le concept de **reconnaissance audio-visuelle** dans un **Deep Synergy Learning (DSL)**. Le programme s'appuie sur une simulation de deux flux de données – un flux d'**entités visuelles** et un flux d'**entités audio** – et sur la mise à jour des pondérations dans un **Synergistic Connection Network (SCN)** à l'aide d'une fonction de **synergie** $S(i,j)$. Le programme est écrit dans un style académique et structuré, intégrant des formules mathématiques et des explications détaillées.

```

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Fixer une graine aléatoire pour assurer la reproductibilité
np.random.seed(42)

# Paramètres généraux
eta = 0.1      # Taux d'apprentissage local
tau = 0.2       # Coefficient de décroissance
num_iterations = 50 # Nombre d'itérations de la dynamique

# Simulation de deux groupes d'entités correspondant à deux événements distincts.
# Le groupe 1 (par exemple, un concert) aura des entités visuelles et audio centrées autour de (1, 1)
# Le groupe 2 (par exemple, un match de football) aura des entités centrées autour de (5, 5).

# Générer des embeddings visuels pour le groupe 1 et groupe 2
num_video_group1 = 5
num_video_group2 = 5
video_embeddings = np.vstack([
    np.random.randn(num_video_group1, 2) * 0.2 + np.array([1, 1]),
    np.random.randn(num_video_group2, 2) * 0.2 + np.array([5, 5])
])

# Générer des embeddings audio pour le groupe 1 et groupe 2
num_audio_group1 = 5
num_audio_group2 = 5
audio_embeddings = np.vstack([

```

```

        np.random.randn(num_audio_group1, 2) * 0.2 + np.array([1, 1]),
        np.random.randn(num_audio_group2, 2) * 0.2 + np.array([5, 5])
    ])

# Déterminer le nombre total d'entités pour chaque modalité
num_video = video_embeddings.shape[0]
num_audio = audio_embeddings.shape[0]

# Initialiser la matrice des pondérations, notée omega, de dimension (num_video, num_audio)
omega = np.full((num_video, num_audio), 0.05)

# Définir la fonction de synergie S(i, j)
# Ici, nous utilisons la similarité cosinus comme mesure de synergie.
def cosine_similarity(vec1, vec2):
    norm1 = np.linalg.norm(vec1)
    norm2 = np.linalg.norm(vec2)
    if norm1 == 0 or norm2 == 0:
        return 0.0
    return np.dot(vec1, vec2) / (norm1 * norm2)

def synergy(i, j):
    # La synergie entre l'entité visuelle i et l'entité audio j est évaluée via la similarité cosinus.
    # On s'assure de renvoyer une valeur positive (bornée dans [0,1]).
    sim = cosine_similarity(video_embeddings[i], audio_embeddings[j])
    return max(sim, 0)

# Simulation de la dynamique des pondérations
# La mise à jour suit la règle :
#  $\omega_{(i,j)}(t+1) = \omega_{(i,j)}(t) + \eta [ S(i,j) - \tau \omega_{(i,j)}(t) ]$ 
for t in range(num_iterations):
    for i in range(num_video):
        for j in range(num_audio):
            S_ij = synergy(i, j)
            omega[i, j] = omega[i, j] + eta * (S_ij - tau * omega[i, j])
    # Optionnel : affichage périodique de la matrice de pondérations
    if (t + 1) % 10 == 0:
        print(f"Iteration {t + 1} : omega =\n{np.round(omega, 3)}\n")

# Visualiser la matrice finale des pondérations sous forme de heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(omega, annot=True, cmap="viridis")
plt.title("Matrice Finale des Pondérations  $\omega$  (Entités Visuelles  $\times$  Entités Audio)")
plt.xlabel("Entités Audio")
plt.ylabel("Entités Visuelles")
plt.show()

# Visualiser les embeddings dans l'espace 2D pour illustrer les clusters
plt.figure(figsize=(8, 6))
plt.scatter(video_embeddings[:, 0], video_embeddings[:, 1], color="blue", label="Embeddings Visuels")
plt.scatter(audio_embeddings[:, 0], audio_embeddings[:, 1], color="red", marker="x", label="Embeddings Audio")
plt.title("Représentation des Embeddings Visuels et Audio dans  $\mathbb{R}^2$ ")
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")
plt.legend()
plt.show()

```

Dans ce **programme**, nous avons simulé deux groupes d'entités correspondant à deux types d'événements distincts, à l'instar d'un **cluster "concert"** et d'un **cluster "sport football"**. Les **embeddings visuels** (représentés par des vecteurs dans \mathbb{R}^2) et les **embeddings audio** sont générés à partir de distributions gaussiennes centrées respectivement autour de points caractéristiques (par exemple, (1,1) pour le premier groupe et (5,5) pour le second groupe).

La **fonction de synergie** $S(i,j)$ utilisée ici se fonde sur la **similarité cosinus**, ce qui permet de mesurer la proximité angulaire entre un vecteur visuel et un vecteur audio. Le principe mathématique est le suivant : pour deux vecteurs \mathbf{x}_i et \mathbf{x}_j dans \mathbb{R}^d , la similarité cosinus se définit par

$$\text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|},$$

et est bornée entre -1 et 1. Dans notre simulation, nous nous assurons que $S(i,j) \geq 0$ en utilisant la fonction $\max\{0, \cdot\}$.

La **mise à jour** des pondérations $\omega_{i,j}$ est réalisée de manière itérative suivant la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où η et τ sont respectivement le **taux d'apprentissage** et le **coefficent de décroissance**. Cette formule intègre deux composantes essentielles : le **renforcement** de la connexion par un terme proportionnel à la synergie, et la **décroissance** qui empêche la croissance infinie des pondérations. En théorie, si la mise à jour était poursuivie à l'infini sans interruption, chaque $\omega_{i,j}$ convergerait vers un **équilibre** approximatif $\omega_{i,j}^* \approx S(i,j)/\tau$.

Les **simulations** réalisées sur un petit ensemble d'entités permettent d'observer que les pondérations se renforcent pour les paires dont les vecteurs sont proches (ou bien alignés) et restent faibles pour celles dont la synergie est faible. Ce phénomène conduit à l'**émergence** de **clusters** – par exemple, des entités visuelles et audio associées à un concert se regroupent et se distinguent d'un groupe lié à un match de football.

Enfin, nous avons inclus une visualisation de la **matrice** des pondérations ainsi que des **embeddings** dans l'espace \mathbb{R}^2 . Ces graphiques aident à interpréter le processus de **clustering auto-organisé** et confirmant l'efficacité de la **mise à jour locale** dans le cadre du DSL.

Ce programme constitue ainsi une **illustration** concrète du mécanisme de **Deep Synergy Learning** appliquée à un problème de **reconnaissance audio-visuelle**. Les applications potentielles, telles que l'indexation de vidéos, la robotique ou l'accessibilité, peuvent bénéficier de cette approche en permettant de détecter automatiquement des événements en regroupant les flux d'information visuels et audio au sein d'un même **SCN**.

Programme Python 2

Voici une version modifiée du programme qui résout le problème de dimensions incompatibles entre les embeddings visuels (de dimension 1280) et les embeddings audio (de dimension 13). Pour permettre le calcul de la **similarité cosinus** entre ces deux types d'entités, nous utilisons une projection linéaire aléatoire fixe qui mappe les features audio dans un espace de dimension 1280. Cette transformation assure que les deux vecteurs à comparer appartiennent au même espace vectoriel, et permet ainsi de calculer la similarité.

Le code ci-dessous est entièrement commenté et rédigé dans un style académique :

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import subprocess
```

```

import io
import soundfile as sf
import librosa
import librosa.display
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input
from tensorflow.keras.preprocessing import image

# Pour assurer la reproductibilité
np.random.seed(42)

#####
# Section 1 : Extraction des Features Vidéo et Audio
#####

# 1.1. Extraction des frames vidéo à l'aide d'OpenCV
video_path = "video.mp4"
cap = cv2.VideoCapture(video_path)
if not cap.isOpened():
    raise IOError("Erreur lors de l'ouverture du fichier vidéo. Vérifiez que 'video.mp4' existe et est accessible.")

total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
fps = cap.get(cv2.CAP_PROP_FPS)
duration_video = total_frames / fps

# On choisit d'extraire nb_frames frames de façon équidistante
nb_frames = 10
frame_indices = np.linspace(0, total_frames - 1, nb_frames, dtype=int)

video_frames = []
current_frame = 0
extracted = 0
while cap.isOpened() and extracted < nb_frames:
    ret, frame = cap.read()
    if not ret:
        break
    if current_frame in frame_indices:
        frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        video_frames.append(frame_rgb)
        extracted += 1
    current_frame += 1
cap.release()

# 1.2. Extraction des embeddings visuels avec MobileNetV2
# Charger MobileNetV2 pré-entraîné (sans couche de classification et avec pooling global)
model = MobileNetV2(weights="imagenet", include_top=False, pooling='avg')
visual_features = []
for frame in video_frames:
    frame_resized = cv2.resize(frame, (224, 224))
    x = image.img_to_array(frame_resized)
    x = np.expand_dims(x, axis=0)
    x = preprocess_input(x)
    features = model.predict(x)
    visual_features.append(features.flatten())
visual_features = np.array(visual_features) # Dimensions : (nb_frames, 1280)

```

```

# 1.3. Extraction de l'audio depuis video.mp4 via ffmpeg et subprocess
# La commande ffmpeg suivante extrait l'audio en format WAV (mono, 22050 Hz)
command = [
    'ffmpeg',
    '-i', video_path,
    '-f', 'wav',
    '-acodec', 'pcm_s16le',
    '-ac', '1',
    '-ar', '22050',
    'pipe:1'
]
try:
    result = subprocess.run(command, stdout=subprocess.PIPE, stderr=subprocess.PIPE, check=True)
except FileNotFoundError:
    raise FileNotFoundError("ffmpeg n'est pas installé ou n'est pas dans le PATH. Veuillez l'installer pour exécuter ce programme.")
except subprocess.CalledProcessError as e:
    print("Erreur lors de l'extraction audio :", e.stderr.decode())
    raise e

audio_buffer = io.BytesIO(result.stdout)
audio_signal, sr = sf.read(audio_buffer)
if audio_signal.ndim > 1:
    audio_signal = np.mean(audio_signal, axis=1)
duration_audio = len(audio_signal) / sr

# Diviser l'audio en nb_frames segments (un segment par frame)
segment_duration = duration_audio / nb_frames
audio_features = []
for i in range(nb_frames):
    start_sample = int(i * segment_duration * sr)
    end_sample = int((i + 1) * segment_duration * sr)
    segment = audio_signal[start_sample:end_sample]
    mfcc = librosa.feature.mfcc(y=segment, sr=sr, n_mfcc=13)
    mfcc_mean = np.mean(mfcc, axis=1)
    audio_features.append(mfcc_mean)
audio_features = np.array(audio_features) # Dimensions : (nb_frames, 13)

#####
# Section 2 : Projection des Features Audio
#####

# Pour calculer la similarité cosinus entre des vecteurs de dimensions différentes,
# nous projetons les features audio de dimension 13 dans un espace de dimension 1280.
# Nous utilisons une projection linéaire aléatoire fixe.
proj_matrix = np.random.randn(13, 1280) # Matrice de projection de dimension (13, 1280)
audio_features_projected = np.dot(audio_features, proj_matrix) # Dimensions : (nb_frames, 1280)

#####
# Section 3 : Définition de la Fonction de Synergie
#####


def cosine_similarity(vec1, vec2):
    norm1 = np.linalg.norm(vec1)
    norm2 = np.linalg.norm(vec2)
    if norm1 == 0 or norm2 == 0:

```

```

    return 0.0
return np.dot(vec1, vec2) / (norm1 * norm2)

def synergy(i, j):
    # Calculer la similarité cosinus entre l'embedding visuel (dimension 1280) et
    # l'embedding audio projeté (dimension 1280) pour obtenir un score dans [0,1].
    sim = cosine_similarity(visual_features[i], audio_features_projected[j])
    return max(sim, 0)

# Construire la matrice de synergie S_matrix de dimension (nb_frames, nb_frames)
S_matrix = np.zeros((nb_frames, nb_frames))
for i in range(nb_frames):
    for j in range(nb_frames):
        S_matrix[i, j] = synergy(i, j)

#####
# Section 4 : Simulation de la Dynamique du SCN
#####

# Initialiser la matrice des pondérations ω avec une valeur initiale faible
omega = np.full((nb_frames, nb_frames), 0.05)

eta = 0.1 # Taux d'apprentissage
tau = 0.2 # Coefficient de décroissance
num_iterations = 50

# Appliquer la règle de mise à jour du DSL :
#  $\omega(i,j)(t+1) = \omega(i,j)(t) + \eta [ S(i,j) - \tau \omega(i,j)(t) ]$ 
for t in range(num_iterations):
    for i in range(nb_frames):
        for j in range(nb_frames):
            omega[i, j] += eta * (S_matrix[i, j] - tau * omega[i, j])
    if (t + 1) % 10 == 0:
        print(f"Iteration {t+1}:\n{np.round(omega, 3)}\n")

#####
# Section 5 : Visualisation des Résultats
#####

# Visualisation de la matrice finale des pondérations sous forme de heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(omega, annot=True, cmap="viridis")
plt.title("Matrice Finale des Pondérations ω (Visuel × Audio)")
plt.xlabel("Indices Audio")
plt.ylabel("Indices Visuels")
plt.show()

# Projection 2D des embeddings pour une intuition sur leur distribution
plt.figure(figsize=(8, 6))
plt.scatter(visual_features[:, 0], visual_features[:, 1], color="blue", label="Features Visuelles")
plt.scatter(audio_features_projected[:, 0], audio_features_projected[:, 1], color="red", marker="x", label="Features Audio (Proj.)")
plt.title("Projection 2D des Features Visuelles et Audio Projétées")
plt.xlabel("Dimension 1")
plt.ylabel("Dimension 2")

```

```
plt.legend()  
plt.show()
```

Dans ce programme, nous traitons un fichier vidéo unique (*video.mp4*) qui contient à la fois le flux visuel et le flux audio. Les étapes majeures sont les suivantes :

559.Extraction des Frames Vidéo

Les frames sont extraites de manière équidistante avec **OpenCV**. Chaque frame est convertie en format RGB et redimensionnée à 224×224 pixels pour être compatible avec le modèle **MobileNetV2**.

560.Extraction des Embeddings Visuels

Le modèle MobileNetV2 pré-entraîné (avec pooling global) fournit un vecteur d'embedding de dimension 1280 pour chaque frame extraite.

561.Extraction de l'Audio

La commande ffmpeg est exécutée via le module **subprocess** pour extraire l'audio du fichier vidéo. Le signal audio est ensuite lu avec **soundfile** et segmenté en autant de parties que le nombre de frames. Pour chaque segment, les **MFCCs** sont calculés à l'aide de **librosa** et moyennés pour obtenir un vecteur de dimension 13.

562.Projection des Features Audio

Afin d'aligner les dimensions des embeddings visuels (1280) et des embeddings audio (13), nous utilisons une projection linéaire aléatoire fixe qui transforme les vecteurs audios dans un espace de dimension 1280. Cette étape est cruciale pour permettre le calcul de la similarité cosinus entre des vecteurs de même dimension.

563.Définition et Calcul de la Synergie

La synergie $S(i,j)$ est calculée comme la similarité cosinus entre un vecteur d'embedding visuel et le vecteur audio projeté correspondant, assurant ainsi une valeur dans l'intervalle [0,1].

564.Mise à Jour des Pondérations du SCN

La règle de mise à jour utilisée est :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

ce qui correspond à un renforcement proportionnel à la synergie, combiné à une décroissance pour éviter une croissance illimitée. Après plusieurs itérations, la dynamique converge vers des valeurs reflétant la force de la synergie entre les entités.

565.Visualisation

La matrice finale des pondérations est affichée sous forme de heatmap, et une projection 2D des embeddings visuels ainsi que des embeddings audio projetés permet d'obtenir une intuition sur la structuration des clusters dans l'espace des features.

Ce programme fournit ainsi une **illustration concrète** de la reconnaissance audio-visuelle appliquée au Deep Synergy Learning, en intégrant directement l'audio depuis le fichier vidéo sans MoviePy, et en harmonisant les dimensions des données via une projection linéaire.

C. Systèmes Multimédias et Fusion Avancée

Lorsqu'un **SCN** (Synergistic Connection Network) s'étend à plusieurs sources multimédias – texte, audio, images ou encore métadonnées – le **Deep Synergy Learning (DSL)** offre un cadre général pour faire émerger des **macro-clusters** ou des **communautés** qui reflètent des thèmes globaux, des préférences utilisateur ou des types de contenus récurrents. L'idée directrice reprend la logique exposée en **section 2.2.1** (définition d'entités variées) et en **section 2.2.2** (mise à jour locale des liaisons), tout en l'adaptant à des couples d'entités relevant de modalités différentes.

Pour un **système** multimédia complexe, on définit plusieurs catégories d'entités :

- Des entités **texte** $\{\mathcal{T}_k\}$, regroupant par exemple des mots, des tags, des descriptions ou des résumés de contenu.
- Des entités **audio** $\{\mathcal{A}_j\}$, décrites via des embeddings sonores (spectrogrammes, MFCC ou modèles pré-entraînés) et potentiellement associées à des pistes musicales ou des bruitages.
- Des entités **visuelles** $\{\mathcal{V}_i\}$, allant d'images fixes à des extraits ou des embeddings vidéo résumant les frames d'un clip.
- Des entités **métadonnées** $\{\mathcal{M}_r\}$, comme la géolocalisation, les dates, les catégories, ou encore les profils d'utilisateur $\{\mathcal{U}_\ell\}$.

L'ensemble global de ces entités forme le réseau \mathcal{E} , où chaque nœud \mathcal{E}_u provient d'une de ces modalités. On introduit alors une matrice de pondérations ω , de taille $|\mathcal{E}| \times |\mathcal{E}|$. Comme indiqué en **section 2.2.2**, la mise à jour de chaque liaison $\omega_{u,v}$ obéit à une équation de la forme

$$\omega_{u,v}(t+1) = \omega_{u,v}(t) + \eta [S(\mathcal{E}_u, \mathcal{E}_v) - \tau \omega_{u,v}(t)].$$

La **fonction de synergie** $S(\mathcal{E}_u, \mathcal{E}_v)$ s'adapte au type de couple (u, v) . Entre un nœud **texte** et un nœud **image**, on pourra s'appuyer sur une similarité cosinus dans un espace latent commun. Entre un nœud **utilisateur** et un nœud **vidéo**, on peut modéliser la probabilité que cet utilisateur aime la vidéo en exploitant les liens qu'il partage déjà avec des contenus semblables. Entre un **profil** utilisateur et un **tag** textuel, on peut s'appuyer sur les préférences extraites par co-occurrence ou par historique d'utilisation.

De cette façon, la **dynamique** DSL autorise la fusion de multiples modalités dans un même **SCN**. Lorsqu'on applique cette méthode à une **plateforme** multimédia – par exemple, un site de diffusion de vidéos musicales – on obtient un graphe où :

- Les entités “vidéo \mathcal{V}_i ” sont décrites par leurs embeddings visuels ou par des analyses de frames.
- Les entités “audio \mathcal{A}_j ” reflètent les pistes sonores ou les morceaux musicaux sous forme de vecteurs caractéristiques.
- Les entités “texte \mathcal{T}_k ” regroupent titres, tags, descriptions ou catégories associées.
- Les entités “utilisateurs \mathcal{U}_ℓ ” décrivent les profils (préférences, historique de navigation ou d'écoute).

La **synergie** $S(\mathcal{U}_\ell, \mathcal{V}_i)$ peut traduire la probabilité qu'un utilisateur \mathcal{U}_ℓ apprécie la vidéo \mathcal{V}_i . Cette probabilité peut reposer sur la similarité de \mathcal{V}_i avec d'autres vidéos que \mathcal{U}_ℓ a aimées, sur des points communs textuels (\mathcal{T}_k) ou sur des corrélations audio (\mathcal{A}_j). Au fur et à mesure que le réseau s'**auto-organise**, les liens $\omega_{\ell,i}$ se renforcent si l'utilisateur \mathcal{U}_ℓ et la vidéo \mathcal{V}_i présentent une affinité, et décroissent dans le cas contraire, aboutissant à la formation de **communautés** de goût ou de **clusters** de contenus.

À mesure qu'augmente la taille du réseau (nouvelles vidéos, nouvelles pistes audios, nouveaux tags ou nouveaux utilisateurs), la **dynamique** DSL s'applique de manière incrémentale. Le **SCN** calcule la synergie entre la nouvelle entité et les entités existantes ; il met à jour les pondérations ω et, progressivement, la nouvelle entité trouve sa place dans le graphe, éventuellement rattachée à des clusters déjà constitués. Pour contrôler l'explosion du nombre de liaisons, on met en œuvre des **règles de parsimonie** (section 2.2.3) ou d'**inhibition** (chap. 7.4) : on filtre les poids $\omega_{u,v}$ trop faibles et on limite la croissance simultanée de trop nombreuses connexions, ce qui maintient le graphe dans une taille raisonnable.

L'**avantage** central de cette approche repose sur l'**unification** auto-organisée : les **macro-clusters** qui se constituent peuvent intégrer simultanément des éléments sonores, des vidéos, des étiquettes textuelles et des profils utilisateur. Ces groupes à la fois multimodaux et multi-entités décrivent ainsi des thèmes ou des centres d'intérêt partagés, sans recours à un modèle supervisé distinct pour chaque modalité. Cette **fusion** avancée favorise la recommandation de contenus (un utilisateur \mathcal{U}_t peut découvrir de nouvelles vidéos \mathcal{V}_i associées à sa communauté), facilite la recherche (croisement texte-image-audio), et assure la **scalabilité** par la nature incrémentale et distribuée du DSL. En définitive, un **SCN** multimédia se dresse comme un **réseau** de connexions adaptatives où chaque entité, quelle que soit sa modalité, trouve ses partenaires les plus pertinents, laissant émerger des **macro-clusters** globaux riches de sens pour l'exploration, l'indexation et la recommandation à grande échelle.

Conclusion (8.1.2.3)

Les **applications** du **DSL** à un contexte multimodal couvrent un **large** éventail de problématiques :

- 566. **Annotation d'images** : Relier automatiquement des contenus visuels à des mots ou concepts textuels, en s'appuyant sur la dynamique DSL pour découvrir et renforcer les associations correspondantes.
- 567. **Reconnaissance audio-visuelle** : Faire émerger des **clusters** d'événements ou de scènes en croisant frames vidéo et segments audio, afin d'identifier (ou indexer) des moments clés (sport, musique, conversation, etc.).
- 568. **Systèmes multimédias** : Englobant plusieurs sources (texte, audio, visuel, métadonnées, profils utilisateurs), le SCN résulte en une structure riche permettant la **recommandation**, la **classification**, la **découverte** de communautés ou de motifs sémantiques.

Sur le plan **mathématique**, on retrouve toujours la même **logique** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où la **fonction de synergie** S doit être **adaptée** à chaque paire (image–texte, audio–image, texte–audio, etc.). La possibilité de composer ainsi diverses modalités dans un unique **SCN** fait la **force** du **DSL** : la **dynamique** auto-organisée, non supervisée, permet de **révéler** des structures d'association en s'appuyant sur des indices multiples, offrant ainsi des **résultats** plus riches et plus robustes que les traitements unimodaux isolés.

8.1.3. Plan du Chapitre

Afin de déployer clairement la logique du **DSL multimodal**, ce chapitre 8 s'organise en plusieurs sections, chacune dédiée à un **aspect** précis de la fusion entre différentes modalités (vision, langage, audio, etc.). L'objectif est d'étudier **comment** le **DSL** (Deep Synergy Learning) orchestre l'**intégration** de ces flux, tant sur les plans conceptuels (principes de fusion), qu'implémentaires (architectures SCN dédiées) ou mathématiques (définition de la synergie inter-modalités).

8.1.3.1. Présentation de la Structure : (8.2) Principes de la Fusion Multimodale en DSL, (8.3) Vision, (8.4) Langage, (8.5) Audio, etc.

Il est utile, à ce stade, de **présenter** de manière structurée la suite de l'**exposition** autour de la **fusion multimodale** dans le contexte du **Deep Synergy Learning** (DSL). L'objectif principal est d'organiser la réflexion et les développements dans un **Synergistic Connection Network** (SCN) lorsque plusieurs **modalités** (vision, langage, audio, etc.) sont considérées simultanément. Dans ce qui suit, la **section 8.2** se concentre sur les **principes** de la fusion multimodale. Les **sections 8.3, 8.4, 8.5** examinent ensuite, chacune, un canal majeur : la **vision**, le **langage** et l'**audio**. Enfin, une ouverture sur d'autres flux (capteurs divers, signaux EEG, métadonnées) est évoquée dans la continuité.

Section 8.2 (Principes de la Fusion Multimodale) établit tout d'abord le **fondement** théorique. Il y est discuté **pourquoi** fusionner plusieurs canaux (tels que des images, du texte ou des enregistrements sonores) à l'intérieur d'un même SCN ; la **dynamique d'auto-organisation** du DSL y est rappelée sous forme de l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où $\omega_{i,j}$ représente la **pondération** entre deux entités \mathcal{E}_i et \mathcal{E}_j . L'enjeu majeur consiste à définir la **fonction de synergie** S entre des entités issues de modalités différentes, par exemple une **image** et un **segment** textuel, ou bien un **extrait** audio et un **objet** visuel. Les questions de **normalisation** des échelles, de répartition dans la matrice ω selon le type (intra-canal ou inter-canal) et de **gestion** de la **complexité** sont aussi abordées dans cette section 8.2, en lien avec les **optimisations** déjà évoquées au **chapitre 7**.

Dans la **section 8.3 (Vision)**, l'accent est mis sur l'exploitation de flux **visuels** (images ou vidéos) au sein du DSL. On considère comment un **embedding** d'image, noté $\mathbf{x}_i^{(\text{visuel})} \in \mathbb{R}^d$, peut nourrir la **similarité** $\text{sim}(\mathbf{x}_i^{(\text{visuel})}, \mathbf{x}_j^{(\text{visuel})})$ et ainsi former la **synergie** $S(i,j)$ lorsqu'il s'agit de lier deux entités purement visuelles. L'enjeu est de comprendre comment, par la dynamique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

ces entités s'**agglomèrent** en **clusters** couvrant différents niveaux : micro-patches ou régions locales, d'une part, et macro-objets ou catégories plus abstraites, d'autre part (cf. section 8.3). Les ponts avec la **multi-échelle** (au sens du **chapitre 6**) sont mis en avant, puisque le DSL peut faire émerger des **super-nœuds** (ex. un ensemble d'images représentant la même classe de scènes).

La **section 8.4 (Langage)** se penche ensuite sur la façon dont le **DSL** s'applique à des entités **linguistiques** (mots, segments phraséologiques, etc.). Elle décrit la **construction** d'embeddings textuels, comme ceux issus de **word2vec**, **GloVe** ou **transformers** (BERT, GPT), et la manière dont la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ s'exprime par une **distance** ou une **similarité** cosinus entre ces vecteurs linguistiques. Les **ambiguités** lexicales ou la possibilité de regrouper plusieurs mots en un **macro-concept** (super-nœud) donnent lieu à un **SCN** linguistique potentiellement complexe, mais dont la dynamique demeure pilotée par la même équation de mise à jour $\omega_{i,j}(t+1) = \dots$. Les liens inter-modaux, notamment image–texte, prennent alors corps dans la matrice ω , dès lors que $\phi_{\text{img}}(\mathbf{x}_i^{(\text{visuel})})$ et $\phi_{\text{txt}}(\mathbf{x}_j^{(\text{texte})})$ ont été projetés dans un espace latent permettant de définir S .

La **section 8.5 (Audio)** généralise encore plus l'approche à la **dimension acoustique**. Les entités y sont des **segments** sonores, décrits par des **features** telles que des **MFCC**, des **spectrogrammes** ou des embeddings audio (par exemple wave2vec). De nouveau, on retrouve la même formule DSL, et la question essentielle consiste à élaborer $S(\mathcal{E}_i^{(\text{audio})}, \mathcal{E}_j^{(\text{audio})})$ pour des **paires** purement acoustiques, ou à formaliser $S(\text{audio,image})$ ou $S(\text{audio,texte})$ pour les liaisons **cross-modales**. Les **clusters** ainsi produits peuvent recouvrir un large éventail de combinaisons comme, par exemple, un groupe de sons liés à des images cohérentes ou à des tokens textuels désignant ces mêmes sons.

Enfin, la fin de la **section 8.5** (et l'ouverture plus large du chapitre 8) mentionne la possibilité de traiter d'**autres modalités**, qu'il s'agisse de signaux physiologiques, de capteurs embarqués ou d'informations structurées. Le **DSL** conserve sa puissance d'**auto-organisation** tant que l'on parvient à définir, pour toute paire $\mathcal{E}_i, \mathcal{E}_j$, une **synergie** $S(i,j)$ traduisant la compatibilité ou la co-occurrence entre ces deux entités.

Cet enchaînement (cf. section 8.2 sur les principes de la fusion, section 8.3 sur la vision, section 8.4 sur le langage, section 8.5 sur l'audio) reprend la **philosophie** d'ensemble décrite dans la **section 8.1.3.1**. Chacune de ces parties approfondit la logique d'un **SCN** multimodal et insiste sur les **défis** mathématiques (normalisation, sélection parcimonieuse des liaisons, mécanismes d'inhibition pour éviter l'explosion du nombre de liens) et sur les **bénéfices** concrets (formation de **clusters** pertinents, émergence de macro-nœuds représentant des concepts multimédias). Les références directes aux **chapitres 5** (architecture SCN), **7** (optimisations avancées) et **6** (approche multi-échelle) garantissent une cohérence interne, montrant comment la construction pratique des poids $\omega_{i,j}$ s'intègre dans la **dynamique** du **DSL** pour gérer un large éventail de **flux** simultanés.

8.1.3.2. Rappel du lien avec Chap. 5 (architecture SCN) et Chap. 7 (optimisations)

Le présent **chapitre 8**, consacré à la **dimension multimodale** du **Deep Synergy Learning (DSL)**, s'inscrit dans la continuité des **fondations** établies aux **chapitres 5** et **7**. Il est donc indispensable d'en rappeler la **cohérence** et de

souligner la façon dont les **modules** et la **dynamique** décrits auparavant constituent le **socle** permettant de gérer, avec efficacité, la **fusion** de plusieurs **modalités** (image, texte, audio, capteurs divers, etc.). Les deux grandes **références** sont d'une part le **Chap. 5**, qui expose l'**architecture** du **Synergistic Connection Network** (SCN), et d'autre part le **Chap. 7**, qui introduit les **méthodes d'optimisation** et d'**adaptation** destinées à maîtriser la complexité du réseau et à éviter des minima locaux.

A. Lien avec le Chap. 5 : l'architecture SCN comme fondement pour la multimodalité

L'**architecture** SCN, telle qu'elle est décrite au **Chap. 5**, prévoit un **noyau** central où réside la matrice $\{\omega_{i,j}\}$, ainsi que des **modules** chargés, entre autres, du **calcul de la synergie**, de l'**inhibition** ou de la **distribution** des entités. Dans un contexte **multimodal**, cette modularité devient un levier essentiel. Chaque **modalité** (vision, texte, audio, capteurs, etc.) peut être traitée de manière spécialisée par un **module** apte à convertir les données brutes en **entités** internes \mathcal{E}_i (vecteurs de caractéristiques, représentations symboliques), puis à appeler la fonction de **synergie** adaptée. Les **pondérations** $\omega_{i,j}$ sont alors mises à jour au sein du **noyau** unique, garantissant l'**unification** du réseau.

Mathématiquement, la dynamique demeure commandée par la relation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

où la **fonction** $S(\mathcal{E}_i, \mathcal{E}_j)$ dépend de la **modalité** ou du **couple** de modalités concerné. Les dispositions décrites au **Chap. 5** (sections 5.4 et 5.7) permettent d'héberger, dans un **même SCN**, des entités visuelles, textuelles ou auditives, tout en conservant une **architecture** modulaire. Les sous-SCN (voir section 5.7.1.1) peuvent traiter localement chaque modalité, tandis qu'un **super-nœud** (5.7.2) orchestre la **fusion** en reliant ces sous-réseaux entre eux.

Cette **organisation** fournit un **socle** systématique pour déployer le **DSL** sur de grands volumes de données multimodales, en évitant qu'une seule et même matrice ω ne devienne ingérable. Le **Chap. 5** met en évidence cette **vision** modulaire qui se révèle fondamentale dès que l'on aborde, comme au **chapitre 8**, la question de la **synergie** entre plusieurs **canaux** (image, texte, audio) dans un **même SCN**.

B. Lien avec le Chap. 7 : méthodes d'optimisation et adaptation pour la multimodalité

Le **Chap. 7** propose diverses **méthodes** destinées à **optimiser** la configuration d'un SCN, à **éviter** une prolifération de liaisons « moyennes » et à **échapper** aux minima locaux. Cette problématique est d'autant plus **cruciale** en contexte **multimodal**, où le nombre total d'entités n peut être très important (car on accumule les entités issues de chaque modalité) et où la matrice ω peut compter $O(n^2)$ liens.

Les mécanismes décrits au **Chap. 7** sont particulièrement **pertinents** ici. Les techniques de **parsimonie** et d'**inhibition** avancée (voir 7.4) limitent le risque qu'une entité se connecte faiblement à trop de nœuds, ce qui serait encore plus probable lorsque l'on gère plusieurs canaux. Sur le plan **mathématique**, ces mécanismes se traduisent par l'ajout d'un **terme** de régulation dans la dynamique, par exemple une **pénalisation**

$$\gamma \sum_{k \neq j} \omega_{i,k}(t),$$

destinée à forcer la compétition entre les liaisons sortantes d'un même nœud \mathcal{E}_i . Sans une telle régulation, la dynamique

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

risquerait de laisser trop de liens inter-canaux subsister, créant une **surcharge** et nuisant à la **lisibilité** des **clusters** émergents.

Les **techniques** de **recuit simulé** (7.3) ou d'**heuristiques globales** (7.5) jouent, en outre, un rôle important pour résorber les configurations localement bloquées. Par exemple, il se pourrait qu'une association image–texte se soit figée dans un cluster inapproprié ; l'injection d'un **bruit** contrôlé, fonction d'une « température » qui décroît au fil du temps, peut libérer la structure et lui permettre de se réorienter vers un alignement plus juste. Cette capacité d'**exploration** stochastique se révèle précieuse lorsque la dimension du problème (nombre d'entités multimodales) est très élevée.

Le **Chap. 7** aborde enfin l'**apprentissage continu** (7.6), qui autorise l'intégration incrémentale de nouveaux nœuds \mathcal{E}_{new} au sein du SCN sans tout réentraîner depuis zéro. Cela s'avère décisif lorsque les flux multimodaux arrivent en continu (ajout progressif de nouvelles images, de nouveaux segments textuels, etc.). Le DSL peut ainsi élargir progressivement sa matrice $\{\omega_{i,j}\}$ et adapter les liens existants en suivant la même équation d'**auto-organisation**, sous réserve de quelques précautions (par exemple, une normalisation adaptative). Les méthodes décrites dans ce même chapitre (monitoring de la densité, insertion locale) forment ainsi un **complément** indispensable pour assurer la **flexibilité** et la **réactivité** d'un SCN multimodal.

Conclusion (8.1.3.2)

Le **chapitre 8**, qui s'attaque à la **fusion** de plusieurs **modalités** (vision, langage, audio...), ne peut se passer des **concepts** exposés aux **chapitres 5 et 7**. D'une part, l'**architecture SCN** du **Chap. 5** fournit la **charpente** modulaire et distribue la responsabilité du calcul S entre des **modules** adaptés à chaque canal, tout en assurant une unification dans le **noyau** central. D'autre part, les **méthodes d'optimisation** et d'**adaptation** du **Chap. 7** s'avèrent essentielles pour maîtriser la **dynamique** d'un SCN multimodal de grande ampleur ; elles permettent aussi d'**éviter** que la fusion d'entités visuelles, textuelles et sonores ne dérive en un graphe démesurément dense et piégé dans des minima locaux. L'ensemble illustre la **cohérence** globale de la démarche DSL et la **pertinence** de ces briques conceptuelles pour aborder la **multimodalité** de manière systématique.

8.2. Principes Généraux de la Fusion Multimodale dans le DSL

Le **DSL** (Deep Synergy Learning) ne se limite pas à agréger des entités d'une seule nature (par exemple, uniquement textuelles ou uniquement visuelles). Sa force réside dans la **capacité** à traiter différentes **modalités** simultanément (images, textes, sons, signaux capteurs, etc.) et à **auto-organiser** les pondérations $\omega_{i,j}$ en fonction de la **synergie inter-modale**. Dans ce cadre, la synergie $S(i,j)$ ne représente plus seulement une “similarité” habituelle (p. ex., cosinus entre vecteurs d'image), mais peut impliquer la **correspondance** ou la **complémentarité** entre deux modalités distinctes.

8.2.1. Notion de Synergie Inter-Modale

8.2.1.1. Définition : comment $S(i,j)$ relie une entité “visuelle” \mathcal{E}_i et une entité “textuelle” \mathcal{E}_j ?

Dans le cadre d'un **DSL** (Deep Synergy Learning) visant à prendre en compte plusieurs **modalités** (par exemple la vision et le texte), il est nécessaire de définir soigneusement la **synergie** entre une **entité** \mathcal{E}_i issue du canal **visuel** et une **entité** \mathcal{E}_j issue du canal **textuel**. De manière générale, le **Synergistic Connection Network** (SCN) regroupe un ensemble d'entités $\{\mathcal{E}_1, \mathcal{E}_2, \dots\}$ provenant de différents canaux ; chacune \mathcal{E}_i s'appuie sur une **représentation** (embedding, vecteur de caractéristiques, etc.). Lorsqu'on compare deux entités $\mathcal{E}_i \in \text{Vision}$ et $\mathcal{E}_j \in \text{Texte}$, on cherche à construire une **fonction** $S(\mathcal{E}_i, \mathcal{E}_j)$ reflétant le **degré** de correspondance ou de compatibilité entre elles.

A. Entités Visuelles et Entités Textuelles

D'un côté, des **entités visuelles** peuvent correspondre à des *patches* (régions extraites d'une image), à des *objets détectés* (boîtes englobantes fournies par un détecteur) ou à des *embeddings* (vecteurs de caractéristiques produits par un réseau convolutionnel type **ResNet**, **VGG** ou autre). On associe ainsi à chaque entité $\mathcal{E}_i^{(\text{visuel})}$ un vecteur $\mathbf{v}_i \in \mathbb{R}^d$, obtenu en traitant l'image (ou sa région) à l'aide d'un **CNN**.

De l'autre côté, des **entités textuelles** se déclinent sous forme de *mots*, de *tokens* de phrase, de syntagmes ou même de segments de texte plus longs (ex. clauses ou phrases entières). Il est usuel de les représenter par un **embedding** $\mathbf{t}_j \in \mathbb{R}^{d'}$, par exemple un vecteur Word2vec, GloVe ou BERT. L'entité $\mathcal{E}_j^{(\text{texte})}$ encapsule donc cette représentation vectorielle, sur laquelle on va opérer pour définir la similarité.

B. Construction de la Synergie Inter-Modale $S(\mathcal{E}_i, \mathcal{E}_j)$

Pour lier un vecteur visuel $\mathbf{v}_i \in \mathbb{R}^d$ à un vecteur textuel $\mathbf{t}_j \in \mathbb{R}^{d'}$, on doit concevoir une **fonction** de correspondance, notée $f(\mathbf{v}_i, \mathbf{t}_j)$, qui renvoie une **valeur** de similarité ou de complémentarité dans \mathbb{R}^+ ou $[0,1]$. Il existe plusieurs stratégies :

569. Approche par projection dans un espace commun :

Une méthode fréquente consiste à définir deux matrices (ou opérateurs) de projection W_v et W_t qui amènent respectivement \mathbf{v}_i et \mathbf{t}_j dans un même espace latent \mathbb{R}^D . On peut alors calculer un **produit scalaire** :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \sigma \langle W_v \mathbf{v}_i, W_t \mathbf{t}_j \rangle,$$

où σ est une **activation** (par exemple une sigmoïde) garantissant un score dans $[0,1]$. Cette démarche s'apparente aux modèles “vision-language” qui apprennent un **espace commun** où images et textes se rapprochent quand ils désignent le même concept.

570. Similarité cosinus :

Une autre approche, plus directe, consiste à **normaliser** \mathbf{v}_i et \mathbf{t}_j puis à évaluer leur **cosinus** :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \cos(\phi_{\text{vis}}(\mathbf{v}_i), \phi_{\text{txt}}(\mathbf{t}_j)),$$

la fonction ϕ_{vis} et ϕ_{txt} pouvant être de simples normalisations ou des transformations apprises. Le score \cos proche de 1 indique une forte corrélation (même direction), un score proche de 0 suggère une quasi-orthogonalité. Cette évaluation succincte est souvent suffisante pour générer un **degré** de synergie.

571. Autres fonctions :

On peut imaginer des **kernels** plus riches (RBF, polynomial) ou des **modèles** de co-occurrence statistique, si l'on dispose d'un corpus où images et textes sont co-présents. D'un point de vue **DSL**, toute fonction $S(\mathbf{v}_i, \mathbf{t}_j) \geq 0$ est recevable, dans la mesure où elle fournit un **signal** local pour renforcer ou affaiblir les pondérations $\omega_{i,j}$.

C. Complémentarité et Corrélation sémantique

Lorsque le **texte** “cat” et le **patch** d'image représentant un **chat** partagent des **features** sémantiques similaires (que ce soit via un produit scalaire ou un cosinus élevé), on obtient $S(i, j)$ important. Inversement, un mot “banana” en face d'une voiture visuelle génère $S(i, j) \approx 0$. Cette **quantification** détermine ensuite la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$, ce qui **auto-organise** le **SCN** de manière à **associer** des entités textuelles et visuelles vraiment liées.

D. Cas d'Exemple

Dans un scénario de **captioning d'image**, on peut disposer d'entités $\{\mathcal{E}_i^{(\text{image})}\}$ représentant différents objets (patchs) et d'entités $\{\mathcal{E}_j^{(\text{texte})}\}$ représentant des **mots** ou **n-grams**. Le **DSL** renforce les liens $\omega_{i,j}$ lorsque l'on détecte une haute **synergie** S . Les **clusters** (image + mots) qui en résultent indiquent la correspondance “ce patch de l'image est un chat, associé au mot ‘cat’”. Similairement, en **recherche cross-modale**, une requête “tower in Paris” sera reliée à l'image de la Tour Eiffel si leurs embeddings respectifs convergent vers un **score** S élevé.

E. Propriétés Mathématiques et Localité/Globalité

La **dimension** des espaces de représentation (\mathbb{R}^d vs. $\mathbb{R}^{d'}$) varie selon la modalité. Pour **harmoniser**, on recourt à un **mapping** $\varphi: \mathbb{R}^d \times \mathbb{R}^{d'} \rightarrow \mathbb{R}$. De plus, la **localité** ou **globalité** de la comparaison est un choix d'implémentation. Il est possible de comparer un **patch** visuel à un **mot** précis (localité fine) ou bien la totalité de l'image à une **phrase** complète (globalité). Dans un **SCN**, rien n'empêche de combiner ces deux approches si l'on souhaite modéliser plusieurs niveaux.

La **synergie** S peut en outre **évoluer** avec le temps, si un module interne affine les espaces d'embeddings. Le **DSL** s'ajuste alors naturellement, car la mise à jour $\omega_{i,j}$ dépend de la valeur courante de S .

F. Intérêt pour le DSL

L'introduction d'une **synergie inter-modale** est un levier clé pour conférer au **DSL** sa capacité de **fusion** multimédia. L'**auto-organisation** demeure pilotée par la même règle locale, mais la **fonction** S qui la sous-tend diffère selon les paires (visuel–textuel, texte–texte, visuel–visuel, etc.). Par ce biais, le **SCN** peut simultanément :

- **Former** des **clusters** multimodaux (image(s) + mot(s))
- **Relier** spontanément de nouvelles entités textuelles ou visuelles en fonction de leur score de similarité
- **Explorer** un large spectre d'arrangements possibles, s'adaptant ainsi à une grande diversité d'images, de mots et de combinaisons

Cela se traduit par un **réseau** (**SCN**) où, sans supervision explicite, les noeuds visuels et textuels s'**agrègent** selon leur proximité ou leur complémentarité.

Conclusion

La **synergie** $S(\mathcal{E}_i^{(\text{visuel})}, \mathcal{E}_j^{(\text{texte})})$ constitue la **pierre angulaire** de la **fusion** entre canaux “image” et “texte” dans un **SCN**. Elle encapsule, sur le plan **mathématique**, la notion de “**degré de correspondance**” ou de “**cohérence**” entre deux modalités, via des embeddings et des fonctions de similarité adaptées. Cette valeur, réinjectée à chaque itération dans la mise à jour $\omega_{i,j}$, oriente de façon **auto-organisée** la formation de liens forts ou faibles entre entités visuelles et textuelles, créant ainsi des **clusters** inter-modaux qui se stabilisent lorsque la synergie persiste. Sous cet angle, un **DSL** vision–texte peut donc découvrir et consolider les associations “chat” / “cat”, “banana” / “banane”, etc., se passant d’un pipeline artificiel prédéfini, et laissant la **dynamique** de pondérations ω gérer l’**émergence** des correspondances sémantiques.

8.2.1.2. Exemples : similarité d'un label texte avec un embedding d'image, correspondance audio-vidéo, etc.

La **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ peut relier des entités de différentes **modalités**, comme un **label textuel** et un **embedding** d’image, ou encore un **segment audio** et un **extrait vidéo**. Ces correspondances s’inscrivent dans le cadre du **DSL** (Deep Synergy Learning) appliqué à plusieurs canaux (texte, vision, audio, etc.) et permettent au **Synergistic Connection Network** (SCN) de gérer de manière **auto-organisée** des données hétérogènes. Les deux exemples présentés ci-après (texte–image et audio–vidéo) illustrent la façon dont se calcule la **synergie** cross-modale et comment cette valeur S est ensuite réinjectée dans la **dynamique** du DSL pour renforcer ou affaiblir les liaisons $\omega_{i,j}$.

A. Similarité Texte–Image : Label Textuel vs. Embedding d’Image

L’association entre un **mot** (ou un ensemble de mots) et la **représentation** vectorielle d’une image (qu’il s’agisse d’une image complète ou d’un patch visuel) illustre un premier cas d’usage du **Deep Synergy Learning (DSL)** dans un contexte multi-modal. Il s’agit de mettre en place un **SCN** (Synergistic Connection Network) où les entités d’intérêt, notées $\mathcal{E}_i^{(\text{img})}$ et $\mathcal{E}_j^{(\text{txt})}$, se trouvent reliées par des poids $\omega_{i,j}$ dont l’évolution dépend d’une **mesure** de similarité inter-modale.

Dans un cadre sub-symbolique, le **vecteur** visuel \mathbf{v}_{img} provient généralement d’un **réseau** de neurones spécialisé (CNN, ResNet, ViT), qui transforme l’image initiale en un embedding $\mathbf{v}_{\text{img}} \in \mathbb{R}^d$. Ce vecteur capture des caractéristiques comme la forme, la texture ou la répartition d’objets, et constitue l’entité $\mathcal{E}_i^{(\text{img})}$. Parallèlement, on considère un **label** textuel, par exemple “cat”, traduit lui aussi en vecteur $\mathbf{v}_{\text{txt}} \in \mathbb{R}^m$ grâce à un modèle comme Word2Vec, GloVe ou BERT. La dimension m peut diverger de d ; pour unifier les représentations, on applique souvent une **transformation** linéaire $\mathbf{W}: \mathbb{R}^m \rightarrow \mathbb{R}^d$, aboutissant au vecteur $\mathbf{v}_{\text{txt}}' = \mathbf{W} \mathbf{v}_{\text{txt}}$.

Dans un **SCN** multimodal, la **synergie** entre ces deux entités, notée $S(\mathcal{E}_{\text{img}}, \mathcal{E}_{\text{txt}})$, se définit usuellement à partir d’une **similarité** cosinus dans \mathbb{R}^d :

$$S(\mathcal{E}_{\text{img}}, \mathcal{E}_{\text{txt}}) = \frac{\mathbf{v}_{\text{img}} \cdot \mathbf{v}_{\text{txt}}'}{\|\mathbf{v}_{\text{img}}\| \|\mathbf{v}_{\text{txt}}'\|}.$$

Si cette valeur se montre élevée, alors la **pondération** $\omega_{\text{img}, \text{txt}}(t)$ tendra à croître selon la règle de mise à jour présentée en **section 2.2.2**, à savoir

$$\omega_{\text{img}, \text{txt}}(t+1) = \omega_{\text{img}, \text{txt}}(t) + \eta [S(\text{img}, \text{txt}) - \tau \omega_{\text{img}, \text{txt}}(t)].$$

Un **mot** comme “chat” verra donc son lien ω renforcer avec l’entité visuelle représentant effectivement un chat, tandis qu’il restera faible pour un patch d’image figurant plutôt une voiture. Au fil des itérations, cette dynamique **auto-organisée** amène le réseau à regrouper de façon cohérente les entités visuelles et textuelles.

B. Correspondance Audio–Vidéo : Aligner un Segment Sonore et un Flux Visuel

Un deuxième exemple typique concerne la mise en correspondance d’un **segment** audio avec un extrait **vidéo** au sein d’un SCN multimodal. Il s’agit ici de repérer si un certain $\mathcal{E}_{\text{audio}}$, qui peut symboliser un bruit (comme le moteur d’une

voiture, le chant d'un oiseau ou un klaxon), s'aligne sur \mathcal{E}_{vid} , c'est-à-dire un lot de frames ou un embedding vidéo représentant la même réalité visuelle.

Du côté **audio**, on segmente le signal en extraits (frames temporelles ou segments) et l'on calcule un **embedding** $\mathbf{v}_{\text{audio}} \in \mathbb{R}^p$ à l'aide de méthodes comme les MFCC (coefficients cepstraux en fréquence mél), la transformation en spectrogrammes ou encore des modèles avancés de type Wav2Vec. Du côté **vidéo**, on peut recourir à un CNN 2D + temps ou à un transformeur vidéo pour extraire un vecteur $\mathbf{v}_{\text{vid}} \in \mathbb{R}^q$. Dans l'hypothèse d'une comparaison directe, on projette $\mathbf{v}_{\text{audio}}$ et \mathbf{v}_{vid} dans un **espace latent** commun \mathbb{R}^d , puis on définit la synergie par une gaussienne ou un cosinus :

$$S(\mathcal{E}_{\text{audio}}, \mathcal{E}_{\text{vid}}) = \exp(-\alpha \|\mathbf{v}_{\text{audio}}' - \mathbf{v}_{\text{vid}}'\|^2), \quad \text{ou} \quad \cos(\mathbf{v}_{\text{audio}}', \mathbf{v}_{\text{vid}}').$$

On peut également inclure un **facteur temporel**, en considérant qu'un segment audio est d'autant plus apte à renforcer le lien $\omega_{\text{audio}, \text{vid}}$ qu'il se superpose temporellement à la séquence vidéo analysée. La règle de mise à jour DSL vient alors renforcer ou atténuer les liaisons en fonction de la pertinence détectée. Si dans une scène sportive, un bruit de foule coïncide avec la vue d'un stade, $\omega_{\text{audio}, \text{vid}}$ s'accroît et consolide un **cluster** rassemblant cet extrait vidéo et ce fragment sonore.

C. Généralisation et Exploitation dans le DSL

De façon plus générale, le **DSL** se prête à la **fusion** de plusieurs modalités, pas seulement l'image ou l'audio, mais aussi le texte, la métadonnée (localisation, date, etc.) ou encore la mesure capteur (température, mouvement, etc.). Chaque **modalité** se voit associée à un **embedding** adapté, et l'on définit une fonction de **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pour chaque couple d'entités (i, j) . Les modalités "intra-catégorie" (texte–texte, image–image, etc.) utilisent des métriques classiques (cosinus, distance exponentielle), tandis que les modalités "inter-catégorie" (texte–image, audio–vidéo, etc.) font l'objet de projections ou de mécanismes d'alignement plus complexes.

La **mise à jour** de la pondération $\omega_{i,j}$ demeure, dans tous les cas, fidèle au schéma :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)],$$

tel que prescrit en [section 2.2.2](#). Ce processus **auto-organisé** entraîne une **polarisation** des liens en faveur des combinaisons $(\mathcal{E}_i, \mathcal{E}_j)$ qui révèlent une affinité notable. Cette convergence se traduit alors par la formation de **clusters** multimodaux, reliant différents types d'entités s'il existe une cohérence sémantique ou temporelle commune. Par exemple, un cluster peut englober un segment sonore, un patch visuel et un mot (ex. "car"), tous se trouvant "mutuellement cohérents". Les règles de **parsimonie** (coupes de liens faibles) et les **heuristiques** comme le recuit simulé (voir chap. 7.3) ou l'inhibition adaptative (chap. 7.4) contribuent à éviter la sur-densification du graphe et à franchir les minima locaux.

L'**exploitation** de ces clusters multimodaux concerne de nombreux cas d'usage, notamment l'**annotation automatique**, la **détection d'événements audio-visuels**, la **recherche cross-modale**, la **recommandation de contenus**, ainsi que la **construction d'un réseau sémantique généraliste**, où chaque modalité contribue à une représentation commune des mêmes objets. Le DSL agit comme un **socle unificateur**, puisqu'il régit la **dynamique** des pondérations ω d'une manière cohérente et distribuée, mettant en évidence les **associations** saillantes par renforcement local des synergies.

Conclusion

L'exemple du **label textuel** confronté à un **embedding** d'image, ou du **segment audio** comparé au **flux vidéo**, illustre la **logique** sous-jacente dans le DSL multimodal. On conçoit une fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ capable de **mesurer** la similarité (cosinus, distance exponentielle, etc.) entre des embeddings potentiellement hétérogènes. Cette valeur, injectée dans la règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)],$$

garantit l'**auto-organisation** du SCN : lorsque deux entités de canaux différents "se ressemblent" ou coïncident sémantiquement, leurs liens ω se consolident. Ce principe de **synergie** inter-modale, répété pour l'ensemble des

couples (i, j) , peut déboucher sur une structuration globale du réseau, en formant des **clusters** cross-modaux d’images, de mots, de sons, ou même de signaux plus exotiques, à condition de pouvoir définir un **embedding** pour chacun et une **fonction S** adaptée.

8.2.2. Avantages du DSL pour la Fusion

Dans la fusion multimodale, l’objectif est de **combiner** différents types de données (image, texte, audio, capteurs...) pour obtenir une **compréhension** ou une **représentation** plus riche du même concept. Le **Deep Synergy Learning (DSL)**, avec sa logique d’auto-organisation (chap. 1–7), se prête particulièrement bien à cette tâche. En effet, plutôt que de passer par un pipeline figé — où chaque modalité est traitée séparément puis “collée” en sortie —, le DSL propose un **réseau auto-organisé** $\{\omega_{i,j}\}$ qui relie les entités issues de différentes sources en fonction de leur **synergie** (cohérence). Nous présentons ici quelques **avantages** majeurs du DSL pour la fusion multimodale :

572. **Auto-organisation**

573. **Adaptation**

574. **Clusters multimodaux**

8.2.2.1. Auto-organisation

La **force** du **DSL** (Deep Synergy Learning) réside dans sa capacité à opérer de manière **auto-organisée**, c’est-à-dire sans qu’un pipeline rigide ne soit imposé pour “fusionner” deux canaux comme l’image et le texte. Le **Synergistic Connection Network** (SCN) explore plutôt ses propres configurations en **renforçant** ou en **affaiblissant** les pondérations $\omega_{i,j}$ entre entités. Lorsque la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ s’avère élevée, un **lien** solide se crée **spontanément**. À l’inverse, si la synergie est jugée faible, la liaison $\omega_{i,j}$ décroît jusqu’à être négligeable. Cette section illustre comment un flux d’images et un flux textuel se **coordonnent** d’eux-mêmes dans ce **contexte** auto-organisé, selon la règle dynamique du DSL.

A. Formation Spontanée de Liens

Il existe un cas très simple qui illustre l’idée de **formation** spontanée de liaisons dans un **SCN** (Synergistic Connection Network) multimodal. Imaginons qu’une entité \mathcal{E}_{img} représente une **image** contenant un chat, tandis qu’une autre entité \mathcal{E}_{txt} correspond au **mot** “cat”. Supposons en outre que ces entités sont décrites par des **embeddings** vectoriels, l’un pour la partie visuelle, l’autre pour le label textuel. Quand les deux embeddings affichent une **similarité** notable, on peut formaliser cette correspondance via une **synergie** :

$$S(\mathcal{E}_{\text{img}}, \mathcal{E}_{\text{txt}}) = \cos(\phi_{\text{vis}}(\mathbf{v}_{\text{img}}), \phi_{\text{txt}}(\mathbf{v}_{\text{txt}})),$$

où ϕ_{vis} et ϕ_{txt} projettent respectivement l’image et le mot dans un espace commun ou au moins comparable. Si ce score dépasse un **seuil** θ , on infère qu’il existe une proximité sémantique marquée entre l’image “chat” et le label “cat”. Dans la logique du **DSL**, la mise à jour de la pondération ω liant ces deux entités s’effectue suivant la règle de la **section 2.2.2**, c’est-à-dire :

$$\omega_{\text{chat}, \text{cat}}(t+1) = \omega_{\text{chat}, \text{cat}}(t) + \eta [S(\text{chat}, \text{cat}) - \tau \omega_{\text{chat}, \text{cat}}(t)].$$

Tant que la **synergie** $S(\text{chat}, \text{cat})$ demeure élevée, la valeur de $\omega_{\text{chat}, \text{cat}}$ se renforce d’elle-même au fil des itérations. Il n’est pas nécessaire de définir un pipeline rigide où l’image serait classée ou labellisée de façon programmée. C’est la **proximité** vectorielle (embedding visuel vs. embedding textuel) qui **guide** la convergence de $\omega_{\text{chat}, \text{cat}}$. Ainsi, l’apparition d’un cluster reliant “image de chat” et “mot cat” se produit **spontanément**, traduisant la correspondance sémantique identifiée.

B. Émergence sans Pipeline Figé

La spécificité du **DSL** réside dans le fait qu'il **s'affranchit** d'une architecture imposant un ordre de traitement, du type « image → réseau → label ». Au contraire, le **SCN** se présente comme un réseau plus général (graphe) où chaque entité – qu'elle soit une image, un mot, un segment sonore, etc. – peut potentiellement se connecter à toute autre entité. Les liaisons $\omega_{i,j}$ s'organisent localement en fonction de la **valeur** de synergie $S(i,j)$.

Sur le plan **mathématique**, un **processus** auto-organisé remplace l'enchaînement séquentiel d'un pipeline. À chaque itération, la mise à jour locale

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

se produit pour toutes les paires (i,j) . Si un certain couple image–texte, comme “chat” / “cat”, jouit d'une synergie supérieure, alors $\omega_{\text{chat, cat}}$ augmentera progressivement. Le résultat final est une **stabilisation** autour de valeurs élevées pour les liens cohérents, et de valeurs faibles (voire coupées) pour les autres. Aucun pipeline de classification explicite n'est donc imposé ; c'est le **DSL** qui, selon la **section 2.2.3** (parsimonie) et la **section 2.2.4** (états internes éventuels), régule localement la compétition entre liaisons.

C. Exemple : Images et Légendes

Un cas d'école correspond à un dataset contenant des **images** ainsi que des **phrases** de description (légendes). Chaque image reçoit un embedding, et chaque phrase est convertie en un vecteur du même espace ou d'un espace comparable. Une fonction de **synergie** $S(\text{img}, \text{légende})$ mesure la similarité entre l'embedding global de l'image et celui de la légende. Le **DSL** renforce alors les liaisons $\omega_{\text{img, légende}}$ chaque fois que la correspondance apparaît réellement fondée. C'est ainsi qu'un petit ensemble de légendes adéquates se voit rapproché d'une image donnée, tandis qu'une image très proche se concentre sur les mêmes légendes.

En pratique, cela se traduit par l'émergence de **clusters** associant plusieurs images quasi identiques ou apparentées et leurs légendes communes ou synonymes. L'**auto-organisation** rend ce système aisément extensible : si l'on ajoute de nouvelles images et de nouvelles légendes, le **SCN** résout de nouveau la dynamique de ω afin de former des liens plus ou moins forts en fonction du score de **synergie**. Aucune étape d'alignement strict n'est nécessaire ; la **cohérence** mesurée localement suffit à construire la structure globale.

E. Bénéfices et Gains

Une telle **auto-organisation** présente plusieurs **avantages**. D'abord, elle est nettement plus **flexible** qu'un pipeline dirigé. Plutôt que d'acheminer obligatoirement l'image vers un module de classification pour obtenir un label, on laisse tout simplement chaque entité (image, mot) “discuter” avec les autres au moyen de $\omega_{i,j}$. Cela aboutit, s'il y a proximité, à un renforcement mutuel, sinon à un affaiblissement. On parle ainsi de **robustesse** : même si un label a été imparfait ou si l'embedding visuel présente du bruit, la cohérence statistique collective au sein du réseau peut compenser cette faiblesse. L'**évolutivité** est aussi un atout notable : si un nouveau canal, par exemple audio, doit être incorporé, il suffit d'introduire les entités audio $\{\mathcal{E}_{\text{aud}}\}$ dans le **SCN**, de définir la fonction de synergie $S(\text{aud, img}), S(\text{aud, txt})$, puis de relancer ou de poursuivre la mise à jour, sans devoir reconstruire l'ensemble du modèle.

F. Perspectives pour la Fusion

Cette philosophie, détachée d'un pipeline séquentiel, se révèle particulièrement intéressante en contextes **multimodaux** ou multi-sources. Dès lors qu'une **fonction** $S(\mathcal{E}_i, \mathcal{E}_j)$ est définie pour chaque couple (i,j) , il devient possible d'étendre la logique à de multiples canaux (texte, audio, image, vidéo, capteurs, etc.). Chaque entité \mathcal{E}_i se contente de procéder à la mise à jour locale de $\omega_{i,j}$ selon la **section 2.2.2**, ce qui favorise la formation de **clusters** multimodaux de grande variété (par exemple, un patch visuel peut se lier à un mot, un segment sonore, voire un paramètre de localisation).

Sur le plan mathématique, tout ce mécanisme relève d'une **dynamique** de descente, éventuellement assortie de recuit simulé (chap. 7.3) ou d'inhibition adaptative (chap. 7.4). Au lieu de coder manuellement la façon dont un CNN se connecte à un embedding lexical, on laisse la **règle** $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ agir dans un **réseau**. Les **clusters** émergent donc de la **somme** des interactions, consolidant les liens forts et élaguant les liens faibles, selon

la logique de parsimonie exposée en **section 2.2.3**. Cette absence de pipeline figé, couplée à la **capacité d'auto-organisation**, distingue nettement le **DSL** des approches traditionnelles de deep learning, tout en s'appuyant sur des fondements mathématiques et algorithmiques clairs.

Conclusion (8.2.2.1)

Le **DSL** ne prescrit pas de pipeline imposé pour marier l'image et le texte. Il s'appuie sur un **processus auto-organisé** : si la **synergie** $S(i, j)$ est élevée, la pondération $\omega_{i,j}$ se renforce, créant des **liens cross-modaux** ; si la synergie est faible, la liaison s'éteint. Cette dynamique engendre l'**appariement** spontané d'entités visuelles, de mots, ou encore d'audio, sans besoin d'une architecture sur-mesure pour chaque combinaison de canaux. L'**auto-organisation** ouvre ainsi la voie à une **fusion multimodale** adaptative, flexible, et prête à accueillir de nouveaux types d'entités ou de nouvelles modalités, simplement en définissant S pour ces canaux et en s'appuyant sur la règle de mise à jour $\omega_{i,j}(t+1) = \dots$.

8.2.2.2. Adaptation : si un flux est manquant (image floue, son corrompu), la synergie s'ajuste, pas de pipeline figé

Lorsque l'on manipule plusieurs **flux** (vision, audio, texte, etc.) dans un **DSL** (Deep Synergy Learning) appliqué à un **Synergistic Connection Network** (SCN), il n'est pas rare qu'un **canal** se révèle inexploitable ou dégradé : une **image** floue ou une **bande sonore** corrompue, par exemple. Dans des architectures **multimodales** classiques (pipelines), la privation d'un flux peut invalider tout le processus. À l'inverse, la **dynamique** auto-organisée du SCN permet une **révision** spontanée des liaisons $\omega_{i,j}$, où la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ s'adapte en conséquence. Cette caractéristique confère au DSL une **grande robustesse** : même si un flux ou un canal fait défaut, le **réseau** poursuit son évolution et oriente ses liens sur les modalités restantes.

A. Absence ou Altération d'un Flux

Il est possible que certains canaux dans un système multimodal se trouvent momentanément inutilisables ou corrompus. Un exemple fréquent consiste en une image floue qui ne fournit plus d'information exploitable, un enregistrement audio pollué par un fort bruit de fond, ou un texte fortement tronqué. Dans une architecture de fusion classique, l'ensemble du pipeline peut être bloqué si le module correspondant à ce flux n'est plus opérationnel. Le **Deep Synergy Learning (DSL)** s'appuie sur un SCN (Synergistic Connection Network) qui ne dicte pas un enchaînement séquentiel fixe. Chaque entité associée à un flux forme un nœud du réseau, et les pondérations $\omega_{i,j}$ évoluent en fonction de la synergie $S(i, j)$.

Lorsque le flux d'une caméra devient inopérant, la **similarité** image–texte ou image–audio décroît, car l'embedding visuel se trouve altéré. Le SCN répercute cette détérioration dans la mise à jour locale, qui tend à réduire la valeur de $\omega_{i,j}$. La disparition de la synergie ne produit pas de rupture globale, le réseau continue de fonctionner en s'appuyant sur les canaux encore valides. Cette flexibilité permet un fonctionnement auto-adaptatif, en autorisant la **rétraction** des liaisons attachées aux flux défaillants et en laissant intacts les autres liens.

B. Ajustement Automatique de la Synergie

La **règle** de mise à jour du DSL demeure la même, à savoir

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Si un flux est altéré, la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ chute parce que l'embedding n'est plus fiable. Cette diminution provoque une baisse progressive des pondérations $\omega_{i,j}$. En conséquence, toute entité \mathcal{E}_i abîmée s'éloigne de ses connexions précédentes. Certains scénarios modélisent cette baisse de fiabilité par un coefficient $\gamma_f(t)$ qui multiplie la synergie, de manière à refléter la perte de qualité. Le DSL intègre alors directement cette information dans la formule de mise à jour et élimine de fait les liens peu utiles.

C. Maintien d'un Réseau Cohérent sans Pipeline Figé

Dans un pipeline traditionnel, la panne d'un capteur peut entraîner un arrêt ou une sortie erronée à cause de la dépendance stricte entre les modules. Dans le cadre du DSL, les entités continuent d'interagir librement et le canal défaillant se trouve simplement écarté par la chute de ω . Les entités non défaillantes tissent toujours leurs liens normalement, ce qui garantit la persistance de l'auto-organisation. Si une image devient floue ou un microphone se met à produire un signal inexploitable, le SCN réduit les connexions y afférentes, en maintenant intactes les autres liaisons.

Cette stratégie rend le DSL bien plus flexible. Le mécanisme ne repose pas sur l'exigence que tous les flux fonctionnent, mais laisse la dynamique régler de façon autonome la participation relative de chaque entité. Les canaux restés opérationnels conservent leurs ω élevés, tandis que ceux devenus temporairement inutiles s'éteignent.

D. Exemples et Intérêt Pratique

Dans un contexte de **robotique**, un robot équipé de caméras, micros et capteurs lidar peut se retrouver avec des images inexploitables par faible luminosité ou obstruction. Le DSL n'interrompt pas pour autant sa fusion de données. Il diminue les pondérations liées au flux visuel et concentre la mise à jour sur les canaux audio ou tactiles. À l'instant où l'image redevient nette, la synergie image–texte ou image–audio se rétablit et les liaisons ω se renforcent naturellement.

Dans le cas d'un **dialogue multimodal**, une plateforme conversationnelle exploite l'audio (reconnaissance vocale) et le texte (suggestions, retours). Si l'audio présente des coupures ou du brouillage, la valeur de $S(\text{audio}, \text{texte})$ se retrouve pénalisée, ce qui fait retomber les liens correspondants. Le système peut alors s'appuyer sur le flux textuel seul, sans bloquer l'intégralité de la session. Cette adaptation locale se produit au sein du SCN, évitant la nécessité d'une réingénierie complète du pipeline.

En **apprentissage continu**, un flux comme $\mathcal{E}_{\text{image}}$ peut fluctuer en qualité au cours du temps. Si un segment vidéo devient trop bruité, la pondération $\omega_{\text{image}, \dots}$ chute. Lorsque la qualité revient à la normale, l'embedding visuel renforce de nouveau sa synergie et la règle DSL restaure une connexion appropriée. Cette gestion fluide de la variabilité illustre la robustesse du SCN face aux changements de conditions d'observation.

Conclusion

L'aptitude du DSL à **s'adapter** aux flux manquants, corrompus, ou temporairement de mauvaise qualité provient de l'absence d'un **pipeline immuable**. Le **SCN** ne s'effondre pas lorsque l'un des canaux est inutile : la **synergie** S associée se voit réduite, et la mise à jour $\omega_{i,j}$ entraîne l'extinction des liens correspondant à ce flux. Les autres canaux demeurent actifs et continuent de **coopérer**. Cette **auto-organisation** confère au DSL une propriété de **résilience** : le traitement multimodal ne repose plus sur la disponibilité systématique de tous les flux, mais s'ajuste en continu pour **exploiter** (ou ignorer) ceux qui sont opérationnels (ou défaillants), garantissant ainsi une **flexibilité** et une **robustesse** accrues dans des scénarios réels où les données, fréquemment, ne sont pas toujours complètes ni fiables.

8.2.2.3. Clusters multimodaux : possibilité de créer des regroupements complexes {image + son + texte} autour d'un concept

L'une des **caractéristiques** notables d'un **DSL** (Deep Synergy Learning) consistant à gérer plusieurs **modalités** (image, audio, texte, etc.) est la faculté de **composer** ou de **fusionner** diverses mesures de **synergie** pour faire émerger des **clusters** intégrant plusieurs canaux à la fois. Plutôt que de traiter un flux visuel de manière isolée, ou de se limiter à la similarité audio–audio, le **Synergistic Connection Network** (SCN) peut, de manière **auto-organisée**, regrouper simultanément des **objets visuels**, des **segments sonores**, des **extraits textuels** (ou d'autres modalités encore), si la synergie cross-modale s'avère suffisamment élevée. Cette dynamique autorise la construction de "**macro-nœuds**" complexes, incarnant un "**concept**" ou un "**événement**" multimodal.

A. Combinaison des modalités pour former un concept

Un Synergistic Connection Network (SCN) manipule des entités issues de plusieurs canaux, comme l'image, l'audio, le texte ou tout autre flux sensoriel. Chaque entité \mathcal{E}_i peut donc renvoyer à un patch d'image, un court segment sonore, un embedding lexical, ou d'autres formes de représentation. Il est possible de fusionner ces diverses modalités en définissant une synergie globale $S_{\text{multi}}(\mathcal{E}_i, \mathcal{E}_j)$. Une formule courante repose sur la somme de contributions partielles associées à chaque canal, par exemple

$$S_{\text{multi}}(\mathcal{E}_i, \mathcal{E}_j) = \alpha S_{\text{img}}(i, j) + \beta S_{\text{aud}}(i, j) + \gamma S_{\text{txt}}(i, j),$$

avec α, β, γ supérieurs à zéro et où chaque fonction $S_{\text{img}}, S_{\text{aud}}, S_{\text{txt}}$ quantifie une similarité intra-modale (entre embeddings visuels, embeddings audio ou embeddings textuels). L'intérêt de cette construction tient dans la possibilité de gérer un score global de synergie même lorsque plusieurs modalités sont présentes. Le mécanisme de mise à jour du DSL reste identique à celui de la section 2.2.2, ce qui signifie que si ce score $S_{\text{multi}}(\mathcal{E}_i, \mathcal{E}_j)$ est élevé, la pondération $\omega_{i,j}$ se renforce.

B. Constitution de clusters complexes {img + aud + txt}

L'agrégation de plusieurs canaux dans un SCN favorise la création de regroupements plus riches. Un triplet formé d'une entité $\mathcal{E}_{\text{image}}$, d'une entité $\mathcal{E}_{\text{audio}}$ et d'une entité $\mathcal{E}_{\text{texte}}$ peut correspondre à un concept unique, comme un "événement de concert" combinant un patch visuel montrant des musiciens sur scène, un segment sonore captant le morceau joué et une phrase textuelle décrivant la chanson. Les liens entre ces entités se voient renforcés par la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{multi}}(i, j) - \tau \omega_{i,j}(t)].$$

Un exemple est un cluster formé par une image de voiture, un segment audio de moteur et un label textuel "car", donnant lieu à des liens élevés entre image–audio–texte dès lors que la synergie multimodale reste cohérente. L'émergence de ces macro-clusters dépasse la liaison de deux canaux isolés et illustre la capacité du DSL à tisser des associations complexes.

C. Bénéfices pour la Représentation

Lorsque trois canaux ou plus se regroupent dans un même cluster, le réseau acquiert une sémantique plus riche. La présence simultanée de plusieurs flux garantit une meilleure robustesse face à la défaillance éventuelle d'un canal, comme une image floue ou un son corrompu, car la synergie entre d'autres flux peut préserver la cohérence du regroupement. Un cluster associant un mot "beach", un motif acoustique d'océan et un patch montrant du sable informe fortement sur l'existence d'un concept de "plage", plus nettement que ne le ferait la prise en compte d'un seul flux.

D. Mise en œuvre pratique

Le calcul des scores inter-modal se révèle nécessairement plus coûteux, puisqu'il faut évaluer la similarité dans différentes combinaisons. Lorsqu'on manipule n entités réparties sur plusieurs canaux, on peut atteindre un nombre important de comparaisons. Dans la pratique, on adopte des heuristiques pour limiter la taille du réseau ou on applique des mécanismes de filtrage pour ne conserver que les candidats probables.

Le paramétrage de la fusion pose la question de la pondération relative de chaque modalité. Le choix d'une somme linéaire $\alpha + \beta + \gamma$ constitue un exemple simple. On peut également recourir à un modèle plus élaboré, comme un réseau neuronal capable de combiner non linéairement les valeurs de similarité venant de chaque canal. Dans tous les cas, la mise à jour locale

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{\text{multi}}(i, j) - \tau \omega_{i,j}(t)]$$

reste la même, mais l'expression de S_{multi} varie en fonction de la stratégie de fusion.

Un exemple concret peut s'observer dans le cas d'un vlog où chaque segment temporel aligne plusieurs frames vidéo, un extrait sonore et des sous-titres textuels. Une fonction de synergie évalue leur cohérence multimodale et la règle DSL tisse des liens élevés si l'on constate un accord sémantique ou temporel. De ce fait, on repère des macro-clusters explicitant par exemple un passage de plage, avec des mots se rapportant à la mer, le son des vagues et les frames

révélant une côte sablonneuse. Cette auto-organisation simultanée dans le réseau favorise un point de vue unifié sur les données, autorisant la classification, la recherche ou l'annotation multimodale sans pipeline figé.

Conclusion

La **formation de clusters** multimodaux {image + son + texte} dans un **DSL** repose sur la **composition** des synergies intra- et inter-modales, potentiellement par une formule $S_{\text{multi}} = f(S_{\text{img}}, S_{\text{aud}}, S_{\text{txt}})$. Cette capacité permet au **SCN** de découvrir et de **stabiliser** des **regroupements** complexes, correspondant à des concepts englobant plusieurs dimensions sensorielles ou sémantiques. Contrairement à un pipeline classique, où la fusion est imposée par architecture, la **dynamique** auto-organisée du DSL laisse chaque flux **coopérer** ou **concurrencer** les autres flux, et initie la formation d'un **macro-nœud** s'il en résulte un signal de similarité suffisamment fort. On obtient ainsi une **intégration** multimodale plus naturelle et plus flexible, et, finalement, des **structures** de grande richesse dans le réseau.

8.2.3. Défis et Écueils

Dans la démarche **multimodale** du DSL (Deep Synergy Learning), où l'on traite simultanément images, textes, signaux audios, voire d'autres sources (capteurs, données symboliques, etc.), plusieurs **défis** surgissent. Cette section (8.2.3) en liste quelques-uns, parmi lesquels figurent (1) l'hétérogénéité forte des représentations, (2) le surcoût computationnel (du fait de la multiplication des entités) et (3) les problèmes de synchronisation temporelle lorsque plusieurs flux doivent être alignés. Avant de détailler les aspects (8.2.3.2, 8.2.3.3), focalisons-nous sur l'**hétérogénéité** des représentations (8.2.3.1).

8.2.3.1. Hétérogénéité forte des représentations (CNN pour images, Transformers pour textes, spectrogrammes pour audio...)

Il est fréquent qu'un Synergistic Connection Network (SCN) appliqué à la fusion multimodale doive gérer des représentations très différentes selon la modalité considérée. Une architecture CNN (ResNet, VGG ou ViT) peut décrire les entités visuelles, tandis qu'un Transformer (BERT, GPT ou similaire) produit des embeddings textuels contextuels, et qu'un spectrogramme ou un autoencodeur incarne l'information audio. Cette variété assure une exploitation spécialisée de chaque flux, mais complique la mise en place d'une synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ unifiée et la dynamique auto-organisée qui en découle.

A. Différentes Dimensions et Structures de Données

Les entités issues de l'image proviennent souvent d'un vecteur ou d'un tenseur élevé en dimension, suite à l'application d'un CNN ou d'un ViT. Les entités textuelles se retrouvent sous forme de word embeddings (Word2Vec, GloVe) ou d'embeddings contextuels (BERT, GPT), de dimension potentiellement différente de celle des embeddings visuels. L'audio peut être encodé sous forme de spectrogrammes, de MFCC ou à travers un modèle de type Wav2Vec. Chaque flux présente donc des caractéristiques uniques, comme la spatialité pour la vision ou la contextualité pour le texte. Cette disparité implique la nécessité de comparer des embeddings qui ne partagent pas la même échelle ni la même signification.

B. Défi d'Alignement Conceptuel

Lorsque le SCN doit évaluer la synergie entre deux entités multimodales, par exemple une image et un texte, il est indispensable d'avoir une fonction de similarité capable de gérer deux représentations très différentes. Sur le plan mathématique, on peut opter pour un espace commun de dimension D où l'on projette successivement les embeddings issus de l'image et du texte, puis effectuer une comparaison par produit scalaire ou cosinus. Il est aussi possible d'utiliser une fonction plus complexe, comme une co-attention ou un module MLP dédié, qui reçoit les deux vecteurs et produit un score de compatibilité. Sans une telle cohérence conceptuelle, le DSL se trouverait en difficulté pour relier des entités hétérogènes au sein d'un même cluster.

C. Variabilité des Extracteurs

Chaque modalité peut se voir associée à un extracteur spécialisé. Un réseau CNN préentraîné sur ImageNet détecte les formes et les textures, un Transformer BERT s'appuie sur les contextes lexicaux, tandis qu'un encodeur audio ou un spectrogramme met en avant la structure fréquentielle du son. Ces modèles produisent des embeddings qui diffèrent dans leurs échelles, leurs dispersions et leurs biais. Le SCN doit composer avec ce foisonnement de dimensions et de signatures, sous peine d'obtenir des pondérations faussées ou incompatibles entre entités. Sur le plan pratique, on peut introduire autant de modules de synergie que de paires de modalités considérées, chacun étant responsable de comparer deux types d'embeddings de manière cohérente.

D. Interaction avec la Synergie dans le DSL

La formule de base du DSL se maintient. On répète la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$. Cependant, la fonction S dépend du couple de modalités en cause. Lorsqu'il s'agit de deux images, on applique une métrique visuelle de type distance euclidienne ou cosinus. Lorsqu'il s'agit d'un audio et d'un texte, on peut recourir à un alignement audio–texte. Le SCN doit donc étiqueter chaque entité selon sa modalité et appeler la bonne fonction de similarité. Cela revient à disposer d'une table recensant les combinaisons possibles (image–image, image–texte, texte–texte, audio–texte, audio–image, etc.) et le module spécifique à employer.

E. Complexité Opérationnelle

Lorsque plusieurs paires de modalités se côtoient, le volume de comparaisons à exécuter peut augmenter considérablement. Il faut évaluer la synergie pour de nombreuses entités issues de chaque canal, ce qui peut coûter cher en temps de calcul et en mémoire si chaque embedding compte plusieurs centaines ou milliers de dimensions. Le DSL peut alors nécessiter des heuristiques ou un filtrage sélectif pour éviter de réaliser la totalité des comparaisons, en particulier dans un scénario à grande échelle. Il faut aussi maintenir la cohérence des embeddings ou recalculer la similarité à la volée pour éviter un stockage excessif.

F. Conséquences pour la Convergence et la Qualité

Une synergie mal calibrée entre deux modalités peut amener le réseau à se fragmenter en sous-réseaux purement unimodaux. Un ajustement judicieux de la fonction S est souvent nécessaire pour encourager le rapprochement multimodal, par exemple via un préapprentissage de type CLIP ou ALIGN qui aligne déjà l'image et le texte. L'usage d'une phase de recuit (chap. 7) ou d'une inhibition adaptative peut également aider le réseau à sortir d'attracteurs locaux et à tisser des liaisons correctes entre flux très différents. L'aboutissement recherché est un ensemble de clusters exploitant pleinement la complémentarité des canaux, mais ce résultat dépend fortement de la façon dont on définit et normalise les fonctions de similarité multimodales.

Conclusion (8.2.3.1)

Lorsque le DSL aborde des **flux** hétérogènes — images encodées par CNN, textes par Transformers, audio par spectrogrammes ou wave2vec — il doit **composer** avec la diversité :

- Les **représentations** varient en **structure, taille, sémantique**,
- Le **calcul** de $S(\mathcal{E}_i, \mathcal{E}_j)$ nécessite un **alignement** inter-modale explicite (mapping vers un espace commun, kernel multimodal, etc.),
- La **complexité** monte : plus de canaux, plus de modules de similarité, plus de paires d'entités à gérer,
- La **convergence** du SCN peut être plus lente et risquer de générer des **sous-clusters** modaux si le système ne facilite pas la cohérence cross-modale.

Ce constat souligne l'importance de définir, pour chaque paire de modalités, des **fonctions** S robustes et de calibrer l'**auto-organisation** de manière à encourager la découverte de correspondances cross-modales pertinentes (chap. 8.2.3.2, 8.2.3.3). On comprend ainsi qu'un **DSL** multimodal n'est pas seulement un réseau ω plus vaste, mais également un **ensemble** de solutions d'**alignement** et de **normalisation** pour confronter les modèles (CNN, Transformer, etc.) les uns aux autres.

8.2.3.2. Coût computationnel : multiplication du nombre d'entités

Lorsqu'un **DSL** (Deep Synergy Learning) est étendu à la **multimodalité** (image, audio, texte, capteurs...), le **Synergistic Connection Network** (SCN) devient rapidement volumineux. En effet, chaque modalité peut générer un nombre considérable de **descripteurs** ou **segments** (patchs d'image, frames audio, tokens textuels, etc.). La **somme** de ces entités n_{total} peut se multiplier par rapport à une approche **monomodale**, accroissant lourdement le **coût** de calcul si l'on applique naïvement une mise à jour $\omega_{i,j}$ sur toutes les paires (i,j) .

A. Formulation mathématique du coût naïf

Le **SCN** se définit sur un ensemble $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ d'entités. Dans une configuration **monomodale**, on peut avoir n de l'ordre de quelques centaines ou milliers. Dans un contexte **multimodal**, on additionne :

575. n_{vision} entités issues de l'analyse visuelle (ex. *patches, feature maps* d'un CNN, keypoints SIFT, etc.),

576. n_{audio} entités extraites du spectrogramme ou d'un embedding type Wav2Vec,

577. n_{texte} entités correspondant à des tokens, n-grams, o

578.u phrases issues d'un embedding BERT/GPT,

579. éventuellement n_{capteurs} pour d'autres flux sensoriels.

La somme

$$n_{\text{total}} = n_{\text{vision}} + n_{\text{audio}} + n_{\text{texte}} + \dots$$

peut atteindre plusieurs milliers ou dizaines de milliers. Or, la mise à jour naïve de la matrice $\{\omega_{i,j}\}$, si elle repose sur l'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

s'effectue potentiellement pour **tous** les couples (i,j) . Cela représente $O(n_{\text{total}}^2)$ calculs de $S(i,j)$ et autant de mises à jour $\Delta\omega$. D'un point de vue **big-O**, ce **carré** du nombre d'entités peut devenir prohibitif lorsque n_{total} s'avère important.

B. Détails sur l'impact du multimodal

Chaque **modalité** amène son **lot** d'entités :

- En **vision**, il est courant d'extraire une **multitude** de descriptors (patchs CNN, bounding boxes, keypoints...),
- En **audio**, on segmente la bande sonore en frames ou en fragments, chacun pouvant être répercuté dans le SCN,
- En **texte**, on peut gérer mots, tokens ou chunks, engendrant parfois des centaines de nœuds pour un seul document.

Au **total**, même si chaque modalité était en soi gérable (ex. 500 entités pour la vision, 500 pour l'audio, 1000 pour le texte), la **somme** peut rapidement s'envoler (ici 2000 entités). La complexité $O(2000^2) \approx 4 \cdot 10^6$ pondérations est déjà non triviale, d'autant que le **calcul** de $S(i,j)$ peut lui-même nécessiter un **embedding** cross-modal.

En outre, le **SCN** inclut non seulement les liaisons $\omega_{i,j}$ entre entités de même modalité (ex. vision–vision), mais **aussi** les liaisons inter-modales (vision–audio, audio–texte, etc.). Cela multiplie encore les possibilités de couples (i,j) . À chaque **itération**, on passe en revue l'ensemble de la matrice ω , ce qui devient **lourd** si l'on conserve la formule classique $\omega_{i,j}(t+1) = \dots$.

C. Perspective algorithmique

Dans la vision **naïve** où l'on met à jour toutes les paires (i, j) , la charge $O(n_{\text{total}}^2)$ rend l'**itération** DSL extrêmement coûteuse. Cet état de fait **justifie** l'usage de méthodes d'**approximation** et d'**optimisation** (chap. 7). Parmi elles :

580. **Sparsification**

Ne conserver que les liens $\omega_{i,j}$ dépassant un certain seuil, ou limiter le calcul de $S(i, j)$ aux plus proches voisins (k-NN). Au lieu de mettre à jour $\omega_{i,j}$ pour chaque paire (i, j) , on ne traite que celles dont la synergie potentielle semble prometteuse.

581. **Heuristiques globales**

Travailler par “blocs” ou “mini-batches” d’entités, réduire la dimension des embeddings via un autoencodeur commun, ou recourir à un **recuit** (voir section 7.3) appliqué sélectivement sur certaines parties du SCN.

582. **Distribution** sur plusieurs machines

Lorsque n_{total} est très grand, on peut **distribuer** les calculs, chaque noeud effectuant la mise à jour de sa section ω_i , ou $\omega_{:,j}$. Ceci n’annule pas la complexité, mais la parallélise.

583. **Filtrage** amont

Avant même de construire le SCN, on peut filtrer les entités ou les paires improbables. Par exemple, on ne compare l’image et l’audio que si leurs timestamps coïncident dans une séquence vidéo, etc. On évite ainsi $O(n_{\text{img}} \times n_{\text{audio}})$ comparaisons inutiles.

D. Conséquences sur la dynamique et la qualité

En limitant $\omega_{i,j}$ aux seules paires “suspectées” de synergie (ou en imposant un “champ de vision” restreint), on sacrifie partiellement la possibilité de **découvrir** des correspondances imprévues. Cependant, en pratique, cet arbitrage est indispensable pour maîtriser le **coût** exponentiel. La **dynamique** DSL demeure valable sur la sous-partie “sparse” du graphe, et on obtient des **clusters** quasi équivalents à ceux du modèle complet, à condition de choisir un **filtrage** ou une **sparsification** judicieux.

Conclusion (8.2.3.2)

La **multiplication** d’entités amenées par la **fusion** multimodale (images, audio, texte, etc.) accroît fortement le **coût** de calcul du DSL, en raison de la structure $O(n^2)$ de la règle de mise à jour $\omega_{i,j}$. Il est donc **crucial** de mettre en œuvre des heuristiques, des mécanismes de **parsimonie** et éventuellement des schémas de calcul **distribué** pour maintenir la **scalabilité** du SCN. Cette problématique illustre que, même si la **synergie** multimodale ouvre la voie à des **clusters** plus riches, elle impose également des **contraintes** en termes de complexité algorithmique, qu’il faut **gérer** attentivement pour rendre le DSL exploitable sur de gros ensembles de données multimédias.

8.2.3.3. Synchronisation Temporelle (Audio–Vidéo)

Lorsque l’on vise à **fusionner** des flux **audio** et **vidéo** dans un cadre **multimodal**, il ne suffit pas d’évaluer la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ en ignorant la **dimension** temporelle : un segment audio doit correspondre, en principe, à la même portion de la séquence vidéo pour qu’ils décrivent un même événement (sons et images simultanés). Cette problématique de **synchronisation** s’ajoute aux considérations de **similarité** ou de **compatibilité** sémantique déjà discutées (voir section 8.2.1.2). Il faut donc intégrer la **variable temps** dans la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ et dans la mise à jour des pondérations $\omega_{i,j}$, afin de former des **clusters** multimodaux audio–vidéo réellement cohérents.

A. Cadre Général de l'Alignement Temporel

Supposons que le flux **audio** soit découpé en trames $\{A_t\}$ d'une durée Δ_a chacune, tandis que le flux **vidéo** est découpé en images ou frames $\{V_\tau\}$ d'une durée Δ_v . Chaque trame audio A_t peut être considérée comme une **entité** $\mathcal{E}_{a,t}$ dans le **SCN**, et chaque frame vidéo V_τ comme une **entité** $\mathcal{E}_{v,\tau}$.

Pour définir la **synergie** $S(\mathcal{E}_{a,t}, \mathcal{E}_{v,\tau})$, il convient de tenir compte de la **proximité temporelle** : si $\tau \Delta_v$ et $t \Delta_a$ représentent des instants très éloignés, la probabilité que l'image et le son décrivent le même événement audiovisuel diminue. On peut formaliser :

$$S(\mathcal{E}_{a,t}, \mathcal{E}_{v,\tau}) = f(\text{similarité}_{\text{contenu}}(A_t, V_\tau), \text{décalage_temps}(\tau \Delta_v - t \Delta_a)),$$

où $\text{similarité}_{\text{contenu}}(A_t, V_\tau)$ apprécie la **compatibilité sémantique** (par exemple, embedding audio vs. embedding vidéo), et décalage_temps pénalise les couples (t, τ) situés trop loin dans l'axe temporel.

B. Approche SCN : Pondérations et Décalage Temporel

Dans la **dynamique** DSL, chaque liaison $\omega_{(a,t),(v,\tau)}$ entre une trame audio $\mathcal{E}_{a,t}$ et une frame vidéo $\mathcal{E}_{v,\tau}$ subit la règle habituelle :

$$\omega_{(a,t),(v,\tau)}(k+1) = \omega_{(a,t),(v,\tau)}(k) + \eta [S(\mathcal{E}_{a,t}, \mathcal{E}_{v,\tau}) - \tau_d \omega_{(a,t),(v,\tau)}(k)],$$

où τ_d est la constante de décroissance (analogique à τ dans la formule DSL, à ne pas confondre avec l'index τ temporel de la vidéo). Le terme

$$S(\mathcal{E}_{a,t}, \mathcal{E}_{v,\tau})$$

se calcule en considérant :

584. **La proximité temporelle** $\Delta T = |\tau \Delta_v - t \Delta_a|$. On peut pénaliser fortement les paires (t, τ) pour lesquelles ΔT dépasse un certain seuil.

585. **La similarité des descripteurs** (spectrogramme vs. frame CNN, par exemple).

Une implémentation simple pourrait être :

$$S(A_t, V_\tau) = \cos(\phi_{\text{audio}}(A_t), \phi_{\text{vid}}(V_\tau)) \times \exp(-\alpha \Delta T),$$

avec un facteur gaussien $\exp(-\alpha \Delta T)$ minorant la synergie si A_t et V_τ sont trop décalés dans le temps.

C. Intégration d'Algorithmes d'Alignement plus Complexes

Lorsque l'audio et la vidéo ne sont pas strictement **échantillonnes** aux mêmes moments (ex. un flux vidéo à 30 FPS et un audio à 16 kHz), il existe plusieurs solutions :

586. **Fenêtre glissante** : on recherche la **frame vidéo** la plus proche temporellement de la trame audio A_t . Les paires plus distantes sont ignorées ou reçoivent une synergie quasi nulle.

587. **Dynamic Time Warping (DTW)** : si la vitesse d'enchaînement vidéo ou audio varie dans le temps (par ex. flux corrompu ou ralentissement), un algorithme DTW peut déterminer la "warp path" qui associe chaque trame audio à une frame vidéo la plus susceptible de correspondre, fournissant un **mapping** $(t \rightarrow \tau)$. Dans le SCN, on peut alors insérer un pénalty si (t, τ) s'écarte de cette warp path.

Le **DSL** s'appuie sur ces mécanismes pour **pondérer** la similarité purement sémantique ($\text{similarité}_{\text{contenu}}$) par un coefficient reflétant la **cohérence** temporelle. Ainsi, le **SCN** ne relie fortement un segment audio $\mathcal{E}_{a,t}$ qu'aux frames vidéo $\mathcal{E}_{v,\tau}$ qui cadrent en temps (et en contenu) avec lui.

D. Bénéfices pour l'Auto-Organisation

Un SCN multimodal où **audio** et **vidéo** s'alignent temporellement amène plusieurs avantages :

588. Clusters audio–vidéo de meilleure qualité

Chaque **événement** (bruit de voiture + vue de la voiture), chaque **action** (parole + mouvement labial) peut être regroupé, renforçant la synergie sur la période temporelle adéquate.

589. Désactivation automatique des liens hors synchronisation

Si un extrait audio se trouve 5 s avant la frame vidéo représentant la même scène, la synergie chute, donc ω ne se renforce pas. Au final, on obtient un **alignement** implicite des entités $(a, t) \leftrightarrow (v, \tau)$ partageant vraiment l'instant ou la courte fenêtre de temps.

590. Émergence de macro-clusters spatio-temporels

En combinant la notion de **similitude** de contenu et la **fenêtre** temporelle, le DSL peut construire un **super-nœud** représentant un événement multimédia : un groupe de frames vidéo, un segment audio correspondant, éventuellement un bloc textuel (sous-titre ou annotation) inséré à la même période.

E. Conclusion (8.2.3.3)

La **synchronisation temporelle** constitue un point crucial lors de la fusion **audio–vidéo** : si l'on n'intègre pas la dimension temporelle, le SCN risque de lier par erreur un segment audio à une portion vidéo qui ne se produit pas simultanément. Les solutions consistent à :

- inclure un **facteur** $\exp(-\alpha \Delta T)$ (ou autre pénalisation) dans la **synergie** $S(A_t, V_\tau)$,
- recourir à des algorithmes d'**alignement** plus sophistiqués (fenêtre coulissante, DTW) pour guider le couplage $(t \leftrightarrow \tau)$.

Cette vision **temporelle** du DSL enrichit la notion d'**auto-organisation** : les **pondérations** $\omega_{(a,t),(v,\tau)}$ se **renforcent** ou se **désactivent** en fonction non seulement de la **similarité** sémantique (embedding audio vs. embedding vidéo), mais aussi de leur **coïncidence** ou **proximité** dans l'échelle du temps. De ce fait, les **clusters** générés par le SCN peuvent refléter des événements multimédias plus fidèles à la **réalité** audio-visuelle.

8.3. Modélisation des Entités Visuelles

Dans le contexte du **DSL multimodal**, la **composante visuelle** (images, vidéos) joue un rôle important : les entités d'information \mathcal{E}_i peuvent être de nature strictement visuelle ou fusionnées avec d'autres modalités (texte, audio, etc.). Pour tirer parti d'un **SCN** (Synergistic Connection Network) manipulant ces entités, il est nécessaire de définir une **représentation** ou "embedding" visuelle pertinente, assurant que la **synergie** $S(\text{image}_i, \text{image}_j)$ reflète au mieux la similarité ou la complémentarité entre deux images.

8.3.1. Représentations d'Images

Au cours des dernières années, l'essor des **réseaux de neurones convolutifs** (CNN) a permis de générer des **embeddings** puissants et robustes pour décrire le **contenu** d'une image. Les sections suivantes détaillent comment on extrait ces vecteurs, comment ils se comparent à des approches plus classiques (points d'intérêt locaux, global pooling, etc.), et quels **formats** ou **dimensions** sont typiquement employés dans la littérature.

8.3.1.1. Embeddings issus de CNN (ex. ResNet, VGG)

Un **CNN** (Convolutional Neural Network) constitue, dans le cadre de la **vision par ordinateur**, l'un des moyens les plus éprouvés pour **extraire** des représentations vectorielles (ou *embeddings*) à partir d'une **image**. Ces embeddings se révèlent utiles pour un **DSL** (Deep Synergy Learning) appliqué à la modalité **image**, car ils fournissent un **vecteur** $\mathbf{f}_i \in \mathbb{R}^d$ permettant de comparer différentes images, de calculer la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ (voir chapitres précédents) et d'**auto-organiser** un **Synergistic Connection Network** (SCN) fondé sur la **similarité** ou la **distance** dans l'espace des **features**.

A. Principe des Convolutions et Extraction de Caractéristiques

Un **réseau** convolutif (CNN) Φ opère sur une **image** \mathbf{x} , typiquement un tenseur $\mathbf{x} \in \mathbb{R}^{H \times W \times 3}$ (RGB). Sur le plan **mathématique**, ce type d'architecture est une **composition** de :

1. **Couches de convolution** : $\text{Conv}(\mathbf{x})$, appliquant des **noyaux** de taille $K \times K$ sur la carte de caractéristiques,
2. **Fonctions d'activation** (ReLU, ELU, etc.),
3. **Pooling** (max pooling, average pooling) pour réduire la dimension spatiale.

En notant σ la non-linéarité (ReLU, p. ex.) et Pool l'opération de pooling, on peut formaliser :

$$\mathbf{f} = \Phi(\mathbf{x}) = \text{Pool}\left(\sigma\left(\text{Conv}_K(\dots(\mathbf{x}))\right)\right).$$

Les convolutions successives extraient des **motifs visuels** (bords, textures, formes), et l'empilement des couches représente une **hiérarchie** de plus en plus abstraite. Ce processus aboutit à une **feature map** finale, ou un **vecteur** $\mathbf{f} \in \mathbb{R}^d$ lorsqu'on "aplatis" les dimensions spatiales.

B. Réseaux Pré-entraînés

Pour obtenir un **embedding** de qualité, la pratique la plus courante consiste à recourir à un **CNN** pré-entraîné sur un vaste corpus d'images. **ImageNet** (1.2 million d'images classées en 1000 catégories) est l'exemple classique : ResNet, VGG, MobileNet, etc., ont été **entraînés** à distinguer ces classes, ce qui leur confère une bonne capacité à capturer des **caractéristiques** génériques (textures, contours, détails d'objets).

On récupère alors, dans le **DSL**, cette fonction Φ déjà apprise. Chaque **image** \mathbf{x} se voit convertie en un **vecteur** $\mathbf{f} \in \mathbb{R}^d$:

$$\mathbf{f} = \Phi(\mathbf{x}),$$

souvent en extrayant la **dernière couche** avant la classification, notée par exemple “pool5” ou “fc7” selon l’architecture. Pour ResNet-50, la dimension $d \approx 2048$. Pour VGG16 (fc7), $d = 4096$. D’autres modèles (EfficientNet, MobileNet) ont des dimensions plus modestes (ex. 1280).

Ces **embeddings** se montrent relativement robustes et “généralistes”, car ils capturent des **motifs** utiles pour reconnaître un large éventail de catégories. Dans un **SCN**, deux images similaires (même classe ou sujet visuel) reçoivent des vecteurs $\mathbf{f}_i, \mathbf{f}_j$ proches, conduisant à un **score** $S(i, j)$ élevé et favorisant la **cohésion** dans un cluster.

C. Opérations Complémentaires

Bien que ce vecteur \mathbf{f}_i soit déjà une représentation, il peut nécessiter des **opérations** supplémentaires.

Réduction de dimension

Pour VGG16, le vecteur 4096-d peut être jugé trop grand. On applique une **PCA**, un **autoencodeur** ou une simple projection linéaire $\mathbf{U} \in \mathbb{R}^{d' \times d}$ pour ramener \mathbf{f}_i en $\mathbf{g}_i \in \mathbb{R}^{d'}$. Cela **allège** aussi le calcul de $S(i, j)$.

Normalisation

En DSL, il est très courant de **normaliser** chaque embedding à la norme Euclidienne 1 :

$$\mathbf{f}_i \leftarrow \frac{\mathbf{f}_i}{\|\mathbf{f}_i\|_2}$$

afin de faciliter une **similarité cosinus** (également interprétée comme un **produit scalaire**). Un vecteur normalisé \mathbf{f}_i aide à stabiliser les mesures de distance ou les kernels RBF, et simplifie la **dynamique** de la synergie.

Conclusion

Les **embeddings** issus de **CNN** (type ResNet, VGG, etc.) constituent la **pierre angulaire** de la représentation **visuelle** dans un **DSL**. Sur le plan **mathématique**, on dispose d’une **fonction** $\Phi: \mathbf{x} \mapsto \mathbf{f}$ qui, pour chaque image \mathbf{x} , génère un vecteur $\mathbf{f} \in \mathbb{R}^d$. Cette fonction est généralement **pré-entraînée**, permettant de réaliser un **transfert d’apprentissage**. Une fois ce vecteur obtenu, le **SCN** peut calculer la **synergie** entre deux images $\mathcal{E}_i, \mathcal{E}_j$ (c.-à-d. $\mathbf{f}_i, \mathbf{f}_j$) par des mesures comme la similarité cosinus, la distance euclidienne, etc. L’**auto-organisation** s’appuie alors sur ces valeurs pour renforcer ou affaiblir $\omega_{i,j}$. Cette façon de procéder fournit une **base** solide : la représentation **visuelle** est stable, discriminante et souvent généraliste, ce qui constitue un prérequis pour la **synergie** multimodale (notamment pour l’alignement image–texte, image–audio, etc.).

8.3.1.2. Caractéristiques locales (SIFT, SURF) vs. globales (pooling)

Dans le **Deep Synergy Learning (DSL)** appliqué à la modalité **visuelle**, la question du **type** de descripteurs que l’on associe à chaque image se révèle fondamentale. Faut-il privilégier des **caractéristiques locales**, détectées autour de points d’intérêt et codées par des vecteurs comme **SIFT** ou **SURF**, ou plutôt résumer l’image en un **vecteur global** obtenu par un *pooling* (moyen ou max) sur l’ensemble de la carte de caractéristiques ? Cette interrogation n’est pas uniquement pratique : elle influe directement sur la **finesse** de la mesure de **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ et sur la **complexité** du **Synergistic Connection Network (SCN)**.

Les sections précédentes (8.3.1.1) ont montré comment un **CNN** (ResNet, VGG, etc.) pouvait extraire des embeddings globaux. Il reste néanmoins pertinent de discuter la représentation **locale** (points clés) qui peut, dans un cadre **DSL**, coexister ou être combinée à ces embeddings globaux.

A. Caractéristiques locales : SIFT, SURF, etc.

Les descripteurs dits *locaux* se concentrent sur des **points d'intérêt** détectés dans l'image, chacun aboutissant à un vecteur décrivant sa texture et son orientation dans un voisinage restreint. Les méthodes **SIFT** (Scale-Invariant Feature Transform) et **SURF** (Speeded-Up Robust Features) sont parmi les plus connues.

SIFT, par exemple, cherche des **extrema d'échelle** et de position dans l'espace de différences de gaussiennes, ce qui permet de localiser un point clé (x_k, y_k) approximativement invariant aux changements d'échelle et de rotation. Autour de ce point clé, on calcule un **descripteur** $\mathbf{d}_k \in \mathbb{R}^{128}$ en subdivisant un patch en cellules. Sur le plan mathématique, si l'on note $\nabla I(x, y)$ le gradient du niveau de gris au pixel (x, y) et $\theta_{x,y}$ son orientation, alors, pour chaque cellule $c \subset \Omega$ (où Ω est le voisinage du point clé), on accumule un histogramme $\text{Hist}_c(\theta)$:

$$\text{Hist}_c(\theta) = \sum_{(x,y) \in c} w(x, y) \delta(\theta_{x,y}, \theta),$$

où $w(x, y)$ est un poids (généralement une gaussienne centrée au point clé) et θ parcourt une discréétisation en 8 directions principales. L'assemblage de ces histogrammes sur 4×4 cellules fournit un vecteur final de taille 128. SURF emploie un principe similaire mais recourt à des ondelettes de Haar intégrales pour constituer un vecteur plus compact (64 ou 128 dimensions).

Dans un **DSL**, on peut considérer chaque point clé $\mathbf{d}_{i,k}$ de l'image \mathcal{I}_i comme une entité $\mathcal{E}_{i,k}$. La **synergie** $S(\mathcal{I}_i, \mathcal{I}_j)$ entre deux images peut alors s'obtenir via un *matching* local. Une manière de la formaliser est de prendre le max sur tous les couples (k, ℓ) :

$$S(\mathcal{I}_i, \mathcal{I}_j) = \max_{k, \ell} \text{Sim}(\mathbf{d}_{i,k}, \mathbf{d}_{j,\ell}),$$

où Sim peut être un produit scalaire ou un ℓ_2 inversé, en notant que la normalisation de $\mathbf{d}_{i,k}$ peut simplifier la comparaison. Alternativement, on peut additionner ces similarités si l'on veut un score plus global.

L'avantage de ces **descripteurs locaux** est leur robustesse aux transformations partielles (lorsque seule une portion d'image correspond à l'autre) et leur invariance (changement d'échelle, légère rotation). En revanche, le **coût** devient significatif : si chaque image comporte un grand nombre de points clés, la dimension du SCN gonfle et la dynamique $O(n^2)$ (où n est le nombre total d'entités) peut être difficile à gérer, comme l'illustrent les considérations du chapitre 7.

B. Caractéristiques globales : pooling ou représentation agrégée

À l'opposé, on peut vouloir un **vecteur** unique décrivant l'ensemble de l'image. Lorsque l'on exploite un réseau de neurones convolutifs (ResNet, VGG), chaque pixel aboutit, dans les couches profondes, à une **feature map** en (W, H, C) . On peut alors appliquer un *global average pooling* (ou un *global max pooling*) qui, pour chaque canal c :

$$g_c = \frac{1}{W \times H} \sum_{x=1}^W \sum_{y=1}^H f_c(x, y),$$

où $f_c(x, y)$ sont les valeurs de la feature map dans le canal c . On aboutit alors à un vecteur $\mathbf{g} \in \mathbb{R}^C$ qui constitue une **représentation globale** de l'image \mathcal{I} . Cette approche se traduit, dans un SCN, par la création d'une entité \mathcal{E}_i associée au vecteur \mathbf{g}_i . La **similarité** se mesure facilement (cosinus, RBF) et la mise à jour $\omega_{i,j}$ se fait par la formule DSL usuelle.

L'avantage est une **grande simplicité** : un seul vecteur par image, comparaisons rapides, robustesse face à de légères variations. L'inconvénient est que si deux images ne coïncident que localement (un petit objet identique sur des fonds différents), le *pooling* global peut “noyer” la différence ou la similarité.

C. Comparaison locale vs. globale dans un DSL

Le **DSL** peut aisément intégrer ces deux échelles de description, permettant un ajustement par la **dynamique** de la mise à jour ω . On peut ainsi associer, pour chaque image \mathcal{I}_i , un ensemble $\{\mathbf{d}_{i,k}\}_{k=1 \dots K_i}$ de *keypoints* (nœuds “locaux”) et un *embedding* global \mathbf{g}_i (nœud “global”). La **synergie** $S(\mathcal{I}_i, \mathcal{I}_j)$ peut alors se définir comme une combinaison :

$$S(\mathcal{I}_i, \mathcal{I}_j) = \alpha \max_{k,\ell} \text{Sim}(\mathbf{d}_{i,k}, \mathbf{d}_{j,\ell}) + (1 - \alpha) \text{Sim}(\mathbf{g}_i, \mathbf{g}_j),$$

ou toute autre fonction similaire, où $\alpha \in [0,1]$ pèse l’importance relative du *matching* local et du *matching* global. Cette souplesse s’intègre bien dans la logique d’un **SCN** multi-échelle (cf. chap. 6), où l'**auto-organisation** peut pointer, dans certains cas, vers des correspondances précises de patchs, et dans d’autres, vers un accord global d’images entières.

D. Conclusion

La question de l’emploi de **descripteurs locaux** (SIFT, SURF) ou de **descripteurs globaux** (pooling CNN) soulève un compromis entre la **précision** du *matching* local, la **complexité** du SCN, et la **vitesse** de comparaison des images. Les algorithmes locaux offrent une robustesse supérieure lorsqu’une petite zone d’image correspond à une autre (même objet partiellement visible), mais ils accroissent la charge de calcul et la masse d’entités. Les techniques globales (pooling) permettent un vecteur unique par image et simplifient la mise à jour $\omega_{i,j}$, mais peuvent passer à côté de variations locales essentielles. Dans un **DSL**, il est tout à fait envisageable de **combiner** ces deux approches dans la dynamique auto-organisée, de sorte que la **formation** de clusters d’images exploite à la fois l’information locale et le résumé global, et que le réseau lui-même décide, via la mise à jour de la pondération $\omega_{i,j}$, de la **pertinence** ou non de l’un ou l’autre type de descripteur.

8.3.1.3. Avantages de normaliser, dimension typique (256, 512, etc.)

Dans le cadre du **Deep Synergy Learning** (DSL) appliqué aux données visuelles, la représentation de chaque entité par un **vecteur** (*embedding*) soulève deux interrogations. La première concerne le **choix** de la **dimension** de ce vecteur (256, 512, voire 1024). La seconde se rapporte à la **nécessité** de normaliser lesdits vecteurs (c’est-à-dire leur imposer une norme ou un écart-type constant). Ces considérations influencent à la fois la **richesse** de la représentation et la **stabilité** de la dynamique auto-organisée au sein du **Synergistic Connection Network** (SCN).

A. Dimension typique : 256, 512, etc.

Une dimension de quelques centaines, comme 256 ou 512, s’avère souvent un **compromis** pertinent. Il est souhaitable que le vecteur $\mathbf{v}_i \in \mathbb{R}^d$ soit suffisamment grand pour encoder la **variété** des caractéristiques de l’entité visuelle (couleurs, formes, textures, etc.), mais pas au point de rendre le calcul $O(n^2 d)$ inabordable pour un SCN de grande taille. Sur le plan mathématique, si l’on considère deux entités \mathcal{E}_i et \mathcal{E}_j dotées de vecteurs $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^d$, le **produit scalaire** $\mathbf{v}_i \cdot \mathbf{v}_j$ (ou la distance euclidienne $\|\mathbf{v}_i - \mathbf{v}_j\|$) gagne en **précision** lorsqu’on dispose d’un espace de dimension plus grande. On peut, par exemple, définir la **synergie**

$$S(i, j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$$

dans le cas d’une similarité cosinus. Une dimension de l’ordre de 512 peut ainsi mieux capturer les multiples facettes de l’image, en opposition à un vecteur de très faible taille (ex. 16 ou 32) qui risquerait de manquer de **discriminabilité**. À l’inverse, des dimensions trop élevées (1024 ou plus) peuvent conduire à un surcoût de stockage et de calcul, et la “malédiction de la dimension” peut diluer la pertinence des distances.

B. Avantages de normaliser les vecteurs

La seconde préoccupation porte sur la **normalisation** de \mathbf{v}_i . Il est fréquent, dans un **SCN**, de chercher à aligner la norme de chaque embedding à une valeur fixe, par exemple 1, ce qui place \mathbf{v}_i sur la **sphère unité** \mathcal{S}^{d-1} .

Lorsque les vecteurs sont tous normalisés, la similarité cosinus

$$S(i,j) = \mathbf{v}_i \cdot \mathbf{v}_j \quad (\text{puisque } \|\mathbf{v}_i\| = \|\mathbf{v}_j\| = 1),$$

se simplifie et présente une **échelle** de valeurs plus stable. On évite ainsi que certaines entités, possédant un embedding de norme très élevée, dominent le calcul. Dans la dynamique DSL $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$, cette borne entre -1 et +1 facilite le réglage des paramètres η et τ .

L'usage de la sphère unité prévient des problèmes liés à des vecteurs de norme extrême, et la distance ou le produit scalaire entre vecteurs normalisés exhibent un comportement plus régulier. On peut, par ailleurs, décider d'un petit offset pour forcer la positivité (entre 0 et 1), par exemple en appliquant $1 + \mathbf{v}_i \cdot \mathbf{v}_j / 2$, mais la démarche la plus courante reste de s'en tenir à $[-1, +1]$.

On peut réécrire la mise à jour de $\omega_{i,j}$ lorsque l'on s'appuie sur la normalisation. Si $\|\mathbf{v}_i\| = 1$ et $\|\mathbf{v}_j\| = 1$, alors

$$S(i,j) = \mathbf{v}_i \cdot \mathbf{v}_j.$$

La suite $\{\omega_{i,j}(t)\}$ évolue dans un intervalle restreint selon la valeur du produit scalaire, consolidant ainsi la **stabilité** de la dynamique.

C. Conclusion : dimension et normalisation

L'emploi d'une **dimension** de l'ordre de 256 ou 512 pour les **embeddings** visuels constitue un choix courant et équilibré, car il offre une **capacité** représentative suffisante sans surcharger le SCN. De même, la **normalisation** des vecteurs, qui consiste à imposer $\|\mathbf{v}_i\| = 1$, soutient la **cohérence** du calcul de synergie S . Le fait qu'un vecteur ait alors toujours la même norme simplifie l'usage de la similarité cosinus et évite l'influence démesurée de quelques embeddings de forte amplitude. En conséquence, la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

gagne en **prédictibilité**, puisque $S(i,j)$ reste borné et stable. Cette démarche se généralise aisément aux autres canaux (texte, audio), où l'on pratique également une normalisation, facilitant ainsi la fusion multimodale (chap. 8.4, 8.5).

8.3.2. Synergie Visuelle-Visuelle

La **synergie** entre deux **images** (par ex. image_i et image_j) peut se définir via une mesure de **similarité** ou de **distance** au sein d'un espace d'embrayage visuel. Dans la perspective du **DSL** (Deep Synergy Learning), on attribue une pondération $\omega_{i,j}$ reflétant la force de l'association entre image_i et image_j . Au préalable, on doit donc **calculer** la synergie

$$S(\text{image}_i, \text{image}_j) \in \mathbb{R}$$

qui servira de "signal" pour le renforcement ou l'inhibition des liens visuels dans le **SCN**.

8.3.2.1. Calcul de $S(\text{image}_i, \text{image}_j)$: cosinus, distance euclidienne

Les entités **visuelles** dans un **Deep Synergy Learning** (DSL) sont souvent codées par des **vecteurs** extraits d'images, appelés *embeddings*, qu'ils proviennent de descripteurs classiques (SIFT, SURF, HOG) ou de représentations profondes (CNN, ViT, etc.). Une fois ces vecteurs définis, le **Synergistic Connection Network** (SCN) dispose d'un outil pour mesurer la **synergie** entre deux images image_i et image_j via une fonction $S(\text{image}_i, \text{image}_j)$. Le plus souvent, cette synergie repose sur une **distance** ou une **similarité** dans l'espace des embeddings.

Lorsque l'on compare deux images image_i et image_j , on commence par les **représenter** chacune sous forme d'un **vecteur** $\mathbf{v}_i, \mathbf{v}_j \in \mathbb{R}^d$. Selon le scénario, ces vecteurs peuvent provenir de méthodes variées. Certains environnements

privilégiert des *features* “classiques” comme SIFT, SURF, ou HOG, d’autres exploitent un réseau de neurones convolutionnel (ResNet, VGG...) dont on récupère la dernière couche “fully-connected” ou un “global average pooling” (GAP) pour aboutir à un vecteur \mathbf{v} . Il est également possible de combiner plusieurs sources, par exemple des canaux de couleur et de texture. Sur le plan **mathématique**, la conclusion est que chaque image image_i est associée à un vecteur $\mathbf{v}_i \in \mathbb{R}^d$. Pour tout couple (i, j) , on doit donc définir une fonction

$$S(\text{image}_i, \text{image}_j) = f(\mathbf{v}_i, \mathbf{v}_j),$$

et la plus grande liberté est laissée au concepteur pour choisir la forme de cette fonction.

La méthode la plus directe pour quantifier la **similarité** entre deux vecteurs consiste à recourir à la **similarité cosinus** ou à la **distance euclidienne**.

Dans ce cas, on pose

$$S(\text{image}_i, \text{image}_j) = \cos(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}.$$

Si les embeddings sont préalablement normalisés (voir section 8.3.1.3) afin que $\|\mathbf{v}_i\| = 1$, alors la similarité cosinus s’identifie simplement au produit scalaire $\mathbf{v}_i \cdot \mathbf{v}_j$. Un score voisin de 1 signale deux images jugées très proches, tandis qu’un score proche de -1 (pour un cosinus brut) ou 0 (pour un cosinus borné positivement) indique une faible correspondance.

Une autre possibilité est de recourir à la **distance** $d(\mathbf{v}_i, \mathbf{v}_j)$, où la plus courante est la norme ℓ_2 . On peut ensuite définir

$$S(\text{image}_i, \text{image}_j) = \frac{1}{1 + d(\mathbf{v}_i, \mathbf{v}_j)},$$

ou l’on peut préférer un noyau gaussien

$$S(\text{image}_i, \text{image}_j) = \exp(-\alpha \|\mathbf{v}_i - \mathbf{v}_j\|^2),$$

selon l’esprit d’un **RBF-kernel**. L’idée est similaire : plus la distance est petite, plus le score S est grand, ce qui signale une plus grande proximité entre les deux images.

Dans bien des applications, il est bénéfique de **normaliser** chaque \mathbf{v}_i . En appliquant $\mathbf{v}_i \leftarrow \mathbf{v}_i / \|\mathbf{v}_i\|$, on simplifie la formule du cosinus en un simple produit scalaire. Cette pratique unifie l’échelle des embeddings et empêche qu’un vecteur de très forte norme “fausse” les calculs de similarité, ce qui est important lorsque la mise à jour $\omega_{i,j}$ dépend du score S . D’un point de vue purement **mathématique**, cette approche revient à projeter \mathbf{v}_i sur la sphère unité \mathcal{S}^{d-1} , où les distances ou produits scalaires peuvent être plus aisément comparés. Il est également courant de borner la valeur de S à [0,1] en appliquant un offset, par exemple

$$\tilde{S}(i, j) = \frac{1 + \cos(\mathbf{v}_i, \mathbf{v}_j)}{2}.$$

Cette transformation s’intègre parfaitement dans la logique du DSL, où l’on aime manipuler un score positif.

Le **DSL** ne fixe pas de norme unique. Il n’exige qu’une fonction S renvoyant un score localement “cohérent” avec la notion d’entités semblables ou dissemblables. En pratique, l’on retrouve principalement le cosinus ou l’inverse d’une distance euclidienne, mais rien n’empêche de définir une mesure plus sophistiquée (ex. un **kernel** non linéaire ou une **information mutuelle** si l’on traite des distributions). L’essentiel est d’obtenir un **score** qui pointe vers “haute synergie” si deux images sont estimées proches, et “faible synergie” sinon, ce qui oriente la dynamique $\omega_{i,j}$ pour renforcer ou diminuer les liens.

A. Rôle du score : de la similarité à la synergie

Une fois la fonction $S(\text{image}_i, \text{image}_j)$ définie, on l’incorpore dans la **règle** de mise à jour DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\text{image}_i, \text{image}_j) - \tau \omega_{i,j}(t)].$$

Cela signifie qu'un **score** élevé pousse $\omega_{i,j}$ à croître, reflétant la proximité entre les deux images. Au fil du temps, un **SCN** voit se former des **clusters** de liaisons robustes parmi les entités jugées proches, tandis que d'autres liens décroissent. Ce mécanisme **auto-organisé** crée une partition ou une structuration reflétant la **cohérence** visuelle.

B. Détails mathématiques et normalisation

Il existe différentes façons de paramétrier cette fonction. Par exemple, on peut introduire un facteur α dans le RBF-kernel :

$$S(\text{image}_i, \text{image}_j) = \exp(-\alpha \| \mathbf{v}_i - \mathbf{v}_j \|^2).$$

Avec une telle fonction, les images très proches (faible $\| \mathbf{v}_i - \mathbf{v}_j \|$) reçoivent un score proche de 1, tandis que les images distantes ont un score décroissant exponentiellement. Dans le cas d'un cosinus, on peut corriger la fourchette de valeur $[-1, +1]$ pour la replacer dans $[0,1]$. D'un point de vue **numérique**, il est souvent pratique de **normaliser** \mathbf{v}_i pour aboutir à un intervalle fixe, ce qui évite tout réglage supplémentaire dans l'équation DSL.

C. Cas d'images brutes vs. embeddings profonds

Historiquement, on aurait pu comparer deux images en traitant directement les pixels (par exemple, distance L2 entre leur contenu brut), mais cela se heurte rapidement à des problèmes d'invariance (lumière, rotation, etc.), et la dimension devient énorme (un vecteur $128 \times 128 \times 3 = 49152$ pour une image en 128×128 couleur). Dans la pratique contemporaine, on exploite davantage des **embeddings** issus d'un réseau profond (cf. chap. 8.3.1.1) pour aboutir à un vecteur $\mathbf{v}_i \in \mathbb{R}^d$ de taille plus raisonnable (128, 256, 512, etc.) et plus **représentatif** sémantiquement. Cette évolution mathématique facilite le calcul de S et rend la convergence du SCN plus stable.

Conclusion

Le calcul de $S(\text{image}_i, \text{image}_j)$ dans un **DSL** destiné à la vision repose la plupart du temps sur des fonctions de type **cosinus** ou **distance** (euclidienne, Minkowski, RBF). Sur le plan **mathématique**, il suffit de disposer d'un **embedding** $\mathbf{v}_i \in \mathbb{R}^d$ pour chaque image image_i , après quoi on peut choisir une mesure appropriée. Si l'on **normalise** chaque \mathbf{v}_i sur la sphère unité, la similarité cosinus se ramène à un simple produit scalaire. Ce score introduit dans la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\text{image}_i, \text{image}_j) - \tau \omega_{i,j}(t)]$$

orienté la **dynamique auto-organisée** du SCN pour renforcer les liens entre images **proches** et atténuer les autres, générant ainsi des **clusters** visuels auto-émergents.

8.3.2.2. Approche par hashage (LSH) pour accélérer la similarité d'images

Dans les applications **multimodales** du **Deep Synergy Learning** (DSL) où l'on doit traiter un nombre conséquent d'entités visuelles (pouvant se chiffrer en milliers ou en millions), la comparaison exhaustive de toutes les paires $(\mathcal{E}_i, \mathcal{E}_j)$ mène à une complexité en $O(n^2)$. Cette explosion pose un problème majeur lorsqu'on désire construire un **Synergistic Connection Network** (SCN) sur un large ensemble d'images. Pour remédier à cela, on recourt souvent à des **techniques de hashage** localement sensible, dites **LSH** (*Locality-Sensitive Hashing*), qui visent à **rapprocher** les entités potentiellement semblables dans des "buckets" ou codes communs, et à s'abstenir de comparer explicitement toutes les paires.

A. Principe général du LSH

Le **LSH** se fonde sur l'idée qu'il existe des **fonctions** de hachage, notées $h: \mathbb{R}^d \rightarrow \mathcal{C}$, ayant la propriété suivante : si $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ sont proches (par exemple au sens d'une norme $\| \mathbf{x}_i - \mathbf{x}_j \|$ ou d'un angle cosinus), la probabilité que $h(\mathbf{x}_i) = h(\mathbf{x}_j)$ soit élevée. L'objectif mathématique est :

$$\|\mathbf{x}_i - \mathbf{x}_j\| \text{ petite} \Rightarrow \text{Prob}[h(\mathbf{x}_i) = h(\mathbf{x}_j)] \approx 1,$$

et inversement, si $\|\mathbf{x}_i - \mathbf{x}_j\|$ est grande, la collision de leurs codes devient improbable.

Dans la pratique, on ne se contente pas d'une unique fonction de hachage, mais on en définit plusieurs, notées h_1, h_2, \dots, h_L . Les résultats de ces fonctions sont concaténés en un **code** ou un **bucket**. Deux entités $\mathbf{x}_i, \mathbf{x}_j$ se retrouvent alors rangées dans le même bucket si $(h_1(\mathbf{x}_i), \dots, h_L(\mathbf{x}_i)) = (h_1(\mathbf{x}_j), \dots, h_L(\mathbf{x}_j))$. L'effet combiné est de **discriminer** plus finement les entités proches de celles qui ne le sont pas.

Lorsque l'on souhaite calculer la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ dans un **SCN**, on peut ne se concentrer que sur les paires dont les codes LSH sont identiques ou suffisamment proches. On passe ainsi d'une complexité naïve $O(n^2)$ à un processus plus ciblé, qui compare bien moins de paires.

B. Exemples de fonctions LSH pour images

Les algorithmes de LSH diffèrent selon la **distance** ou la **mesure** qu'on veut préserver.

Lorsqu'on emploie la **similarité cosinus**, on peut définir des hyperplans aléatoires $\{\mathbf{r}_k\}_{k=1}^L \subset \mathbb{R}^d$. Pour un vecteur \mathbf{x} , on prend un bit $b_k = \text{sgn}(\mathbf{r}_k \cdot \mathbf{x})$. Les L bits forment alors un code binaire de longueur L . Deux vecteurs $\mathbf{x}_i, \mathbf{x}_j$ proches en angle ont de grandes chances de donner la même suite (b_1, \dots, b_L) . On retrouve ainsi la propriété de hachage localement sensible.

Dans les applications de “hachage perceptuel” d’images, on peut redimensionner et convertir chaque image (ou son embedding) via une transformée en cosinus discrète (DCT), puis binariser les coefficients les plus bas ou la médiane. Cela donne un code binaire (ex. 64 bits), où deux images similaires (au sens perceptuel) se retrouveront souvent avec le même code.

Pour représenter une image comme un ensemble (ou un multi-ensemble) de *features* (par exemple, un sac de mots visuels), on peut appliquer des permutations aléatoires et prendre le plus petit index, un concept appelé **minwise hashing**. Les ensembles présentant une grande intersection partagent plus fréquemment le même hash.

C. Intégration dans un DSL

Dans un **SCN** gérant une foule d’images $\{\mathcal{E}_i\}_{i=1\dots n}$, on souhaite limiter le calcul explicite de la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ à des paires “pertinentes”. On peut alors indexer chaque embedding $\mathbf{x}_i \in \mathbb{R}^d$ dans une structure LSH. Lorsqu'on insère la nouvelle entité \mathbf{x}_i , on détermine son code LSH et on ne compare \mathbf{x}_i qu’aux entités déjà présentes dans le même bucket (ou un voisinage restreint en code). De cette façon, on se concentre sur des **candidats** de haute similarité potentielle, passant la complexité d'un $O(n^2)$ à quelque chose plus proche de $O(n \cdot \text{avgBucketSize})$, ce qui peut s'avérer beaucoup plus tractable.

Dans la **dynamique** du **DSL**, cela signifie qu’au lieu de parcourir exhaustivement tous les index j pour une entité i et de mettre à jour $\omega_{i,j}(t+1)$, on parcourt seulement les paires (i, j) dont les codes LSH s’avèrent proches (potentiellement identiques). On résout ainsi la mise à jour $\omega_{i,j}(t+1) = \dots$ uniquement sur ce sous-ensemble réduit, ce qui accélère la formation d'un **SCN** de grande échelle.

D. Aspects mathématiques et erreurs LSH

Le LSH procure un **contrôle** probabiliste : deux entités “vraiment proches” (au sens de $\|\mathbf{x}_i - \mathbf{x}_j\|$ ou $\mathbf{x}_i \cdot \mathbf{x}_j$) ont une haute probabilité d’être placées dans le même code ; deux entités très distantes ne le sont qu’avec une probabilité faible. Il y a cependant des “faux négatifs” (des entités proches mais qui tombent en codes différents) et des “faux positifs” (entités pas si proches, mais se retrouvant dans le même code). Sur un **DSL**, cela peut induire respectivement un manque dans la mise à jour de $\omega_{i,j}$ pour certaines paires dignes d’intérêt, ou une comparaison superflue de quelques paires non pertinentes. Tant que les probabilités de ces cas demeurent faibles, le gain en complexité (moins de paires testées) l’emporte.

Conclusion

Le **hashage localement sensible** (*Locality-Sensitive Hashing*) est un instrument essentiel pour **accélérer** la recherche de similarités entre un grand nombre d'images dans le cadre d'un **Deep Synergy Learning** multimodal. L'idée est de substituer à un parcours $O(n^2)$ un index LSH qui ne compare que les embeddings partageant un même code ou un code voisin, réduisant ainsi le nombre de paires $\{(i, j)\}$ sur lesquelles on évalue la fonction S . Cette démarche s'accorde aux principes du **SCN** : la mise à jour $\omega_{i,j}(t+1)$ n'a plus besoin d'être appliquée à toutes les paires, mais seulement à celles probablement proches dans l'espace d'embeddings, car "ciblées" par un bucket LSH. De cette manière, on conserve un bon niveau de **fiableté** pour retrouver les voisins, tout en surmontant la contrainte majeure d'une complexité $O(n^2)$ potentiellement ingérable pour un large flux d'images.

8.3.2.3. Cas d'un flux vidéo : segmentation en frames ou analyse spatio-temporelle ?

La **vidéo** constitue un exemple clé de la dimension **multimodale** examinée dans la section 8.3.2, où la notion de **temps** s'ajoute à l'espace purement visuel de chaque image. Lorsqu'on souhaite construire un **Synergistic Connection Network** (SCN) pour un **flux vidéo**, on peut adopter un modèle où l'on segmente la séquence en *frames* indépendantes (comme autant de "photos" séparées) ou, au contraire, essayer de capturer la **dynamique spatio-temporelle** qui relie ces images consécutives. Le **Deep Synergy Learning** (DSL) reste flexible quant aux choix de représentation, mais la complexité mathématique et la pertinence des **clusters** ou **entités** créés peuvent différer considérablement.

A. Segmentation en frames simples

Un choix classique et simplifié consiste à traiter la vidéo "frame par frame". Dans ce cas, chaque image \mathcal{E}_t correspond à la vue au temps $t\Delta t$. Sur le plan **mathématique**, on dispose d'un vecteur $\mathbf{v}_t \in \mathbb{R}^d$ pour chaque frame, obtenu typiquement par un **CNN** (ResNet, VGG...) ou un autre extracteur visuel. Le **DSL** manipule alors un ensemble $\{\mathbf{v}_1, \dots, \mathbf{v}_n\}$, où n est le nombre de frames retenues. La **synergie** $S(\mathcal{E}_t, \mathcal{E}_{t'})$ peut être définie, par exemple, comme une **similarité cosinus** ou l'inverse d'une **distance** $\|\mathbf{v}_t - \mathbf{v}_{t'}\|$.

Le principal intérêt de cette segmentation tient à la facilité de mise en œuvre : on applique les techniques de traitement d'images statiques (extraction d'embeddings) à chaque frame de la vidéo, puis on alimente le **SCN**. Le **DSL** ne voit alors que des images, indépendantes. Cela peut convenir si l'objectif se limite à un échantillonnage léger dans le flux (e.g. prendre une image toutes les 10 frames) pour avoir un aperçu global. Sur le plan mathématique, la charge de calcul dépend du nombre total de frames ; chaque itération du DSL s'opère en $O(n^2)$ si on compare tous les couples (t, t') . Cette multiplication du nombre de frames peut évidemment se heurter à des problèmes de complexité (chap. 7), conduisant à recourir à des heuristiques pour limiter la comparaison exhaustive.

Cette approche *frame par frame* néglige la **continuité temporelle** : deux frames successifs partagent généralement des motifs quasi identiques, ce qui peut créer des **clusters** trop dominants entre frames proches, sans qu'on exploite explicitement l'information de mouvement. Elle n'est donc pas adaptée si on désire repérer des "événements dynamiques" (gestes, actions) se déroulant sur plusieurs frames.

B. Analyse spatio-temporelle

Une seconde option plus riche modélise la vidéo de manière spatio-temporelle. Au lieu de ne considérer que l'espace (x, y) pour un instant t , on tient compte également de la dimension temps. On peut alors découper la vidéo en "volumes" (blocs (x, y, t)) ou **patches 3D**. Sur le plan **mathématique**, chaque entité \mathcal{E}_α correspond à un sous-cube de la séquence, par exemple 16 frames contiguës sur 112×112 pixels, pour lequel on calcule un **embedding 3D** (réseau CNN 3D, S3D, I3D, etc.) ou un **modèle** spatio-temporel plus complexe.

La **synergie** $S(\mathcal{E}_\alpha, \mathcal{E}_\beta)$ devient alors une mesure de similarité entre deux volumes (ex. un cosinus sur leurs vecteurs $\mathbf{u}_\alpha, \mathbf{u}_\beta \in \mathbb{R}^d$). Ceci permet de capter le **mouvement** et la **dynamique** dans la séquence : deux actions semblables (saut, coup de raquette, etc.) partagent un embedding spatio-temporel plus élevé. L'approche s'avère très puissante pour reconnaître des **événements** (actions complexes) qui se déroulent sur plusieurs frames. Elle accroît toutefois la **dimension** du descripteur (un vecteur 3D plus volumineux) et le nombre total d'entités (tous les blocs potentiels α), exacerbant la **complexité** $O(n^2)$ dans la mise à jour DSL.

C. Conséquences mathématiques

Le passage à un espace spatio-temporel modifie la définition de S . Par exemple, on peut écrire

$$S(\mathcal{E}_\alpha, \mathcal{E}_\beta) = \lambda_1 \text{sim}(\mathbf{v}_\alpha^{(\text{apparence})}, \mathbf{v}_\beta^{(\text{apparence})}) + \lambda_2 \text{sim}(\mathbf{v}_\alpha^{(\text{mouvement})}, \mathbf{v}_\beta^{(\text{mouvement})}),$$

afin de coupler l'aspect *apparence* (les frames) et l'aspect *dynamique* (le flot optique, l'embedding 3D). Le **DSL** manipule ensuite la règle

$$\omega_{\alpha,\beta}(t+1) = \omega_{\alpha,\beta}(t) + \eta [S(\mathcal{E}_\alpha, \mathcal{E}_\beta) - \tau \omega_{\alpha,\beta}(t)],$$

ce qui construit un **SCN** reflétant non seulement la ressemblance spatiale, mais également la **cohérence temporelle**. Les coûts croissent, car la quantité d'entités (blocs spatio-temporels) peut être très importante, invitant à la prudence (éventuelles méthodes heuristiques ou approximations, cf. chap. 7).

D. Choix entre segmentation en frames ou spatio-temporel

Le choix dépend de la finalité. Pour de simples tâches de repérage ou de détection de scène globale, la segmentation *frame par frame* peut suffire, en ignorant l'information de mouvement, plus rapide à mettre en place. En revanche, pour capturer des “actions” s'étalant sur plusieurs instants, la **dimension temporelle** doit être intégrée dans la représentation, ce qui implique un embedding spatio-temporel ou au moins une agrégation de frames successives. Une approche hybride est également possible : on segmente grossièrement en frames (ou micro-blocs), et l'on ne lance une analyse plus profonde spatio-temporellement que sur les segments jugés intéressants. Sur le plan **mathématique**, cela se traduit par la coexistence, dans le SCN, d'entités “frame” et d'entités “blocs 3D”, entre lesquelles la **synergie** se définit de manière cohérente.

Conclusion

Pour traiter un flux vidéo dans un **DSL**, deux grandes stratégies se distinguent. La première repose sur la **segmentation en frames**, où chaque image est traitée indépendamment. La seconde adopte une approche d'**analyse spatio-temporelle**, dans laquelle chaque entité correspond à un volume 3D ou à un patch temporel. Le premier choix est plus simple, mais néglige la continuité temporelle ; le second capture la **dynamique** d'action mais accroît la complexité. Un **SCN** peut accueillir l'un ou l'autre (ou un mélange) selon les besoins du système. D'un point de vue **mathématique**, cela se répercute sur la définition de la **synergie** $S(\cdot, \cdot)$ (cosinus, distance, etc.) et sur la taille n d'entités à gérer dans la règle de mise à jour $\omega_{i,j}(t+1)$. Chaque concepteur doit jauger l'ampleur du flux vidéo, la granularité temporelle requise et le **coût** de calcul, pour décider d'une représentation (frames statiques ou volumes 3D) adaptée à l'objectif visé (clustering de scènes, détection d'événements, reconnaissance d'actions, etc.).

8.3.3. Liens avec d'Autres Modalités

Lorsque l'on parle de **DSL Multimodal**, on ne se limite pas à un seul couplage (vision + audio, par exemple) ; au contraire, il est fréquent d'**enchaîner** ou de **composer** plusieurs modalités. La **synergie** peut alors s'étendre entre la **vision**, le **texte**, l'**audio**, ou d'autres flux (capteurs, données symboliques), chacun interagissant pour renforcer la **compréhension** globale. Dans cette section 8.3.3, nous mettons l'accent sur la **visée** du DSL à l'heure de **concilier** (ou de relier) la vision avec d'autres flux, en particulier le **texte** et l'**audio**, afin de clarifier comment le **SCN** (Synergistic Connection Network) fusionne des informations potentiellement très différentes (images, séquences textuelles, signaux sonores).

8.3.3.1. Vision–Texte : ex. l'étiquette textuelle d'une image (caption)

L'association entre **vision** et **texte** constitue l'une des illustrations les plus abouties de la **multimodalité** évoquée dans les sections précédentes. Dans un **Deep Synergy Learning** (DSL) appliqué à un **Synergistic Connection Network** (SCN), il est fréquent d'introduire deux **ensembles** d'entités, l'un relatif aux **descripteurs** (embeddings) issus d'**images**, l'autre associé aux **segments textuels** (mots, étiquettes, phrases). Les liaisons $\omega_{i,j}$ entre \mathcal{V}_i (vision) et \mathcal{T}_j

(texte) se renforcent dès que la **synergie** $S(\mathcal{V}_i, \mathcal{T}_j)$ révèle une adéquation marquée. Ce mécanisme permet, par exemple, de faire émerger un **couplage** entre une image et son label (“cat”), ou encore une légende plus détaillée.

A. Définition mathématique de la synergie Vision–Texte

Pour relier une **image** (ou un patch) \mathcal{V}_i à un **extrait textuel** \mathcal{T}_j , il s'avère utile de disposer de deux **embeddings** dans un espace commun ou partiellement commun. Dans des approches récentes (type CLIP), on apprend des projections $\phi_{\text{img}}(\mathbf{v}_i)$ et $\phi_{\text{txt}}(\mathbf{t}_j)$ qui convergent vers un espace latent \mathbb{R}^d . Le score de **similarité** peut alors se définir comme :

$$S(\mathcal{V}_i, \mathcal{T}_j) = \cos(\phi_{\text{img}}(\mathbf{v}_i), \phi_{\text{txt}}(\mathbf{t}_j)) = \frac{\phi_{\text{img}}(\mathbf{v}_i) \cdot \phi_{\text{txt}}(\mathbf{t}_j)}{\|\phi_{\text{img}}(\mathbf{v}_i)\| \|\phi_{\text{txt}}(\mathbf{t}_j)\|}.$$

Plus ce cosinus est élevé (proche de 1), plus le modèle estime que le texte correspond visuellement (et sémantiquement) à l'image.

Le **Synergistic Connection Network** peut aussi manipuler plusieurs *patchs* pour une image et plusieurs *mots* pour un texte. Chaque entité visuelle $\mathcal{V}_{i,k}$ décrit un patch ou un objet détecté, chaque entité textuelle $\mathcal{T}_{j,\ell}$ décrit un token ou un segment. La **synergie** S entre un patch $\mathcal{V}_{i,k}$ et un mot $\mathcal{T}_{j,\ell}$ peut se formaliser via un cosinus, un RBF-kernel, ou une fonction quelconque de distance. On obtient ensuite un **cluster** combinant plusieurs patchs et mots si la dynamique auto-organisée du DSL accroît fortement les liaisons $\omega_{(i,k),(j,\ell)}$.

B. Processus de mise à jour ω et formation de clusters Vision–Texte

La règle fondamentale reste :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{V}_i, \mathcal{T}_j) - \tau \omega_{i,j}(t)].$$

Si la synergie S entre \mathcal{V}_i et \mathcal{T}_j est élevée (i.e. forte similarité dans l'espace latent ou cosinus élevé), $\omega_{i,j}(t+1)$ augmente, traduisant une correspondance solide (par exemple, l'étiquette textuelle “dog” apparaît juste pour l'image représentant un chien). Lorsque l'étiquette ne convient pas (score ≈ 0), la pondération $\omega_{i,j}$ diminue peu à peu.

Après plusieurs itérations, on constate que certaines entités visuelles $\{\mathcal{V}_i\}$ et certains descripteurs textuels $\{\mathcal{T}_j\}$ présentent des $\omega_{i,j}$ fortement établis, signant l'**appariement** entre l'image et son label ou sa légende. On observe alors des **clusters** multimodaux, où un ensemble d'images (ou de patchs) se raccorde à un mot, ou une phrase plus ou moins longue. Cet effet de **renforcement** est la version auto-organisée d'une annotation, ou **captioning**, guidée par la logique du DSL.

C. Applications et bénéfices

Ce mécanisme de **synergie** Vision–Texte s'apparente à la construction d'une légende : l'image \mathcal{V}_i “attire” les mots \mathcal{T}_j qui la décrivent le mieux. Sur le plan mathématique, le réseau renforce $\omega_{i,j}$ pour les paires cosinus ou distance favorables, et le cluster final associe un sous-ensemble de vocables à l'image. Cela aboutit à une forme de captioning ou d'**annotation** non supervisée, même si l'on peut y ajouter plus de contraintes ou d'inhibition (chap. 7) pour modérer le sur-appariement.

Si le SCN relie chaque image à des mots, la requête textuelle “cat” se traduit automatiquement par la recherche des entités \mathcal{T}_j associées au mot “cat” et des images qui leur sont connectées dans le réseau. Les **pondérations** ω ou la synergie S jouent un rôle d'index sémantique : on retrouve quasi instantanément les images dont la légende ou le label est “cat”.

Sur le plan théorique, l'**auto-organisation** opère si la dimension des embeddings (image, texte) est suffisante et si la fonction S reflète la proximité sémantique. La descente implicite du DSL accroît $\omega_{i,j}$ pour les paires qui se contretiennent (mots et images vraiment voisins dans l'espace latent), conduisant à une **cohérence** à l'échelle du réseau.

Conclusion

Le **lien** entre la **vision** et le **texte** dans un cadre **DSL** (Deep Synergy Learning) offre un paradigme **auto-organisé** pour associer une image (ou un patch) à un mot (ou une légende). Sur le plan **mathématique**, on calcule un **score** $S(\mathcal{V}_i, \mathcal{T}_j)$ (typiquement un cosinus ou un kernel) qui indique la proximité sémantique. La mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ traduit alors le **renforcement** progressif des couples (image, label) qui se correspondent, menant à des **clusters** Vision–Texte auto-émergents. Cette propriété facilite l'**annotation** automatique (si un mot s'avère proche d'une image, on l'associe à cette image), la **recherche** cross-modale (un mot permet de localiser les images avec lesquelles il s'est lié) et plus globalement la constitution d'une **architecture** multimodale unifiant images et segments linguistiques dans un **SCN**. Cela s'insère de façon naturelle dans la suite (cf. 8.3.3.2, 8.3.3.3) consacrée aux autres combinaisons multimodales (e.g. texte–audio).

8.3.3.2. Vision–Audio : correspondance entre la scène visuelle et l'environnement sonore (ex. oiseaux, etc.)

Lorsque l'on désire fusionner la **vision** et l'**audio** au sein d'un **Deep Synergy Learning** (DSL) multimodal, il est fréquent de se retrouver face à des situations où la **scène visuelle** (image, vidéo) recèle des informations sur la présence d'entités (ex. oiseaux, véhicules, objets divers), tandis que l'**audio** apporte un **contexte** supplémentaire (chants, bruits ambients). Le **Synergistic Connection Network** (SCN) peut alors établir des **liens** entre des entités visuelles et sonores si la **synergie** mesurée $S(\mathcal{E}_{\text{vis}}, \mathcal{E}_{\text{aud}})$ est suffisamment élevée. Cet exemple Vision–Audio illustre particulièrement la puissance de l'**auto-organisation** dans un cadre multimodal : la dynamique ω tend à clusteriser ensemble les éléments **visuels** et **auditifs** qui coïncident ou se corroborent, aboutissant à une représentation unifiée de la scène et de son environnement sonore.

A. Entités visuelles vs. entités sonores

D'un côté, on dispose de **descripteurs visuels** $\mathbf{v}_i \in \mathbb{R}^d$, que l'on peut considérer soit comme un **embedding** unique de l'ensemble de l'image (ou d'une frame vidéo), soit comme un ensemble de **patches** ou de bounding boxes détectées (plusieurs $\mathbf{v}_{i,k}$). D'un autre côté, l'**audio** est quant à lui découpé en **trames** ou en **segments**, par exemple via des MFCC ou un spectrogramme. Chaque segment audio $\mathbf{a}_j \in \mathbb{R}^m$ correspond à un court intervalle temporel (quelques millisecondes à une seconde), permettant une reconnaissance potentielle d'événements sonores (ex. chant d'oiseau, bruissement de feuilles, klaxon, etc.).

Dans un **SCN** multimodal, on introduit alors des **nœuds** $\mathcal{E}_{\text{vis},i}$ pour le canal visuel et $\mathcal{E}_{\text{aud},j}$ pour le canal audio. Les pondérations $\omega_{(\text{vis},i),(\text{aud},j)}$ traduisent la **compatibilité** ou la **corrélation** entre ces entités. Le **DSL** se charge de mettre à jour ces pondérations par la formule :

$$\omega_{(\text{vis},i),(\text{aud},j)}(t+1) = \omega_{(\text{vis},i),(\text{aud},j)}(t) + \eta [S(\mathcal{E}_{\text{vis},i}, \mathcal{E}_{\text{aud},j}) - \tau \omega_{(\text{vis},i),(\text{aud},j)}(t)].$$

B. Calcul de la synergie $S(\mathcal{E}_{\text{vis},i}, \mathcal{E}_{\text{aud},j})$

La **synergie** $S(\mathbf{v}_i, \mathbf{a}_j)$ peut se fonder sur diverses propriétés cross-modales, selon le but recherché. L'une des approches les plus directes repose sur la **coïncidence temporelle** : si la trame audio \mathbf{a}_j correspond à l'instant t_j et le segment visuel \mathbf{v}_i à l'instant t_i (ou à un intervalle $[t_i, t_i + \Delta]$), alors on estime la synergie plus grande lorsque $|t_i - t_j|$ est petit. Pour refléter cette dépendance, on peut définir :

$$S(\mathbf{v}_i, \mathbf{a}_j) = \rho(\Delta t_{i,j}) \cdot D(\mathbf{v}_i, \mathbf{a}_j),$$

où $\rho(\Delta t_{i,j})$ est une fonction (par exemple gaussienne ou tophat) qui vaut 1 si $\Delta t_{i,j} < \Delta_{\max}$ et décroît à mesure que l'écart temporel croît. Le terme $D(\mathbf{v}_i, \mathbf{a}_j)$ représente une **compatibilité** de contenu : par exemple, un modèle d'identification d'objets visuels (oiseau dans \mathbf{v}_i) et un classifieur audio (chant d'oiseau dans \mathbf{a}_j) peuvent produire une probabilité conjointe de correspondance. Un schéma plus élaboré consiste à projeter \mathbf{v}_i et \mathbf{a}_j dans un **espace latent** commun (type CLIP multimodal) et d'utiliser un cosinus ou un RBF-kernel entre $\phi(\mathbf{v}_i)$ et $\phi(\mathbf{a}_j)$. L'intégration de la variable temporelle (décalage) peut alors se formaliser par une pénalisation $\exp(-\alpha |t_i - t_j|)$.

C. Correspondance scène–environnement : exemple oiseaux

Un exemple emblématique est celui d'une **scène** filmant la nature (un bosquet, un marécage) et d'une **piste audio** où l'on entend des chants d'oiseaux. Sur le plan mathématique, chaque entité visuelle \mathbf{v}_i décrit un patch où l'on détecte un oiseau, tandis que chaque segment audio \mathbf{a}_j correspond à un intervalle temporel où un algorithme de reconnaissance identifie un chant d'oiseau. La **synergie** $S(\mathbf{v}_i, \mathbf{a}_j)$ est alors élevée si l'on repère, d'une part, la proximité temporelle (le chant survient exactement quand l'oiseau est visible) et, d'autre part, la proximité sémantique (oiseaux de la même espèce, par exemple).

Dans le **SCN**, la pondération $\omega_{(vis,i),(aud,j)}$ se renforce donc lorsqu'un chant d'oiseau coïncide dans le temps (et la sémantique) avec la vision d'un oiseau. Au fur et à mesure que la mise à jour :

$$\omega_{(vis,i),(aud,j)}(t+1) = \omega_{(vis,i),(aud,j)}(t) + \eta [S(\mathbf{v}_i, \mathbf{a}_j) - \tau \omega_{(vis,i),(aud,j)}(t)]$$

accumule ces signaux, on voit **émerger** un cluster inter-modal reliant "patchs d'oiseaux" et "segments audio de chants", illustrant la **cohérence** du flux multimodal : la vue et l'ouïe se **soutiennent** mutuellement.

D. Structure multimodale et gains

Le SCN ainsi formé propose plusieurs avantages. D'abord, il peut **découvrir** des correspondances entre des entités visuelles mal identifiées et des signaux sonores explicites (le chant d'oiseau explicite le fait que le patch visuel flou est bien un oiseau). Ensuite, il rend possible des **clusters** plus riches que dans un mode unimodal : on obtient des **macro-nœuds** associant un ensemble de segments audio et un ensemble de patchs visuels. Sur le plan **mathématique**, la dynamique en $O(n^2)$ peut toutefois contraindre la taille du dataset, sauf à employer des heuristiques (LSH, k-NN, voir chap. 7). Mais la **philosophie** du DSL reste la même : on auto-organise un réseau où se renforcent les liaisons trans-modales en cas de coïncidence.

Conclusion

La **vision** et l'**audio** offrent un exemple emblématique de **synergie** inter-modale dans le cadre d'un **DSL**. En couplant des entités visuelles \mathbf{v}_i (images, patchs) et des entités auditives \mathbf{a}_j (segments sonores) via une fonction $S(\mathbf{v}_i, \mathbf{a}_j)$ tenant compte de la cohérence temporelle et sémantique, la mise à jour $\omega_{(vis,i),(aud,j)}(t+1) = \dots$ favorise la formation de **clusters** audio-vision significatifs. Ainsi, la **dynamique** auto-organisée de ce **Synergistic Connection Network** permet d'identifier, par exemple, le lien entre un oiseau visible et le chant entendu, ou tout autre événement couplé (véhicule passant, bruit de moteur, etc.). Ce scénario incarne la **logique** multimodale du DSL : la **fusion** des canaux (vision, audio) naît d'une **dynamique locale** des pondérations, aboutissant à des **macro-nœuds** cross-modaux cohérents et robustes.

8.3.3.3. Exemples concrets (images de chats accompagnées de miaulements)

Les exemples les plus parlants pour comprendre la **synergie** multimodale d'un **Deep Synergy Learning** (DSL) se manifestent souvent par de petites scènes associant **visuel** et **audio** de manière évidente, comme le cas d'un **chat** et de son **miaulement**. Lorsque l'on combine ces données au sein d'un **Synergistic Connection Network** (SCN), on obtient une représentation qui auto-organise non seulement les entités **visuelles** relatives au chat, mais aussi les entités **auditives** correspondant à des miaulements, et fait émerger leurs correspondances par la dynamique de la mise à jour ω .

A. Contexte et mise en place

Supposons qu'il existe un ensemble d'**images** ou de **frames** montrant différents chats. Chaque image \mathcal{E}_i est associée à un vecteur d'**embedding** $\mathbf{x}_i \in \mathbb{R}^d$. De surcroît, on dispose d'**extraits audio** où l'on entend divers **miaulements**. Chacun de ces extraits \mathcal{E}_j est décrit par un vecteur $\mathbf{y}_j \in \mathbb{R}^{d'}$, par exemple à partir d'une représentation MFCC ou d'un embedding appris. Dans un **SCN** multimodal, on introduit donc deux familles d'entités : $\{\mathcal{E}_i^{(vis)}\}$ pour les images de chats, $\{\mathcal{E}_j^{(aud)}\}$ pour les miaulements.

L'objectif est d'étudier la **synergie** $S(\mathbf{x}_i, \mathbf{y}_j)$ entre un embedding d'image \mathbf{x}_i et un embedding audio \mathbf{y}_j . Une **haute** synergie peut signifier qu'un miaulement particulier correspond à l'image d'un certain chat, ou qu'ils appartiennent au même contexte (ex. le même chat filmé et entendu simultanément).

B. Construction de la synergie multimodale

Le calcul de $S(\mathbf{x}_i, \mathbf{y}_j)$ dépend de la modalité visée. On peut se limiter à la “coïncidence” temporelle et au fait qu'une photo d'un chat noir correspond à un enregistrement audio réputé être un miaulement de chat. Sur un plan **mathématique**, un score simple se définit en repérant, par exemple, un classifieur audio qui détecte “chat” et un classifieur vision qui détecte “chat”, et en pondérant leur accord :

$$S(\mathbf{x}_i, \mathbf{y}_j) = \mathbf{1}(\text{label}_{\text{vis}}(\mathbf{x}_i) = "cat") \cdot \mathbf{1}(\text{label}_{\text{aud}}(\mathbf{y}_j) = "cat meow").$$

Une formule plus avancée peut projeter \mathbf{x}_i et \mathbf{y}_j dans un **espace latent** Φ (typiquement appris pour associer images et sons), puis utiliser une distance ou une similarité :

$$S(\mathbf{x}_i, \mathbf{y}_j) = \exp(-\alpha \parallel \Phi(\mathbf{x}_i) - \Phi(\mathbf{y}_j) \parallel^2).$$

Cette fonction de synergie, insérée dans la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathbf{x}_i, \mathbf{y}_j) - \tau \omega_{i,j}(t)],$$

renforce les liaisons $\omega_{i,j}$ entre l'image i et le son j quand ils s'avèrent pertinents.

C. Synergie émergente : clusters “image + miaulement”

Dans un **SCN** multimodal, la dynamique $\omega_{(vis,i),(aud,j)}(t)$ finit par **souligner** les correspondances stables entre certains groupes de photos de chats et certains types de miaulements. Ainsi, un certain “chat tigré” peut se coupler à un “miaulement strident”, ou un “chaton” se voir relier à un “miaulement aigu”. On voit donc émerger des **clusters** mixtes contenant à la fois des entités visuelles et des entités audio.

Supposons un petit dataset de 10 images de chat (différentes races, postures) et 5 enregistrements de miaulements. Au départ, $\omega_{i,j} \approx 0$. Si un classifieur ou un module de similarité multi-modal signale que l'image \mathbf{x}_3 (chat siamois) coïncide souvent en contexte avec le miaulement \mathbf{y}_2 (miaulement “siamois”), alors $\omega_{3,2}(t)$ va croître. D'autres associations resteront faibles. Sur le plan **mathématique**, ce renforcement dérive de la valeur $S(\mathbf{x}_3, \mathbf{y}_2) \approx 0.8$ par exemple, alors que $S(\mathbf{x}_1, \mathbf{y}_2)$ serait autour de 0.1. Après plusieurs itérations, un **cluster** se solidifie autour de {images siamoises} \cup {miaulement siamois}.

D. Observations et conclusions

Ces scénarios “images de chats + miaulements” démontrent à petite échelle la **puissance** du DSL multimodal. La **dynamique** auto-organisée :

4. Crée des **liens** plus forts (pondérations ω) entre des images et des sons compatibles,
5. Constitue des **clusters** multicanaux, par exemple un macro-nœud contenant un même chat vu sous différents angles et le même miaulement sur plusieurs enregistrements,
6. Permet de **trouver** ou de **valider** (sans supervision explicite) quelles entités audio s'alignent le plus avec quelles entités visuelles.

Cette **approche** conserve toute la flexibilité du DSL (chap. 2.2) : on peut y intégrer d'autres attributs (couleurs, textures, types de miaulements), ou différents timescales (frames vidéo vs. segments audio temporels), et laisser la mise à jour $\omega_{i,j}(t+1) = \dots$ converger vers un réseau cohérent associant chaque chat à son miaulement. Ce petit exemple, triviallement “chat + meow”, n'est qu'un avant-goût de la **capacité** du **SCN** à faire émerger des correspondances **cross-modales** plus complexes dans des applications réelles.

8.4. Modélisation des Entités Linguistiques

Lorsque l'on traite des données textuelles au sein d'un **SCN** (Synergistic Connection Network), la question de la **représentation** des mots, segments, phrases ou documents se révèle cruciale : la **synergie** $S(i,j)$ entre deux entités linguistiques dépend fortement de la manière dont on les encode. La section 8.4 aborde donc la **modélisation** des informations **linguistiques** (mots, phrases, documents) sous forme sub-symbolique (embeddings vectoriels) ou symbolique (ensembles de mots, règles) et explicite comment le **DSL** (Deep Synergy Learning) s'en sert pour calculer la synergie entre entités textuelles, ou entre le texte et d'autres modalités (image, audio).

8.4.1. Représentations Textuelles

Les entités linguistiques d'un **SCN** peuvent être des **tokens** (mots ou sous-mots), des **phrases**, voire des **documents** complets. Au niveau le plus bas (micro), on manipule souvent des **embeddings** vectoriels ; au niveau plus global (macro), on peut agréger plusieurs tokens/phrases en un super-nœud correspondant à un thème ou un concept (chap. 6 sur l'apprentissage multi-échelle). Nous distinguons principalement deux **catégories** de représentations : (1) les **embeddings** sub-symboliques (ex. Word2Vec, BERT), (2) les **représentations** symboliques (ensembles de mots-clés, règles logiques).

8.4.1.1. Word embeddings (Word2Vec, GloVe), contextualisés (BERT, GPT)

L'une des pièces fondamentales du **Deep Synergy Learning** (DSL) appliquée aux données **textuelles** est la manière de représenter chaque **mot** ou **segment** de phrase sous forme d'un **vecteur** dans un espace latent. Historiquement, des méthodes dites *statiques* comme **Word2Vec** ou **GloVe** ont été introduites, pour ensuite évoluer vers des méthodes *contextualisées* comme **BERT** ou **GPT**. Cette évolution mathématique reflète une sophistication croissante de la notion de "similarité sémantique" entre mots.

Embeddings statiques : Word2Vec, GloVe

Les approches de type **Word2Vec** (Skip-gram, CBOW) ou **GloVe** (Global Vectors) produisent une **projection vectorielle** de dimension d pour chaque mot du vocabulaire, notée $\mathbf{v}_w \in \mathbb{R}^d$. L'idée mathématique repose sur l'observation que des mots apparaissant fréquemment dans le même contexte (voisinage) se rapprochent dans l'espace vectoriel. Dans le cas de **Word2Vec (Skip-gram)**, l'objectif d'entraînement s'écrit

$$\max \prod_{(w,c) \in \mathcal{D}} P(c | w),$$

où \mathcal{D} est un ensemble de paires (mot w , mot-contexte c) tirées d'une large collection de textes, et où la probabilité $P(c | w)$ se modélise via un petit réseau de neurones. La **similarité cosinus**

$$\cos(\mathbf{v}_w, \mathbf{v}_{w'}) = \frac{\mathbf{v}_w \cdot \mathbf{v}_{w'}}{\|\mathbf{v}_w\| \|\mathbf{v}_{w'}\|}$$

mesure la proximité sémantique entre deux mots (w) et (w'). **GloVe**, quant à lui, opère sur la **matrice** de co-occurrences globales. Il s'appuie sur un tableau ($X_{\{w,w'\}}$) qui *comptabilise le nombre d'occurrences simultanées des mots (w) et (w')*. L'objectif est alors d'ajuster les vecteurs de telle sorte que (\mathbf{v}_w) soit proportionnel à ($\log(X_{\{w,w'\}})$). Dans les deux cas, chaque mot w hérite donc d'un unique **vecteur** \mathbf{v}_w . Ces embeddings sont dits *statiques* car un mot polysémique conserve la même représentation, indépendamment du contexte réel de la phrase. Les dimensions courantes vont de 50 à 300, ce qui assure un compromis entre la richesse sémantique et la facilité de manipulation.

Embeddings contextualisés : BERT, GPT

Les progrès récents en **traitement du langage** ont introduit la notion de **contexte**. Un mot comme "bank" en anglais (rive ou banque) ne saurait être réduit à un unique vecteur. Les modèles **BERT**, **GPT**, **RoBERTa**, etc., s'appuient sur

l’architecture **Transformer** pour produire des vecteurs qui varient selon la phrase. Mathématiquement, un Transformer recourt à des **couches** d’attention multi-têtes, formellement :

$$\text{Attention}(Q, K, V) = \text{softmax}\left(QK^\top/\sqrt{d}\right)V,$$

où Q, K, V sont des projections linéaires de l’entrée (chaque token), et d en est la dimension. En **BERT**, on entraîne l’encodeur Transformer via des objectifs tels que “masked language modeling” (prédir les tokens masqués) et “next sentence prediction”. En **GPT**, on adopte une modélisation auto-régressive (chaîne de Markov cachée, où l’on prédit chaque token après les précédents). Les vecteurs résultants $\mathbf{v}_{w,t} \in \mathbb{R}^{768}$ (par exemple) reflètent la **signification** d’un mot w en position t , tenant compte de toute la phrase. Chaque occurrence du même mot dispose alors d’un embedding potentiellement différent, levant la limite des approches statiques. D’un point de vue mathématique, on obtient un nuage de vecteurs plus précis, aux dimensions parfois élevées (768, 1024...), mieux adaptés aux ambiguïtés lexicales.

Dimension et variation

Les modèles statiques (Word2Vec, GloVe) proposent des dimensions typiques de l’ordre de 100 à 300, tandis que **BERT base** en comporte souvent 768, GPT-2 large peut s’étendre jusqu’à 1024, et d’autres modèles extralarges (p. ex. GPT-3) vont encore plus loin. Manipuler des vecteurs en dimension 768 ou 1024 dans un **DSL** accroît la **richesse** potentielle de $S(\cdot, \cdot)$, mais alourdit le calcul $O(n^2d)$. Des techniques de normalisation (section 8.3.1.3) ou d’approximate nearest neighbor (chap. 7.2.3) sont alors invoquées pour gérer la complexité.

A. Synergie et DSL

Dans le cadre du **DSL**, chaque token ou mot \mathcal{E}_i devient un **nœud** dans le **SCN**. Son embedding $\mathbf{v}_i \in \mathbb{R}^d$ (issu de Word2Vec, BERT, etc.) permet de définir la **synergie** :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \max\{0, \cos(\mathbf{v}_i, \mathbf{v}_j)\} \quad \text{ou} \quad \exp(-\lambda \|\mathbf{v}_i - \mathbf{v}_j\|),$$

de sorte que deux **tokens** textuels reçoivent un score élevé s’ils sont jugés proches dans l’espace sémantique. La règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

entraîne un **renforcement** $\omega_{i,j}$ si les embeddings se ressemblent, formant ainsi des **clusters** de mots sémantiquement proches. Dans un **DSL** multimodal, ce même principe peut lier des tokens textuels à d’autres types d’entités (images, audio), d’où la dimension centrale du choix d’un bon embedding.

B. Retombées dans le SCN

Le fait de travailler avec des embeddings **contextualisés** (BERT, GPT) offre la capacité de distinguer, au sein du **SCN**, plusieurs occurrences d’un même mot dans des contextes différents. Chacune se voit attribuer un embedding $\mathbf{v}_{w,t}$. Les nœuds correspondant à “bank” dans un texte financier se rapprochent entre eux, tandis que ceux qui indiquent “bank” en tant que “rive de fleuve” forment un autre **cluster**. En outre, si l’on agrège les tokens d’une phrase ou d’un document, on peut obtenir un vecteur global document \mathbf{d} (par moyenne ou attention pooling). Il devient alors possible de constituer des **macro-nœuds** pour des paragraphes, des chapitres, etc., toujours selon le même modèle de synergie et de mise à jour ω .

Le **coût** algorithmique des comparaisons cosinus $O(n^2d)$ peut nécessiter des heuristiques (LSH, k-NN) si le nombre de tokens n et la dimension d sont grands. Mais le gain en **précision sémantique** est considérable, surtout lorsque l’on applique le **DSL** à des corpus textuels riches ou hétérogènes.

Conclusion

Les **embeddings** sub-symboliques, qu’ils soient statiques (Word2Vec, GloVe) ou **contextualisés** (BERT, GPT), constituent la **fondation** pour représenter le **texte** dans un **Synergistic Connection Network**. Sur le plan **mathématique**, il s’agit de placer chaque token (ou segment) \mathcal{E}_i dans un espace vectoriel $\mathbf{v}_i \in \mathbb{R}^d$, où la **similarité** cosinus ou la **distance** eucildienne répercutent la proximité sémantique. Les méthodes statiques, de dimension 50–300, conviennent à une approche globale mais ignorent la polysemie, tandis que les méthodes contextualisées,

dimensions 768 ou plus, captent les nuances de sens selon le contexte. Cette différence influence la **dynamique** du DSL. Un embedding lexical de meilleure qualité améliore la **synergie**, rendant la formation des **clusters** textuels plus cohérente et pertinente. Ainsi, le choix de Word2Vec, GloVe, BERT ou GPT conditionne la **qualité** et la **finesse** de l'auto-organisation textuelle, et ouvre la voie à une **fusion** multimodale plus performante lorsque l'on associe le texte à d'autres canaux (vision, audio, etc.).

8.4.1.2. Forme vectorielle sub-symbolique, dimension variable (300, 768...)

La **représentation** d'entités linguistiques ou multimédias par des **vecteurs** réels est au cœur de l'approche sub-symbolique. Dans un **Deep Synergy Learning** (DSL) appliqué à un **Synergistic Connection Network** (SCN), il est fréquent de disposer d'**embeddings** (Word2Vec, GloVe, BERT, GPT, CNN, etc.) dont la dimension s'élève à 300, 512, 768, voire plus. Le présent segment met en lumière les implications mathématiques et conceptuelles de l'usage de tels vecteurs à la fois riches et complexes.

A. Vectorisation : concept et dimensionnalité

Le **principe** consiste à associer, à chaque entité \mathcal{E}_i (un mot, un token BERT, un patch d'image, etc.), un vecteur $\mathbf{v}_i \in \mathbb{R}^d$. Ces vecteurs se qualifient de *sub-symboliques* puisqu'ils n'expriment pas directement une structure logique ou des symboles explicites, mais condensent l'information (ex. sémantique, contextuelle) en des coordonnées réelles. Les tailles les plus fréquentes dans la pratique se situent entre 300 et 1024, selon que l'on emploie des modèles statiques (Word2Vec/GloVe) ou des architectures transformer (BERT, GPT) plus profondes.

Sur le plan **mathématique**, ce choix de dimension d représente un compromis entre la **capacité** à capturer des nuances (un vecteur trop faible ne peut encoder suffisamment de variabilité) et la **complexité** algébrique (coût en $O(n^2d)$ si on compare tous les vecteurs). Certains modèles de traitement de texte se limitent à 300 dimensions (Word2Vec/GloVe), quand d'autres comme BERT base utilisent 768, GPT-2 large peut monter à 1024, etc. Dans un **SCN**, rien n'interdit d'avoir des vecteurs de dimensions différentes selon les modalités, quitte à définir un **mapping** ou des formules S distinctes.

B. Calcul de la synergie $S(i, j)$ à partir de vecteurs

Dans un **DSL**, la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ doit rendre compte de la **distance** ou de la **similarité** entre \mathbf{v}_i et \mathbf{v}_j . Les fonctions les plus usuelles incluent :

- **La similarité cosinus** $\cos(\mathbf{v}_i, \mathbf{v}_j)$, définie par

$$\cos(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}.$$

- **La distance euclidienne** $\|\mathbf{v}_i - \mathbf{v}_j\|$, puis on inverse ou exponentie cette distance afin d'obtenir un score de proximité (ex. RBF-kernel $\exp(-\alpha \|\mathbf{v}_i - \mathbf{v}_j\|^2)$).

En appliquant le **DSL**, on associe à chaque couple (i, j) une pondération $\omega_{i,j}$, mise à jour par la règle

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathbf{v}_i, \mathbf{v}_j) - \tau \omega_{i,j}(t)].$$

Deux entités à vecteurs très proches (cosinus élevé, distance faible) voient $\omega_{i,j}$ s'accentuer, trahissant leur "affinité" dans le **SCN**. Ce mécanisme engendre des **clusters** ou des "communautés" de vecteurs similaires.

C. Intégration dans le SCN

Sur le plan **architectural** (chap. 5), on réserve un "Module Synergie" pour traiter les vecteurs sub-symboliques. Ce module calcule $S(i, j)$ à partir des embeddings $\mathbf{v}_i, \mathbf{v}_j$. En cas de **multimodalité** hétérogène (texte vs. image), on peut projeter chaque vecteur dans un espace commun ou définir plusieurs synergies spécialisées.

Les **dimensions** typiques 300, 768, 1024 s'accompagnent de techniques de normalisation (on pose $\mathbf{v} \leftarrow \mathbf{v}/\|\mathbf{v}\|$) afin de mieux stabiliser les valeurs $S(i, j)$ et d'empêcher qu'une entité à très grande norme ne domine le calcul. Dans un **SCN** de grande envergure (n élevé), la comparaison $O(n^2d)$ s'avère coûteuse, d'où l'introduction de méthodes d'approximation (LSH, k-NN) évoquées en chap. 7 pour éviter la combinatoire complète.

D. Considérations mathématiques autour de la grande dimension

La “malédiction de la dimension” fait qu'en très haute dimension, la distance $\|\mathbf{v}_i - \mathbf{v}_j\|$ perd parfois en contraste, comme les points peuvent tous se retrouver (en termes relatifs) assez espacés. C'est pourquoi la **similarité cosinus** ou un **kernel** (RBF) adéquat est souvent plus indiquée, accompagnée d'une étape de normalisation. De plus, le DSL n'exige pas un usage unique de cosinus ou euclidienne. D'autres formules $S(\mathbf{v}_i, \mathbf{v}_j)$ sont admissibles, pourvu qu'elles reflètent la proximité souhaitée.

Conclusion

La **représentation** sub-symbolique en vecteurs de taille 300, 768 ou autre demeure un **socle** du **DSL** pour manipuler des entités textuelles (Word2Vec, GloVe, BERT...) ou multimédias (CNN, transformers, etc.). Les vecteurs se placent dans \mathbb{R}^d , où la dimension d reflète un compromis entre la **richesse** de l'information et la **charge** de calcul. Sur le plan **mathématique**, on définit la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ comme une fonction dépendant de $\|\mathbf{v}_i - \mathbf{v}_j\|$ ou $\cos(\mathbf{v}_i, \mathbf{v}_j)$. Ce score gouverne l'équation de mise à jour $\omega_{i,j}(t+1) = \dots$, menant à un **Synergistic Connection Network** où les entités dotées de vecteurs proches se lient fortement. Ainsi, le DSL exploite la **puissance** d'un embedding sub-symbolique expressif tout en s'adaptant automatiquement via la **dynamique** auto-organisée.

8.4.1.3. Approches symboliques (ensembles de mots, règles logiques) vs. embeddings (et exemples d'overlap)

Les **représentations** dans un **Deep Synergy Learning** (DSL) peuvent s'inspirer tantôt d'un **mode symbolique** (ex. ensembles de mots, règles logiques, ontologies), tantôt d'un **mode sub-symbolique** (embeddings vectoriels issus de réseaux neuronaux). Cette coexistence illustre une dualité classique entre la **transparence** et la **rigueur** du symbolique d'une part, et la **flexibilité** ainsi que la **robustesse** des vecteurs appris d'autre part. Le **Synergistic Connection Network** (SCN) du DSL, loin d'en privilégier exclusivement un, peut exploiter les deux et définir la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ comme une combinaison ou un choix adaptatif.

A. Représentations symboliques : ensembles de mots, règles logiques

Un cas courant consiste à modéliser une entité \mathcal{E}_i comme un **ensemble** de mots (ou de concepts) $\{w_1, \dots, w_k\} \subseteq V$. On peut alors définir la **synergie** $S_{\text{symb}}(\mathcal{E}_i, \mathcal{E}_j)$ selon une **mesure d'overlap**. L'exemple le plus cité demeure l'**indice de Jaccard**, qui se formule :

$$S_{\text{symb}}(\mathcal{E}_i, \mathcal{E}_j) = \frac{|\mathcal{E}_i \cap \mathcal{E}_j|}{|\mathcal{E}_i \cup \mathcal{E}_j|}.$$

Si \mathcal{E}_i et \mathcal{E}_j partagent de nombreux mots, leur Jaccard est élevé. Dans un **SCN**, cela se traduit par une **pondération** $\omega_{i,j}$ plus forte lorsque les ensembles sont proches. Cette approche symbolique est **explicable** : on peut lister les mots communs justifiant la synergie, et prendre en compte des variations plus raffinées (pondération TF-IDF plutôt qu'un simple set).

Au-delà des simples ensembles de mots, on peut disposer de **règles** (ex. “si X alors Y”), ou d'une **ontologie** plus structurée (graphes conceptuels, taxonomies). Pour comparer deux entités symboliques $\mathcal{E}_i, \mathcal{E}_j$ — chacune étant un mini-ensemble de règles ou un segment ontologique — on définit un score S_{symb} via la fraction de clauses communes, la **compatibilité logique**, ou l'exemple d'unification (combien de substitutions rendent les deux bases cohérentes). Une forme simplifiée serait :

$$S_{\text{symb}}(\mathcal{E}_i, \mathcal{E}_j) = \text{overlap}(\mathcal{R}_i, \mathcal{R}_j) = \frac{|\mathcal{R}_i \cap \mathcal{R}_j|}{|\mathcal{R}_i \cup \mathcal{R}_j|}.$$

Cette expression d'overlap, proche du Jaccard, fonctionne tant qu'on voit les ensembles logiques $\mathcal{R}_i, \mathcal{R}_j$ comme un listing de formules. D'un point de vue **DSL**, l'inconvénient réside dans un éventuel **coût** combinatoire de comparaison si $\mathcal{R}_i, \mathcal{R}_j$ sont volumineux ou si la logique est complexe, mais on y gagne en **lisibilité** et en **fondement théorique**.

B. Embeddings sub-symboliques

Les approches sub-symboliques reposent sur la **vectorisation** de chaque entité \mathcal{E}_i en $\mathbf{v}_i \in \mathbb{R}^d$. On peut tirer \mathbf{v}_i d'une méthode Word2Vec, GloVe, BERT ou GPT (dimension 300, 768, 1024, etc.). Sur le plan **mathématique**, deux vecteurs $\mathbf{v}_i, \mathbf{v}_j$ se comparent via la **similarité** cosinus ou la **distance** euclidienne, puis on en dérive un score :

$$S_{\text{embed}}(\mathbf{v}_i, \mathbf{v}_j) = \max\{0, \mathbf{v}_i \cdot \mathbf{v}_j\} \quad (\text{si normalisés, par ex.}),$$

ou un RBF-kernel $\exp(-\|\mathbf{v}_i - \mathbf{v}_j\|^2/\sigma^2)$.

La sous-représentation sub-symbolique se montre **robuste** aux variations lexicales, gère la synonymie et la polysémie (surtout pour BERT). Elle permet des comparaisons en produit scalaire, nettement moins complexes qu'une unification de clauses logiques. L'inconvénient réside dans la **boîte noire**, car il devient moins évident de déterminer quels "mots" ou "règles" expliquent la proximité entre deux embeddings. D'un point de vue **DSL**, cela favorise toutefois la construction d'un **SCN** où l'on compare rapidement un grand nombre d'entités.

C. Concilier Symbolique et Embeddings dans le DSL

Un **SCN** multimodal ou multi-représentation peut intégrer :

591. **Une composante symbolique** : ensemble de mots \mathcal{W}_i ou ensemble de règles \mathcal{R}_i . On définit une synergie $S_{\text{symb}}(i, j)$ via un *overlap* (ex. Jaccard).

592. **Une composante embedding** : vecteur \mathbf{v}_i . On définit un autre score $S_{\text{embed}}(i, j)$ via cosinus ou RBF.

593. **Fusion** :

$$S(i, j) = \alpha S_{\text{symb}}(i, j) + (1 - \alpha) S_{\text{embed}}(i, j),$$

ou toute autre forme de combinaison. La mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$$

tient alors compte simultanément de la portion symbolique et de la portion sub-symbolique.

Exemple

Si \mathcal{E}_i est un document associant $\mathcal{W}_i \subseteq V$ et un embedding BERT $\mathbf{v}_i \in \mathbb{R}^{768}$, alors la synergie avec un autre document \mathcal{E}_j combine $\text{overlap}(\mathcal{W}_i, \mathcal{W}_j)$ et $\cos(\mathbf{v}_i, \mathbf{v}_j)$. On peut ainsi mettre en avant la **convergence** symbolique (des mots identiques) et la **proximité** sémantique sub-symbolique (mêmes thèmes, même usage lexical).

D. Implications et choix de conception

On peut concrètement choisir l'**overlap** Jaccard pour l'ensemble de mots \mathcal{W}_i , par exemple :

$$S_{\text{symb}}(\mathcal{W}_i, \mathcal{W}_j) = \frac{|\mathcal{W}_i \cap \mathcal{W}_j|}{|\mathcal{W}_i \cup \mathcal{W}_j|}.$$

Dans un **DSL** textuel, cela se traduit par un haut score si deux entités partagent la plupart de leurs mots ou concepts. À l'inverse, si elles sont disjointes lexicalement, le score tombe à 0 (ou proche).

La **partie symbolique** demeure plus "interprétable" (on sait quels mots recouvrent l'entité) et s'intègre mieux à des modules logiques (on peut raisonner sur les règles). Par contre, la **partie embedding** apporte un surcroît de robustesse envers la synonymie, l'orthographe, la flexion, etc. En combinant les deux, le SCN devient plus complet, mais le calcul du score $S(i, j)$ s'allourdit.

Cette hybridation se révèle précieuse dans des systèmes cognitifs ou multi-agents, où la **logique** (symbolique) et la **similarité** (sub-symbolique) doivent coexister. Un **SCN** opérant la mise à jour $\omega \leftarrow \omega + \eta[S - \tau \omega]$ sur un score mixte parvient à organiser les entités selon les *deux* critères, délimitant des clusters où l'**overlap** lexical est fort *et/ou* la **distance embedding** est faible.

Conclusion

Les **approches symboliques** (ensembles de mots, règles logiques) et les **embeddings** (représentations sub-symboliques) ne se **concurrencent** pas nécessairement dans un **DSL**, elles peuvent plutôt se **compléter**. Les entités symboliques profitent du critère d'**overlap** (exemple Jaccard ou unification de règles) pour leur synergie, tandis que les entités vectorielles recourent à une **distance** ou **similarité** continue. Le **SCN** (Synergistic Connection Network) intègre ces deux approches en définissant

$$S(i, j) = \alpha S_{\text{symb}}(i, j) + (1 - \alpha) S_{\text{embed}}(i, j),$$

et en laissant la mise à jour $\omega_{i,j}(t+1) = \dots$ faire émerger un **réseau** respectant simultanément les contraintes logico-symboliques et les critères de similarité vectorielle.

Sur le plan **mathématique**, cette stratégie confère une **richesse** double : l'**explicabilité** (symbolique) et la **performance** (embedding). Dans un **DSL** multimodal ou multi-représentation, cette fusion ouvre la voie à des **clusters** plus pertinents et plus interprétables, offrant une vision **hybride** entre symboles et nombres.

8.4.2. Synergie Langage-Langage

Lorsqu'il s'agit de mesurer la **synergie** entre deux entités purement **linguistiques** (ex. phrases, documents, segments de texte), le **SCN** (Synergistic Connection Network) doit disposer de **fonctions** capables d'évaluer la **similarité textuelle** ou le **partage sémantique**. Cette **section** (8.4.2) met l'accent sur la manière dont le **DSL** (Deep Synergy Learning) gère la synergie entre deux **blocs** de langage, qu'il s'agisse de phrases, de paragraphes ou de textes entiers.

8.4.2.1. Mesures de similarité textuelle (cosinus, tokens communs, etc.)

Pour **évaluer** la **synergie** entre deux entités textuelles dans un **Deep Synergy Learning** (DSL), il est essentiel de disposer d'une **mesure** de similarité adaptée. Les méthodes classiques reposent sur des représentations vectorielles ou sur le comptage de tokens (mots, n-grammes), et elles s'intègrent parfaitement dans un **Synergistic Connection Network** (SCN). Le DSL se contente d'une fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ en sortie, quel que soit le détail de son calcul.

Une approche omniprésente consiste à convertir chaque entité textuelle \mathcal{E}_i en un **vecteur** $\mathbf{v}_i \in \mathbb{R}^d$. Ce vecteur peut correspondre aux **embeddings** d'un mot, d'une phrase ou d'un document, issus de Word2Vec, GloVe, BERT, GPT, ou encore d'une analyse TF-IDF. Une fois ces vecteurs produits, la mesure la plus straightforward demeure la **similarité cosinus** :

$$\text{SimCos}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}.$$

Si $\text{SimCos} \approx 1$, cela indique que \mathbf{v}_i et \mathbf{v}_j pointent dans des directions similaires, gage d'une forte ressemblance sémantique entre les entités textuelles \mathcal{E}_i et \mathcal{E}_j . Dans un **SCN**, cette valeur de similarité peut alimenter la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ de manière directe, par exemple :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \max(0, \text{SimCos}(\mathbf{v}_i, \mathbf{v}_j)),$$

afin de garantir un score positif. Sur le plan **mathématique**, le calcul du cosinus s'avère $O(d)$, ce qui reste relativement accessible tant que le nombre d'entités n et la dimension d ne deviennent pas trop grands (cf. chap. 7 pour la question de la complexité $O(n^2d)$).

Une seconde famille de mesures s'appuie sur le **comptage** de tokens (mots, n-grammes). Si \mathcal{E}_i et \mathcal{E}_j sont modélisés comme des **ensembles** de termes T_i et T_j , on peut définir un score d'**overlap**. Un exemple typique est l'**indice de Jaccard** :

$$\text{Jaccard}(\mathcal{E}_i, \mathcal{E}_j) = \frac{|T_i \cap T_j|}{|T_i \cup T_j|}.$$

Une version différente introduit un **facteur** de normalisation basé sur la longueur, ou des **poids** TF-IDF pour accentuer l'importance des mots rares et réduire celle des mots très fréquents. Par exemple, pour un vecteur TF-IDF \mathbf{v}_i (de dimension d), la **similarité cosinus** :

$$\text{Sim}_{\text{tfidf}}(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$$

renvoie un score élevé si \mathcal{E}_i et \mathcal{E}_j partagent des mots importants, identifiés par la TF-IDF. Ce principe est souvent utilisé pour comparer rapidement des documents textuels dans le cadre d'un SCN.

Dans la **mise en œuvre** concrète, on peut mixer **un score** Jaccard (ou overlap) pour souligner l'identité exacte de certains mots-clés et **une similarité cosinus** sur embeddings afin de gérer la proximité sémantique plus fine.

On peut alors pondérer ces deux volets :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \alpha \text{Jaccard}(T_i, T_j) + (1 - \alpha) \text{SimCos}(\mathbf{v}_i, \mathbf{v}_j).$$

A. Intégration au DSL : synergie $S(i, j)$

Au sein d'un **SCN**, la règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

applique la **dynamique** auto-organisée. Un texte \mathcal{E}_i voit son lien $\omega_{i,j}$ avec \mathcal{E}_j renforcé si $S(\mathcal{E}_i, \mathcal{E}_j)$ est assez grand. Ainsi, des textes (phrases, documents) partageant de nombreux tokens ou jouissant d'une parenté sémantique dans l'espace embedding reçoivent une synergie plus élevée, ce qui les fait converger dans un **cluster** commun.

B. Ajustements et normalisations

Certaines variantes imposent un **seuil** pour éviter d'affecter un lien inutilement. En pratique,

$$S'(i, j) = \max(0, S(i, j) - \delta)$$

peut garantir que seules les paires avec $S(i, j) > \delta$ méritent d'entrer dans la mise à jour. Il est également coutumier de **normaliser** la similarité dans $[0,1]$, par exemple en réécrivant :

$$\text{SimCos}(\mathbf{v}_i, \mathbf{v}_j) \leftarrow \frac{1 + \text{SimCos}(\mathbf{v}_i, \mathbf{v}_j)}{2}$$

si SimCos peut devenir négatif.

Conclusion

Dans un **DSL** appliqué au **texte**, plusieurs stratégies de **similarité** permettent de définir la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$. Les plus usuelles incluent :

594. **Cosinus** sur des **vecteurs** (embeddings Word2Vec, BERT, TF-IDF),

595. **Jaccard** ou **overlap** sur l'**ensemble** de tokens (mots, n-grammes).

Cette fonction de similarité, une fois choisie, s'intègre dans la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$, entraînant des **clusters** auto-organisés de documents ou de segments textuels sémantiquement proches. Le réglage (filtrage par un seuil, fusion de plusieurs mesures) influe sur la granularité des **communautés** linguistiques détectées, démontrant la **flexibilité** du DSL pour organiser le contenu textuel.

8.4.2.2. “Topic synergy” : si deux documents ou phrases partagent un thème, la pondération se renforce

Dans l’analyse textuelle, il est fréquent de repérer, pour chaque document ou phrase, les **thèmes** (ou *topics*) qui y sont prédominants. Au sein d’un **Deep Synergy Learning** (DSL), cette notion de **thème** peut être exploitée pour définir une **synergie** entre deux entités textuelles \mathcal{E}_i et \mathcal{E}_j . Si ces deux entités abordent des sujets proches, leur **pondération** $\omega_{i,j}$ tend à se renforcer, formant des **clusters** de textes thématiquement similaires. Cette logique dite “topic synergy” enrichit le **Synergistic Connection Network** (SCN) en permettant une organisation auto-émergente par **thèmes**.

A. Évaluation de la “topic synergy”

Pour estimer la proximité thématique, on modélise chaque entité textuelle \mathcal{E}_i par un **vecteur de topics** $\mathbf{v}_i \in \mathbb{R}^K$. Souvent, on emploie un algorithme de topic modeling comme **LDA** (Latent Dirichlet Allocation) ou **NMF** (Nonnegative Matrix Factorization) pour extraire K grandes “thématiques” dans un corpus. Le vecteur $\mathbf{v}_i = (v_{i,1}, \dots, v_{i,K})$ décrit la distribution de l’entité i sur les K thèmes. Chaque composante $v_{i,k}$ indique le **poids** ou la **probabilité** du thème k dans \mathcal{E}_i .

Pour deux entités $\mathcal{E}_i, \mathcal{E}_j$ on définit la **synergie** par la similarité entre \mathbf{v}_i et \mathbf{v}_j . Une formule répandue demeure la **similarité cosinus** :

$$S_{\text{topic}}(i, j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|}$$

Si $\cos(\mathbf{v}_i, \mathbf{v}_j) \approx 1$, cela signale un fort recouvrement thématique, c’est-à-dire que les **topiques** dominants de \mathcal{E}_i coïncident avec ceux de \mathcal{E}_j . On peut encore opter pour la distance euclidienne inversée $1/(1+\|\mathbf{v}_i - \mathbf{v}_j\|)$ ou d’autres kernels. Sur le plan **mathématique**, ce n’est qu’une **fonction** $S_{\text{topic}}: \mathbb{R}^K \times \mathbb{R}^K \rightarrow \mathbb{R}^+$; le DSL en tire un score de synergie entre entités.

La règle de mise à jour (cf. chap. 4)

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

utilise $S(\mathcal{E}_i, \mathcal{E}_j)$. Si on s’appuie sur la “topic synergy” comme critère, cela signifie que deux documents partageant fortement un même topique se voient **renforcer** leur liaison $\omega_{i,j}$. Progressivement, des **clusters** se dessinent autour de textes **thématiquement** convergents.

B. Pertinence pour la collaboration de documents ou de phrases

Au niveau **macro**, la distribution de topiques $\mathbf{v}_i \in \mathbb{R}^K$ encode quel pourcentage de chaque document \mathcal{E}_i se rattache aux différents thèmes. Deux documents abordant le thème “Finance” (haute composante sur le topic “Finance”) et “Économie internationale” se rapprochent, alors qu’un document traitant de “Biologie marine” sera éloigné. Ce SCN se spécialise alors en regroupant les documents par **thèmes**, révélant des clusters sémantiques.

À un niveau plus fin, on peut découper un texte en **phrases** ou **paragraphes**, puis leur attribuer des **distributions** de thèmes locales. Les clusters du SCN refléteront alors non pas l’appartenance globale d’un document, mais la portion thématique précise où la synergie s’avère forte (par ex. deux paragraphes décrivant la même technologie ou le même concept).

Dans un système plus complexe, un document fortement relié à un “Topic A” peut “coopérer” avec un autre document partageant ce Topic A, renforçant la connexion $\omega_{i,j}$. Cette *auto-organisation* favorise la **recommandation** d’éléments semblables (ou l’extraction de chapitres sur un thème commun).

C. Mathématiques avancées : combiner “topic synergy” avec d’autres synergies

On peut aisément combiner la “topic synergy” à d’autres mesures, par exemple la similarité lexicale brute ou une similarité cosinus sur des embeddings contextuels (BERT). Mathématiquement, on définit

$$S(i, j) = \alpha S_{\text{topic}}(i, j) + (1 - \alpha) \text{SimCos}(\mathbf{u}_i, \mathbf{u}_j).$$

Le **DSL** intègre ce score dans la mise à jour $\omega_{i,j}$.

Si la distribution de topics \mathbf{v}_i se modifie (ex. on entraîne un modèle de topic modeling plus élaboré), alors la fonction $S_{\text{topic}}(i, j)$ change, déclenchant une reconfiguration des liaisons $\omega_{i,j}$. Ceci illustre la flexibilité d’un **SCN** où la synergie s’ajuste dès que la représentation sous-jacente évolue.

D. Conclusion

La “topic synergy” désigne un **score** qui accroît la pondération $\omega_{i,j}$ entre entités textuelles partageant un **même** thème. Concrètement, on :

596. **Définit** un vecteur thématique $\mathbf{v}_i \in \mathbb{R}^K$ par un algorithme de topic modeling,

597. **Compare** deux vecteurs $\mathbf{v}_i, \mathbf{v}_j$ via une similarité cosinus ou autre,

598. **Insère** le score obtenu dans la formule DSL $\omega_{i,j}(t+1) = \dots$.

Le **SCN** résultant concentre les liaisons entre documents/phrases qui coïncident thématiquement, formant des **clusters** sémantiques plus lisibles et facilitant la **navigation** ou l’**analyse**. Les entités purement distinctes en thèmes restent faiblement connectées. En somme, le critère “topic synergy” offre un **fil** supplémentaire pour la **cohésion** textuelle, s’ajoutant aux similarités lexicales ou embeddings plus généraux, et enrichit la logique d’**auto-organisation** au sein du **DSL**.

8.4.2.3. Inhibition possible si on veut éviter la fusion de textes trop divergents

Dans un **Synergistic Connection Network** (SCN) appliqué à la **fusion** ou la **comparaison** de segments textuels (documents, paragraphes, phrases), on peut rechercher une **cohérence** interne au sein des clusters de textes. Il arrive cependant qu’une mesure de similarité (cosinus, overlap lexical, etc.) attribue un score modéré à deux entités très différentes, risquant alors de les “fusionner” artificiellement et de gâcher la structure thématique. Pour éviter ce phénomène, il est fréquent d’introduire des mécanismes d’**inhibition** dans le **Deep Synergy Learning** (DSL). Ce terme “inhibition” renvoie à une **compétition** empêchant une entité de se lier simultanément à trop d’autres entités, ou de se lier à des entités trop éloignées, renforçant ainsi la cohérence globale.

A. Contexte : fusion de textes et divergence thématique

Dans un scénario où l’on compare des entités textuelles \mathcal{E}_i et \mathcal{E}_j , la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ peut prendre la forme d’une **similarité** cosinus sur des embeddings ou d’un **overlap** de tokens (section 8.4.2.1). Si S s’avère légèrement non nulle pour deux textes très différents, la règle de mise à jour DSL

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$$

peut finir par **entretenir** un lien $\omega_{i,j}$ trop important, conduisant à une “fusion” excessive ou non souhaitée. On obtient alors des **clusters** hétérogènes où les textes se mélangeant alors même qu’ils traitent de thématiques opposées. C’est là qu’intervient l’**inhibition**, un moyen de contenir la prolifération de liens “moyennement justifiés”.

B. Rôle de l’inhibition pour éviter une fusion inappropriée

Le principe d’inhibition indique qu’une entité \mathcal{E}_i dispose d’une **capacité limitée** à établir des connexions fortes. Formulé mathématiquement, on ajoute un **terme** d’inhibition dans la mise à jour :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

où $\gamma > 0$ décrit la force de l'inhibition. De fait, si \mathcal{E}_i cherche déjà à entretenir des liaisons $\omega_{i,k}$ non négligeables avec plusieurs textes, le nouveau lien $\omega_{i,j}$ se voit pénalisé. Les entités entrent en **compétition**, un segment textuel refuse de multiplier les liaisons moyennes, privilégiant les **quelques** plus pertinentes.

Supposons qu'un texte i ait déjà un fort lien $\omega_{i,k}$ avec un segment k très proche thématiquement. S'il tente d'en créer un second $\omega_{i,j}$ vers un texte j assez différent, la somme $\sum_k \omega_{i,k}$ devient élevée, provoquant une inhibition qui maintient $\omega_{i,j}$ à un niveau bas. Ainsi, on évite la "fusion" artificielle de textes hétérogènes.

C. Formules d'inhibition "anti-fusion"

Au-delà du schéma général, on peut concevoir des pénalisations spécifiques, par exemple :

$$\Delta_{\text{inhib}}(i,j) = -\gamma(\omega_{i,j}(t) \mathbf{1}_{S(i,j)<\delta}),$$

où $\mathbf{1}_{S(i,j)<\delta}$ indique que la similarité S est jugée trop faible, mais pas encore nulle. Une telle fonction punit explicitement le cas de liaisons "moyennes", les incitant à décroître rapidement. On peut aussi incrémenter un compteur de liaisons "moyennes" et imposer un maximum par entité \mathcal{E}_i . Sur un plan **mathématique**, cela se traduit par un "cut-off" ou un "softmax" restreignant la somme $\sum_j \omega_{i,j}$.

D. Conclusion

Lors de la **fusion** de segments textuels dans un **SCN**, la **dynamique** DSL peut déraper si elle crée des liens "moyennement élevés" entre des textes très divergents, ruinant la clarté des **clusters**. Le **mécanisme** d'inhibition, déjà exploré en chapitres 4 et 7, permet ici de **refuser** ou **restreindre** la fusion de contenus sans réelle cohérence thématique. Sur un plan **mathématique**, l'ajout d'un terme d'inhibition dans la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \mathcal{I}_{i,j},$$

où $\mathcal{I}_{i,j}$ représente la force d'inhibition, contraint $\omega_{i,j}$ à se réduire si \mathcal{E}_i maintient déjà trop de liaisons ou si $S(i,j)$ n'est pas assez fiable. Cela **garantit** qu'on n'associe pas artificiellement des textes aux thématiques trop distinctes, assurant une **structure** plus cohérente dans le SCN. Ainsi, au sein d'un **DSL** textuel, l'**inhibition** incarne un **levier** essentiel pour préserver la netteté des **clusters** sémantiques.

8.4.3. Liens Texte–Autres Modalités

Lors du traitement de données multimodales, la **composante textuelle** occupe souvent un rôle central en fournissant des **descripteurs sémantiques** tels que mots-clés, légendes ou tags pour les autres modalités, tout en s'appuyant sur celles-ci pour enrichir le contexte. Dans le cadre du **DSL** (Deep Synergy Learning), on cherche à **relier** les entités textuelles (phrases, segments, mots) aux entités d'autres modes (images, audio, etc.) via une **synergie** $S(\mathcal{E}_{\text{texte}}, \mathcal{E}_{\text{autre}})$. Le chapitre 8.4.3 propose donc d'examiner plusieurs **dimensions** de ce lien texte–autres modalités, en insistant sur la manière dont le SCN (Synergistic Connection Network) tisse des **associations** ou clusters entre ces différents flux d'information.

8.4.3.1. Texte–Image : association sémantique (caption, tags)

Un **Synergistic Connection Network** (SCN) appliqué à un contexte **multimodal** peut relier du **texte** (mots, phrases, descriptions) à des **images** (embeddings visuels, patchs, régions d'intérêt). L'idée est d'évaluer pour chaque couple $(\mathcal{E}_{\text{text}}, \mathcal{E}_{\text{img}})$ une **synergie** $S(\mathcal{E}_{\text{text}}, \mathcal{E}_{\text{img}})$ qui reflète la correspondance ou la compatibilité entre le texte et l'image. Lorsqu'on s'intéresse aux associations sémantiques, on peut utiliser ce **SCN** pour des tâches comme le **captioning** (génération de légendes) et le **tagging** (assignation de mots-clés) d'images, sans recourir à un pipeline dédié.

A. Représentation des entités texte et image

Embeddings textuels

Chaque unité textuelle \mathcal{E}_{txt} (mot, phrase, document) peut être convertie en un vecteur $\mathbf{v}_{\text{txt}} \in \mathbb{R}^{d_t}$, via, par exemple, Word2Vec, GloVe, BERT, GPT, ou TF-IDF. Cette étape confère à l'entité textuelle une position dans un espace sémantique. On comparera ensuite deux vecteurs \mathbf{v}_{txt} par un produit scalaire ou une distance.

Pour chaque image \mathcal{E}_{img} , on recourt à un **réseau convolutionnel** (ex. ResNet, VGG) ou un **transformer** (ViT) pour extraire un embedding $\mathbf{v}_{\text{img}} \in \mathbb{R}^{d_i}$. L'analyse peut se faire globalement sur l'image entière ou localement sur des bounding boxes ou patchs.

Dans un **DSL** multimodal, la **synergie** $S(\mathcal{E}_{\text{txt}}, \mathcal{E}_{\text{img}})$ découle d'une **fonction** $\kappa(\mathbf{v}_{\text{txt}}, \mathbf{v}_{\text{img}})$. Selon la mise en œuvre, on peut

$$\kappa(\mathbf{v}_{\text{txt}}, \mathbf{v}_{\text{img}}) = \mathbf{v}_{\text{txt}}^\top \mathbf{W} \mathbf{v}_{\text{img}},$$

ou plus simplement projeter $\mathbf{v}_{\text{txt}}, \mathbf{v}_{\text{img}}$ dans un espace commun (style CLIP) et faire un **cosinus**. Le **SCN** se contente d'une **valeur** positive décrivant la proximité sémantique entre le texte et l'image.

B. Association sémantique (captions, tags)

Dans un **SCN**, si un segment textuel \mathcal{E}_{txt} affiche une forte synergie $S(\mathcal{E}_{\text{txt}}, \mathcal{E}_{\text{img}})$ avec une image \mathcal{E}_{img} , la règle de mise à jour DSL

$$\omega_{(\text{img}, \text{txt})}(t+1) = \omega_{(\text{img}, \text{txt})}(t) + \eta [S(\mathbf{v}_{\text{img}}, \mathbf{v}_{\text{txt}}) - \tau \omega_{(\text{img}, \text{txt})}(t)]$$

renforce ce lien ω . Au terme de l'auto-organisation, on verra **clusters** contenant l'image et un ensemble de tokens/phrases très liés, interprétable comme une légende. Par exemple, une image de chat se retrouvera fortement associée aux mots "cat", "feline", "grey fur", etc.

Sur un plan plus discret, on peut assigner des **tags** (mots-clés) à une image si la synergie cross-modale est au-dessus d'un certain seuil. En pratique, cela veut dire qu'un vecteur pour "dog" se rapprochera fortement de l'embedding visuel détectant un canidé. L'**auto-organisation** conduit la pondération $\omega_{(\text{img}, \text{"dog"})}$ à s'élever, signifiant que l'image reçoit le tag "dog". D'autres tags moins compatibles déclinent.

C. Avantages mathématiques : couplage embeddings–synergie

D'un point de vue **algébrique**, on peut chercher à projeter \mathbf{v}_{txt} et \mathbf{v}_{img} vers un espace commun \mathbb{R}^d grâce à des matrices \mathbf{W}_{txt} et \mathbf{W}_{img} . Ensuite, la **similarité** cross-modal $\kappa(\mathbf{v}_{\text{txt}}, \mathbf{v}_{\text{img}})$ devient un simple produit scalaire. L'approche **CLIP** (OpenAI) opère un apprentissage supervisé sur un gigantesque corpus d'images-légendes pour aligner \mathbf{v}_{img} et \mathbf{v}_{txt} .

Le grand apport du DSL est qu'il **renforce** de façon auto-organisée un lien $\omega_{i,j}$ si la synergie demeure élevée. Ainsi, aucune instruction explicite n'est requise pour dire "ce texte décrit cette image" ; la dynamique ω le **découvre** si, en pratique, κ reste haut pour un couple image–texte. On aboutit alors à un **cluster** d'images et un cluster de mots/phrases partiellement confondus, traduisant la correspondance sémantique.

D. Cas d'utilisation dans un SCN global

Une fois un **SCN** construit (images + segments textuels), on peut, pour un nouveau mot, repérer quelles images possèdent déjà une liaison élevée ω avec ce mot (ou un mot synonyme). Le **DSL** agit comme un **index** sémantique en exploitant les arcs ω qui relient des textes proches du nouveau mot à telle ou telle image.

En sens inverse, on peut découvrir quels mots ou légendes s'agglomèrent autour d'un groupe d'images semblables, ce qui vaut étiquetage ou classification. La structure **auto-organisée** permet également de "fusionner" l'information visuelle et textuelle pour générer ou compléter des légendes de manière plus souple.

Conclusion

Dans un **DSL** multimodal, la **synergie** texte–image résume l'**association sémantique** entre un embedding textuel \mathbf{v}_{txt} et un embedding visuel \mathbf{v}_{img} . Il peut s’agir d’un produit scalaire (avec ou sans projection commune) ou d’une distance inversée. Dans le **Synergistic Connection Network**, la **dynamique** :

$$\omega_{(\text{img}, \text{txt})}(t+1) = \omega_{(\text{img}, \text{txt})}(t) + \eta [S(\mathbf{v}_{\text{img}}, \mathbf{v}_{\text{txt}}) - \tau \omega_{(\text{img}, \text{txt})}(t)]$$

renforce les liens correspondants, produisant des **clusters** ou macro-nœuds reliant images et libellés textuels. Cette alliance s’avère cruciale pour :

- **Tagger** automatiquement des images selon leurs descriptifs textuels,
- **Créer** des légendes (*captioning*) de façon plus ou moins supervisée,
- **Rechercher** des images à partir d’un mot ou requête textuelle (et vice versa).

Le SCN fournit donc un cadre **auto-adaptatif** pour la **fusion** texte–image, associant progressivement les entités les plus compatibles, jusqu’à former une **structure** où chaque image se trouve étiquetée (ou légendée) par le texte le mieux corrélé, tout en révélant des associations utiles pour la **navigation** sémantique.

8.4.3.2. Texte–Audio : ex. transcript, reconnaissance vocale

Lorsque l’on aborde un **contexte** multimodal (Chap. 8), la **fusion** entre flux **audio** et **texte** constitue une application de choix. Dans de nombreux cas, il s’agit de **reconnaissance vocale** ou de **transcription** où le **signal** acoustique, qu’il soit continu (discours, conversation) ou segmenté (clips audio), est mis en correspondance avec des entités textuelles plus ou moins élaborées. Le **Deep Synergy Learning** (DSL) fournit un **Synergistic Connection Network** (SCN) où l’on peut simultanément gérer des entités **audio** et **texte**, et définir une **synergie** qui reflète la **cohérence** entre ces deux modalités.

A. Configuration Générale : Audio + Texte

Le flux **audio** se décompose en segments $\{\mathcal{A}_k\}$, chacun pouvant être caractérisé par des **features** acoustiques $\mathbf{v}_{a,k} \in \mathbb{R}^{d_a}$. Ces descripteurs incluent souvent des coefficients MFCC, un spectrogramme réduit, ou des indices de prosodie (intensité, pitch). Parallèlement, on dispose d’un ensemble d’entités **texte**, par exemple $\{\mathcal{T}_i\}$, qui peuvent être des **mots**, des **tokens** (BERT/GPT) ou des **segments** plus larges (phrases, paragraphes). Chaque entité textuelle se voit associée à un **vecteur** $\mathbf{v}_{t,i} \in \mathbb{R}^{d_t}$, obtenu via un embedding (Word2Vec, GloVe, transformeur) ou via un simple TF-IDF.

Le **SCN** (Synergistic Connection Network) regroupe dans un **même** graphe les entités audio et texte, et définit une pondération $\omega_{(a,k),(t,i)}$ entre le segment audio \mathcal{A}_k et le segment texte \mathcal{T}_i . La **synergie** $S(\mathcal{A}_k, \mathcal{T}_i)$ doit alors rendre compte d’une **compatibilité** audio–texte. Si l’on vise la **reconnaissance vocale**, on souhaite repérer lequel des mots $\{\mathcal{T}_i\}$ s’aligne le mieux avec le segment acoustique \mathcal{A}_k . Dans un cadre plus général, on peut chercher à mesurer la “correspondance sémantique” entre un contenu verbal (texte) et un indice acoustique (ton, style).

B. Cas d’Usage : Transcript et Reconnaissance Vocale

Le cas de la **reconnaissance vocale** ou de la **transcription** se prête particulièrement bien à la logique DSL. Une mise en correspondance classique se présenterait ainsi :

$$\omega_{(a,k),(t,i)}(t+1) = \omega_{(a,k),(t,i)}(t) + \eta [S(\mathcal{A}_k, \mathcal{T}_i) - \tau \omega_{(a,k),(t,i)}(t)].$$

Plus la **similarité** ou la **cohérence** $S(\mathcal{A}_k, \mathcal{T}_i)$ est élevée, plus le lien ω se renforce au fil des itérations.

Dans un système disposant à la fois d’un **flux audio** et d’une **transcription** (manuelle ou partielle), on peut prendre chaque segment \mathcal{A}_k en regard de chaque portion de texte \mathcal{T}_i . La **synergie** $S(\mathcal{A}_k, \mathcal{T}_i)$ repose par exemple sur un **score**

phonémique (ressemblance entre le spectre détecté et la chaîne phonétique du mot) ou sur un **embedding** commun (par exemple, un vecteur acoustique vs. un vecteur BERT), voire les deux :

$$S(\mathcal{A}_k, \mathcal{T}_i) = \alpha \text{sim}_{\text{phon}}(\mathbf{v}_{a,k}, \mathbf{v}_{t,i}) + \beta \text{sim}_{\text{sem}}(\mathbf{v}_{t,i}, \text{contexte lexical}),$$

où sim_{phon} évalue la proximité acoustique (indicateurs de phonèmes) et sim_{sem} juge la cohérence lexicale. Le **DSL** renforce alors $\omega_{(a,k),(t,i)}$ si ce couple audio–texte demeure pertinent, ce qui affine l’alignement final entre la séquence vocale et les mots transcrits.

Dans un pipeline habituel de speech-to-text, le système produit déjà un alignement hypothétique. Un **SCN** multimodal peut raffiner ce couplage par la **dynamique** auto-organisée. Chaque segment audio peut tester plusieurs hypothèses textuelles. Si un couple $(\mathcal{A}_k, \mathcal{T}_i)$ obtient un score $S(\mathcal{A}_k, \mathcal{T}_i)$ supérieur, il sera **renforcé**, tandis que les liaisons concurrentes se verront “repoussées”. On obtient in fine une structure ω où chaque segment audio lie fortement le texte le plus vraisemblable.

C. Bénéfices d’une Approche DSL pour Texte–Audio

Le **DSL** fournit un cadre unique pour gérer la **cohérence** linguistique et acoustique. Contrairement à un pipeline strictement séquentiel, un SCN peut rassembler en un graphe global divers segments $\{\mathcal{A}_k\}$ et $\{\mathcal{T}_i\}$, qu’il peut **associer** ou **désassocier** selon la synergie calculée. Sur le plan **mathématique**, la mise à jour ω se base uniquement sur $S(\mathcal{A}_k, \mathcal{T}_i)$ et un terme d’amortissement $\tau \omega_{(a,k),(t,i)}$.

Plutôt que d’imposer un modèle de Markov caché ou un alignement rigide, le DSL favorise une **résonance** locale de sorte que si un bloc audio correspond au mot \mathcal{T}_i , on renforce $\omega_{(a,k),(t,i)}$. Cela facilite la **correction** de transcriptions si une hypothèse initiale se révèle incohérente (score en baisse) ou la **découverte** de nouveaux mots si l’on introduit de nouveaux segments textuels.

On peut ajouter dynamiquement de nouveaux extraits audio ou de nouveaux tokens textuels (vocabulaire mis à jour). Le SCN recalcule alors la synergie entre chaque couple $(\mathcal{A}_k, \mathcal{T}_i)$ et réitère les mises à jour ω . Un système d’analyse **temps-réel** peut boucler régulièrement, incorporant de plus en plus de segments audio ou lexical, permettant à l’**auto-organisation** de suivre l’évolution de la conversation.

D. Conclusion

Le **couplage** entre le **texte** et l’**audio** illustre parfaitement la philosophie multimodale du **Deep Synergy Learning**. Les entités **audio** $\{\mathcal{A}_k\}$ et **texte** $\{\mathcal{T}_i\}$ vivent dans un **Synergistic Connection Network** où la **synergie** $S(\mathcal{A}_k, \mathcal{T}_i)$ traduit la correspondance acoustico-linguistique. L’**auto-organisation** conduit à un renforcement $\omega_{(a,k),(t,i)}$ pour les couples cohérents (segments acoustiques alignés aux bons mots), permettant :

- Une **amélioration** ou une validation de la **reconnaissance vocale** (transcription plus précise grâce à l’ajustement continu),
- Une **découverte** ou une mise en évidence de **correspondances** texte–audio (un nouveau mot apparaît dans le flux, tel segment audio correspond à tel champ lexical),
- Une possible extension à d’autres canaux (e.g. image + audio + texte) pour une vision multimodale unifiée.

Sur le plan **mathématique**, la même **mise à jour** $\omega \leftarrow \omega + \eta [S - \tau \omega]$ s’applique, dès lors qu’on s’accorde sur une fonction $S(\mathcal{A}_k, \mathcal{T}_i)$ englobant la similarité phonétoco-lexicale, la sémantique du contexte, etc. Cette stratégie unifie la **reconnaissance**, la **correction** et la **fusion** de la parole et du texte en un seul **réseau** évolutif et auto-organisé.

8.4.3.3. Cas d’usage : un SCN où mots, phrases, images s’agglomèrent autour de concepts communs

Il est souvent instructif de disposer d’un **Synergistic Connection Network** (SCN) qui manipule à la fois des **entités textuelles** (mots, phrases) et des **entités visuelles** (images ou extraits d’images). On souhaite observer la formation de **clusters** ou de **macro-nœuds** où, de façon auto-organisée, on regroupe mots et images décrivant le même **concept** ou

la même **idée**. Dans le **Deep Synergy Learning** (DSL), ce phénomène émerge du fait que la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ valorise les associations sémantiquement ou perceptuellement proches, puis la mise à jour ω concrétise ces proximités en liaisons fortes.

A. Structure Générale du SCN

Un **SCN** peut contenir plusieurs catégories d'entités. D'un côté, il y a les **mots** (ex. “chat”, “arbre”, “voler”) et des **phrases** plus étendues (des segments, des énoncés entiers). Chacune de ces entités textuelles $\mathcal{E}_i^{(\text{txt})}$ se voit associer un **vecteur** $\mathbf{v}_i \in \mathbb{R}^{d_t}$, obtenu par Word2Vec, GloVe, BERT ou tout autre embedding neuronal. De l'autre côté, il y a des **images** (ou des parties d'images) $\mathcal{E}_j^{(\text{img})}$, encodées par un CNN ou un ViT, ce qui donne $\mathbf{u}_j \in \mathbb{R}^{d_v}$. Dans un **DSL** multimodal, le **SCN** contient ainsi un ensemble hétérogène de nœuds (texte, image), reliant chaque paire (i, j) par une **pondération** $\omega_{i,j}$.

Pour deux entités de la **même** modalité (texte–texte ou image–image), on peut définir une similarité standard (cosinus, distance inverse). Pour un **couple** texte–image, on recourt à une **fonction** $f(\mathbf{v}_i, \mathbf{u}_j)$ s'apparentant à un alignement cross-modal (ex. un modèle “image to text” appris type CLIP, ou un simple produit scalaire si l'on dispose d'espaces déjà projetés). On obtient :

$$S(\mathcal{E}_i^{(\text{txt})}, \mathcal{E}_j^{(\text{img})}) = f(\mathbf{v}_i, \mathbf{u}_j),$$

puis la **règle** DSL

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

B. Émergence de Concepts Communs

Supposons qu'un **mot** “chat”, une **phrase** “Le chat dort sur un coussin” et une **image** représentant un chat partagent des embeddings proches. Le calcul de $S(\text{“chat”}, \text{image})$ sera élevé, de même que $S(\text{“Le chat dort...”}, \text{image})$. La **dynamique** DSL renforce alors les liaisons $\omega_{(\text{mot=“chat”}), (\text{image})}$ et $\omega_{(\text{phrase=“Le chat...”}), (\text{image})}$. Au fil des itérations, on voit émerger un **micro-cluster** text–image centrant sur le **concept** “chat”. Le **SCN** ne requiert pas qu'on pré-spécifie ce concept car c'est la **haute synergie** qui unifie ces entités dans un nœud macro, aboutissant à un **cluster** text–image.

Dans un **SCN** de grande envergure, de nombreux mots, phrases et images s'organisent en différents **clusters**. Un premier groupe se forme autour de “chat / feline”, un second autour de “chien / dog”, tandis qu'un troisième regroupe “Paris / Tour Eiffel”, associé à des images de la capitale. Cette auto-organisation survient sans supervision : le **DSL** exploite la mise à jour $\omega_{i,j} \leftarrow \omega_{i,j} + \eta[S(i,j) - \tau \omega_{i,j}]$, la synergie s'établissant naturellement selon la proximité sémantique ou perceptuelle.

C. Modélisation Mathématique de la Synergie Multimodale

Pour comparer un mot $\mathbf{v}_i \in \mathbb{R}^{d_t}$ et une image $\mathbf{u}_j \in \mathbb{R}^{d_v}$, on définit :

$$S_{\text{cross}}(\mathbf{v}_i, \mathbf{u}_j) = \cos(\mathbf{W}_{\text{txt}} \mathbf{v}_i, \mathbf{W}_{\text{img}} \mathbf{u}_j),$$

où \mathbf{W}_{txt} et \mathbf{W}_{img} sont des **matrices** (ou des réseaux) projetant dans un espace commun, puis on applique la **similarité cosinus**. Si le score est élevé, on conclut que l'image et le mot décrivent un même concept. On peut se fier à des approches pré-entraînées, telles que CLIP, pour disposer déjà de vecteurs $\mathbf{v}_i, \mathbf{u}_j$ alignés, et recourir au **produit scalaire** direct.

Lorsqu'on compare texte–texte, on choisit S_{txt} ; quand on compare image–image, on dispose de S_{img} . Le **DSL** peut agréger :

$$S(i,j) = \begin{cases} S_{\text{txt}}(\mathbf{v}_i, \mathbf{v}_j), & (\text{texte–texte}), \\ S_{\text{img}}(\mathbf{u}_i, \mathbf{u}_j), & (\text{image–image}), \\ S_{\text{cross}}(\mathbf{v}_i, \mathbf{u}_j), & (\text{texte–image}). \end{cases}$$

Le **SCN** met alors à jour toutes les $\omega_{i,j}$ en conséquence, générant une structure globale où certains arcs correspondent à similitudes text–text et d’autres à cross-modal text–image.

D. Processus d’Auto-Organisation

À chaque **itération**, la formule

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

applique l'**auto-organisation**. Lorsque $S(\mathcal{E}_i, \mathcal{E}_j)$ est élevé de façon récurrente, $\omega_{i,j}$ grandit, traduisant un lien stable au sein du **SCN**. D’un point de vue **analytique**, on peut prouver que si η et τ restent constants et S fixe, on tend vers un état stationnaire où

$$\omega_{i,j}^* = \frac{S(i,j)}{\tau},$$

pour chaque couple (i,j) . Dans les cas plus complexes, l’existence de termes d’inhibition ou la variabilité de S suscite des bifurcations, mais la logique de regroupement demeure.

Si plusieurs mots, phrases et images sont tous fortement interconnectés (liens ω élevés), ils forment un **macro-nœud** (chap. 6). Cette entité plus large représente un **concept** commun (“sports de balle”, “animaux de ferme”, “vacances estivales”), que le DSL identifie sans supervision explicite. C’est la **philosophie d’auto-organisation** où la **dynamique** localisée sur ω produit spontanément une hiérarchie de regroupements text–image.

E. Gains Opérationnels

Un **SCN** réunissant les **mots, phrases et images** autour de **concepts** facilite des applications telles que la **recherche multimodale** (trouver une image via un mot), le **tag** automatique d’images par le vocabulaire textuel, ou l’**annotation** textuelle (proposer des légendes). La **mise à jour** ω confère à ce système une capacité d’évolution où de nouveaux mots ou images peuvent s’intégrer aux clusters existants dès lors qu’ils présentent une synergie suffisante.

Conclusion

Un **SCN** où **mots, phrases et images** cohabitent illustre une **fusion** multimodale de type texte–image. Chaque entité (mot, phrase, image) possède un **embedding** qui permet de calculer la **synergie** $S(\cdot, \cdot)$. L'**auto-organisation** DSL engendre naturellement des **clusters** (ou macro-nœuds) centrés sur des **concepts communs**, tels que “chat”, “voiture”, “voyage”, etc. Les liens ω se renforcent pour les paires (mot=“cat”, image=chat) ou (phrase=“voiture rouge...”, image=voiture), produisant un **réseau** où coexistent des arcs text–text, image–image, et text–image. D’un point de vue **mathématique**, ce n’est que la mise en œuvre locale de la règle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$. Les résultats mettent en évidence l’émergence de **macro-concepts**, où des nœuds textuels et visuels convergent spontanément autour de thèmes unificateurs sans nécessiter de supervision. Cette formation découle directement de la **dynamique** DSL, qui exploite $S(\cdot, \cdot)$ comme critère de synergie.

8.5. Modélisation des Entités Sonores

Dans un contexte **multimodal**, la dimension **audio** occupe une place cruciale. Elle renseigne sur les **caractéristiques** d'un signal acoustique (voix, musique, bruitages, etc.) et peut s'intégrer au **SCN** (Synergistic Connection Network) en tant qu'entités d'information. Pour que le **DSL** (Deep Synergy Learning) puisse exploiter la **synergie** entre sources audio et autres modalités (vision, texte, etc.), il est indispensable de **représenter** chaque segment audio de façon adéquate.

8.5.1. Représentations Audio

Le choix de la **représentation** (features ou embeddings) appliquée à une **trame** (ou un bloc) de signal sonore conditionne grandement la **qualité** du calcul de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$. On peut distinguer les **features classiques** (MFCC, spectrogrammes) des **embeddings profonds** plus récents (Audio2Vec, etc.).

8.5.1.1. Features classiques (MFCC, spectrogrammes) vs. embeddings profonds (Audio2Vec)

Il existe plusieurs **méthodes** pour représenter un **signal audio** sous forme de **vecteurs** ou de **descripteurs** exploitables par le **Deep Synergy Learning (DSL)** dans le cadre d'un **Synergistic Connection Network (SCN)**. Certains choix relèvent d'**approches classiques**, telles que les **MFCC** ou les **spectrogrammes**, tandis que d'autres recourent à des **embeddings profonds** (ex. Audio2Vec). Chaque famille de techniques possède ses avantages et contraintes, influençant la **synergie** calculée entre segments audio et la structure finale du réseau.

A. Caractéristiques Classiques

Les **MFCC** constituent une représentation historique et largement utilisée en reconnaissance de parole et analyse audio. Le signal $\mathbf{x}(t)$ est découpé en trames temporelles de durée courte (souvent 20 à 40 ms). Pour chaque trame, on calcule un **spectre** à l'aide de la transformée de Fourier discrète :

$$X(\omega) = \sum_{\ell=0}^{L-1} x(\ell) e^{-i\omega\ell},$$

où L représente le nombre d'échantillons dans la trame et ω la variable fréquentielle. On projette ensuite ce spectre sur une **échelle Mel** (partition non linéaire de la bande de fréquences), puis on applique un log de la puissance et, enfin, une **Discrete Cosine Transform (DCT)** pour obtenir les **coefficients cepstraux** \mathbf{c} . En pratique, on retient souvent les 13 premiers coefficients, éventuellement complétés de leurs dérivées ($\Delta\mathbf{c}, \Delta^2\mathbf{c}$). Sur le plan mathématique, la représentation MFCC $\mathbf{c} \in \mathbb{R}^{13}$ condense la forme globale du spectre de parole en tenant compte de l'échelle psychoacoustique Mel. Le principal avantage se situe dans la **compacité** (très faible dimension) et la robustesse en reconnaissance de parole. L'inconvénient est une capacité plus limitée à représenter des nuances complexes (timbres musicaux, émotions, etc.).

Les **spectrogrammes** constituent une approche plus détaillée. Le signal $\mathbf{x}(t)$ est segmenté en fenêtres glissantes, et pour chacune, on calcule la **Short-Time Fourier Transform (STFT)**. On obtient ainsi une matrice $\mathbf{S}(t, f)$ représentant l'amplitude (ou la puissance) en fonction du temps et de la fréquence :

$$\mathbf{S}(t, f) \approx \left| \sum_{\ell} x(\ell) w(\ell - t) e^{-2i\pi f \ell} \right|,$$

où $w(\ell - t)$ désigne une fonction fenêtre localisant la portion temporelle. Sur le plan **mathématique**, cela donne un tableau 2D plus volumineux que les MFCC, mais contenant davantage d'information sur les harmoniques et la structure fréquentielle. Les similarités entre spectrogrammes de segments audio peuvent se calculer par cosinus, distance euclidienne ou kernels RBF. Le principal intérêt réside dans la **richesse** de la représentation ; l'inconvénient tient à la **dimension** plus élevée et au coût de calcul dans un **SCN** volumineux.

B. Embeddings Profonds (Audio2Vec, etc.)

Certaines méthodes s'inspirent du succès de **Word2Vec** dans le monde textuel pour proposer des **embeddings** spécifiques à l'audio. L'objectif est d'apprendre un **réseau** Φ_θ qui, pour tout segment audio \mathbf{x} , produit un vecteur $\mathbf{z} = \Phi_\theta(\mathbf{x}) \in \mathbb{R}^d$. L'apprentissage peut se faire par supervision (classer des locuteurs, reconnaître des mots-clés) ou par approches auto-supervisées (contraste entre segments positifs/négatifs). Une fois entraîné, ce **réseau** Φ fournit un **embedding** audio plus abstrait, capturant des **patterns** variés (intonation, spectre, timing). Mathématiquement, la représentation finale :

$$\mathbf{z} = \Phi_\theta(\mathbf{x}) \in \mathbb{R}^d$$

s'obtient en passant l'onde audio ou son spectrogramme dans un CNN, un RNN ou un transformeur audio. Les **embeddings profonds** (type Audio2Vec) améliorent la généralisation et la robustesse, car ils s'appuient sur des couches profondes. Cependant, ils exigent souvent un **dataset** conséquent et un **entraînement** onéreux (réseau large, multiples époques).

C. Forme mathématique d'un embedding audio

Quelle que soit la méthode, on aboutit finalement à un **vecteur** $\mathbf{z}_i \in \mathbb{R}^d$ pour l'entité audio \mathcal{E}_i . Dans un cadre plus formel, on pose

$$\Phi_\theta(\mathbf{x}) = \mathbf{z} \in \mathbb{R}^d,$$

avec θ ajusté par minimisation d'une perte

$$\mathcal{L}(\theta) = \sum_{\mathbf{x}, \mathbf{y}} \ell(\Phi_\theta(\mathbf{x}), \mathbf{y}).$$

L'apprentissage détermine θ pour classifier ou regrouper des signaux. Une fois θ fixés, le vecteur \mathbf{z} devient la **représentation** sub-symbolique du segment audio.

D. Implication pour le SCN

Dans un **SCN**, le **Deep Synergy Learning** évalue la **distance** (ou la **similarité**) entre deux vecteurs \mathbf{z}_i et \mathbf{z}_j . On peut définir

$$S(i, j) = \exp(-\alpha \|\mathbf{z}_i - \mathbf{z}_j\|^2) \quad \text{ou} \quad \frac{\mathbf{z}_i \cdot \mathbf{z}_j}{\|\mathbf{z}_i\| \|\mathbf{z}_j\|},$$

selon l'option RBF (Gaussienne) ou la similarité cosinus. Cette **synergie** gouverne la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)].$$

Les segments audio s'**agglomèrent** en **clusters** s'ils présentent des vecteurs \mathbf{z}_i voisins. Cette **auto-organisation** acoustique peut également s'étendre à d'autres modalités (texte, image), ce qui fonde la multi-modalité (voir Chap. 8.5.2, 8.5.3).

E. Conclusion

Le **choix** entre **MFCC**, **spectrogrammes** ou **embeddings** profonds (type Audio2Vec) se fait en fonction des **ressources** (dataset, puissance de calcul), de la **granularité** requise (besoin d'informations harmoniques détaillées ou non) et de la **finalité** (reconnaissance de locuteurs, classification musicale, etc.). **Mathématiquement**, toutes ces approches débouchent sur des **vecteurs** \mathbf{z}_i représentant l'entité audio \mathcal{E}_i . Le **DSL** se contente alors de définir la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ par une fonction de distance ou de similarité sur $\mathbf{z}_i, \mathbf{z}_j$. Les **clusters** acoustiques ou multi-modaux obtenus dans le **SCN** reflètent la cohérence audio entre segments, alors que les considérations plus détaillées (MFCC vs. spectro vs. embedding) relèvent d'un compromis entre simplicité, expressivité et coût de mise en œuvre.

8.5.1.2. Normalisation : amplitude, échelle log, etc.

Lorsque l'on souhaite traiter plusieurs **modalités** (audio, vision, texte, capteurs, etc.) ou simplement différentes **sources** de données (descripteurs MFCC vs. spectrogrammes, embeddings, signaux bruts), il s'avère souvent indispensable de recourir à des **techniques de normalisation**. L'objectif est de **mettre à l'échelle** les variables ou les vecteurs de représentation, afin d'éviter qu'une modalité ne "domine" artificiellement les calculs de **synergie** $S(i, j)$ dans le **Deep Synergy Learning** (DSL).

A. Normalisation d'Amplitude

Les données collectées, qu'elles proviennent de l'audio (amplitude ou puissance spectrale), du texte (fréquences de mots, valeurs TF-IDF), ou d'autres modalités, peuvent se situer sur des échelles très diverses. On peut alors se retrouver avec des **vecteurs** \mathbf{x}_i dont la norme $\|\mathbf{x}_i\|$ est très grande (par exemple, une intensité audio ou un histogramme massif), tandis qu'un autre vecteur \mathbf{x}_j possède une norme plus réduite. Pour éviter que l'on compare des grandeurs incomparables, on pratique une **division** ou une **mise à l'échelle** afin de les **ramener** à des amplitudes similaires.

Une forme simple est la **mise à l'échelle min–max**, définie composante par composante :

$$\mathbf{x}'_i = \frac{\mathbf{x}_i - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}$$

ce qui ramène les valeurs dans l'intervalle [0,1]. Pour des représentations vectorielles plus classiques, on recourt souvent à la **normalisation** ℓ_2 :

$$\mathbf{x}'_i = \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|}.$$

Cette dernière est très prisée lorsque la **synergie** se base sur la similarité cosinus, puisqu'on ne regarde alors plus que l'**angle** entre deux vecteurs, et pas leur taille.

La normalisation d'amplitude garantit que la **magnitude** de chaque vecteur reste bornée. Dans un **SCN** multimodal, cela évite qu'une modalité, dont les valeurs numériques seraient élevées, impose systématiquement une **forte** synergie malgré l'absence de vrai lien sémantique ou perceptuel. En définissant, par exemple, $\mathbf{x}'_i = \mathbf{x}_i / \|\mathbf{x}_i\|$, on homogénéise les directions, ce qui rend les similarités plus interprétables.

B. Échelle Logarithmique (Rescaling log)

Il arrive que certaines modalités (certains signaux) varient sur des **ordres de grandeur** très différents, comme 1, 10, 1000. Dans ce contexte, un simple min–max ou division par la norme ne suffit pas toujours. Une **transformation logarithmique** atténue les valeurs extrêmes, ce qui est souvent pertinent pour des grandeurs exponentielles (ex. volume sonore, intensité lumineuse, fréquence de mots en TF-IDF). En pratique, on définit :

$$x'_i = \log(1 + \alpha x_i),$$

avec $\alpha > 0$ choisie pour adapter l'échelle.

Si l'on note \mathbf{x}_i un vecteur de valeurs non négatives, la version "log-scale" s'écrit :

$$\mathbf{x}'_i = \log(1 + \alpha \mathbf{x}_i).$$

Cela **réduit** considérablement l'influence d'un vecteur très large sur la similarité si son écart n'est qu'exponentiel.

En ramenant les valeurs par le **log**, on privilégie les **rapports** plutôt que les **différences** absolues. Dans le **Deep Synergy Learning**, cela revient à ce que la **synergie** S se comporte de façon plus stable quand il y a des valeurs très distinctes. C'est un choix adapté aux distributions sur de multiples échelles (comme l'audio intensité, ou les histogrammes de fréquences pour certains tokens rares/fréquents).

C. Normalisation par Statistiques Globales

On peut aussi opter pour une **standardisation** classique par la moyenne μ et l'écart-type σ . Cela consiste à recentrer et réduire les valeurs :

$$\mathbf{x}'_i = \frac{\mathbf{x}_i - \mu}{\sigma}.$$

Lorsque l'on compare différents flux (audio, vision, texte), on peut effectuer cette opération flux par flux, pour s'assurer qu'ils possèdent tous une moyenne proche de 0 et un écart-type proche de 1.

La standardisation basée sur la médiane et la MAD (Median Absolute Deviation) s'emploie pour minimiser l'effet des outliers. On substitue à la place de μ, σ la **médiane** et la médiane des écarts absolu, puis on ramène \mathbf{x}_i à des valeurs autour de 0 avec une échelle unitaire. Cette méthode confère une plus grande **robustesse** lorsque quelques valeurs extrêmes pourraient biaiser les moyennes.

D. Approche Hybride ou Multi-Pass

Il est possible de combiner plusieurs **stratégies** dans un même SCN multimodal. Par exemple, on peut :

- Prendre le **log** des mesures sur une dimension (ex. intensité audio) pour compacter l'amplitude,
- Appliquer un **min–max** ou une **normalisation** ℓ_2 aux données visuelles,
- Standardiser (moyenne 0, variance 1) les valeurs textuelles issues d'un comptage ou d'un vecteur TF-IDF.

Ensuite, on **concatène** ou on confronte ces vecteurs dans une synergie $S(i, j)$. Sur le plan **mathématique**, la normalisation assure que chaque flux ou dimension contribue de façon plus équilibrée.

E. Implication Mathématique dans le Calcul de Synergie

Lorsque plusieurs modalités coexistent, la normalisation unifie leur amplitude et permet un calcul de $S(i, j)$ plus fidèle, sans qu'un flux à grande échelle ne "domine". Dans le **DSL**, la mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$ demeure plus cohérente si les valeurs de $S(i, j)$ ne sont pas saturées par des différences d'échelle.

Des signaux numériquement plus stables évitent à la dynamique de l'**auto-organisation** de s'emballer ou de converger trop lentement. Sur le plan **mathématique**, cela réduit les gradients ou incrément exagérément grands, ce qui peut accélérer la **convergence**.

Conclusion

La **normalisation** (amplitude, log-scale, standardisation) est une étape **centrale** dans un **SCN** multimodal, afin que chaque modalité ou dimension soit comparée sur une **base d'échelle** équitable. Les principales méthodes incluent :

- **Mise à l'échelle min–max** ou division par la **norme** ℓ_2 ,
- **Transformation** logarithmique pour gérer les valeurs sur plusieurs ordres de grandeur,
- **Standardisation** (Z-score) ou **statistiques** robustes (MAD).

Sur le plan **mathématique**, la normalisation homogénéise les vecteurs $\mathbf{x}_i, \mathbf{x}_j$ avant le calcul de la **synergie** $S(i, j)$. Cela fluidifie la **dynamique** $\omega_{i,j}(t)$ en évitant qu'un flux n'impose ses amplitudes, permettant ainsi aux **clusters** émergents dans le **DSL** de mieux traduire la **réalité** des similarités inter-entités.

8.5.1.3. Traitement de séquences ou de “frames” audio

Un **système** multimodal intégrant un **flux audio** se confronte à la question de la **nature** du signal acoustique. L’information auditive, contrairement à une image statique ou un bloc de texte, se déploie dans le temps. Il est alors fréquent de découper ce signal en **trames** (frames) successives, chacune relativement courte (ex. 10 ms, 20 ms), afin de représenter plus finement la **dynamique** temporelle. Le **Deep Synergy Learning** (DSL) appliqué au **Synergistic Connection Network** (SCN) prend alors en charge un ensemble d’entités \mathcal{E}_m indexées par le temps (m) et veille à **auto-organiser** ces trames ou séquences selon leur cohérence spectrale, phonétique ou prosodique.

A. Segmentation Audio en Frames : Principes et Modèle Mathématique

Le **signal** audio continu $x(t)$ est classiquement découpé en trames temporelles de longueur L , possiblement chevauchantes (overlap). Chaque trame $\mathbf{a}^{(m)} \in \mathbb{R}^d$ correspond, par exemple, à un vecteur MFCC, un spectre ou un embedding audio (chap. 8.5.1.1). Les index $m = 1, 2, \dots$ parcourront ainsi les portions du signal à des intervalles réguliers Δ . D’un point de vue **mathématique**, on peut noter :

$$\mathbf{a}^{(m)} = \text{Transform}(\{x(t)\}_{t=m\cdot\Delta}^{(m+1)\cdot\Delta}),$$

où Transform décrit l’extraction de features (DFT, MFCC, etc.). Chaque $\mathbf{a}^{(m)}$ devient alors une entité \mathcal{E}_m dans le SCN.

Une fois les frames $\mathbf{a}^{(m)}$ calculées, on les intègre dans le **Synergistic Connection Network**. Chaque frame \mathcal{E}_m possédant un vecteur $\mathbf{a}^{(m)} \in \mathbb{R}^d$, la **synergie** $S(\mathcal{E}_m, \mathcal{E}_{m'})$ peut être définie, par exemple, via une distance ou un kernel :

$$S(\mathbf{a}^{(m)}, \mathbf{a}^{(m')}) = \exp(-\alpha \|\mathbf{a}^{(m)} - \mathbf{a}^{(m')}\|^2) \quad (\text{RBF kernel}),$$

ou encore une **similarité cosinus**, $\frac{\mathbf{a}^{(m)} \cdot \mathbf{a}^{(m')}}{\|\mathbf{a}^{(m)}\| \|\mathbf{a}^{(m')}\|}$.

Si la durée globale est T et la longueur d’une trame Δ , on obtient approximativement $\frac{T}{\Delta}$ frames. D’un point de vue **mathématique**, un SCN naïf aurait alors $O(n^2)$ liens potentiels, où $n = \frac{T}{\Delta}$. Pour des enregistrements longs ou un pas Δ très petit, il devient nécessaire de **sparsifier** (Chap. 7.2.3) en limitant les connexions aux trames voisines en temps ou présentant une forte ressemblance.

B. Cohérence Temporelle et Séquences de Frames

Le **DSL** tient compte de la **cohérence** entre frames adjacentes $\mathbf{a}^{(m)}, \mathbf{a}^{(m+1)}$. Dans la plupart des signaux audio, deux trames successives sont similaires, surtout si elles appartiennent au même phonème ou au même effet sonore. Par la **règle** DSL :

$$\omega_{m,m+1}(t+1) = \omega_{m,m+1}(t) + \eta [S(\mathbf{a}^{(m)}, \mathbf{a}^{(m+1)}) - \tau \omega_{m,m+1}(t)],$$

on s’attend à ce que $\omega_{m,m+1}$ devienne élevé lorsque la transition est fluide (même son, même tonalité).

Pour la **parole**, un phonème s’étend sur quelques frames. La similarité demeure élevée dans ce segment, puis chute à la frontière. Le **SCN** regroupe alors les frames cohérentes en un **cluster** local, identifiant potentiellement un segment acoustique “unifié” (voyelle, consonne, etc.). Cette identification n’est pas imposée, mais **émerge** de la dynamique ω .

C. Mécanisme d’Auto-Organisation pour la Séquence Audio

Chaque paire (m, m') (frames indices m, m') voit sa liaison $\omega_{m,m'}(t)$ évoluer selon :

$$\omega_{m,m'}(t+1) = \omega_{m,m'}(t) + \eta [S(\mathbf{a}^{(m)}, \mathbf{a}^{(m')}) - \tau \omega_{m,m'}(t)].$$

Les liaisons inter-frames s’en trouvent **renforcées** si la ressemblance audio (MFCC, spectre, embedding) est réelle, et s’**amenuisent** dans le cas contraire.

Au fur et à mesure des itérations, il se forme des **clusters** de frames contigües ou semblables, traduisant la persistance d'un même phonème ou d'une même note musicale. La **dynamique** DSL fait ressortir la discontinuité là où le son change notablement (d'un phonème [a] à [i], ou d'un accord musical à un autre).

D. Avantages et Aspects Mathématiques

Le **DSL** évite de figer une segmentation en unités phonétiques ou syllabiques. La *force* de $\omega_{m,m}$, reflète naturellement les proximités spectrales. D'un point de vue **analytique**, on se rapproche d'un **clustering** sans paramètre figé, où la continuité est "décidée" localement par le niveau de similarité.

Si on ajoute une **seconde modalité** (par ex. **vidéo** synchronisée), le SCN peut relier la frame audio $\mathbf{a}^{(m)}$ à la frame vidéo $\mathbf{v}^{(m)}$ si une synergie inter-modale est définie. On récupère alors la **cohérence audio-visuelle**, détectant par exemple un mouvement labial correspondant au son produit (voir 8.3.3.2).

La complexité $O(n^2)$ avec $n = \frac{T}{\Delta}$ demeure un défi. On peut, pour réduire les comparaisons inutiles, s'en tenir à un voisinage temporel $|m - m'| \leq w$ ou à un k-NN sur les embeddings audio, aboutissant à une structure plus épars et un calcul de synergie plus sélectif.

Conclusion

Dans un **SCN** appliqué au **traitement de séquences** ou de "frames" audio, chaque segment temporel $\mathbf{a}^{(m)}$ devient une **entité**. Sur un plan **mathématique**, la **synergie** $S(\mathbf{a}^{(m)}, \mathbf{a}^{(m')})$ rend compte de la ressemblance spectrale ou phonétique. Grâce à la **dynamique** DSL :

- Les frames se **connectent** ou se désolidarisent selon leur similarité,
- Des **clusters** se forment, correspondant à des segments audio quasi stables (même phonème, même son),
- L'**analyse** se fait en continu, sans imposer de segmentation rigide.

Cette approche favorise une **auto-organisation** fine du flux audio, pouvant s'étendre à de la **synchronisation** multimodale (audio–vidéo, audio–texte). Le DSL agit alors comme un **pivot** pour découvrir, segmenter ou grouper des frames audio, exploitant la **synergie** des signaux de façon non supervisée.

8.5.2. Synergie Audio-Audio

Dans le contexte **multimodal**, l'analyse **audio-audio** consiste à évaluer la **synergie** entre deux **flux sonores** ou deux *segments* d'enregistrement. L'objectif est de déterminer leur **similarité** ou leur **complémentarité** (cf. chap. 8.1) pour, par exemple, fusionner des signaux cohérents (mêmes sources, mêmes événements sonores), ou au contraire inhiber ceux qui apportent du bruit ou des informations redondantes.

8.5.2.1. Similarité de signatures sonores (distance euclidienne, cosinus)

Dans un **SCN** (Synergistic Connection Network) dédié au traitement ou à la fusion de flux **audio**, la notion de "signature sonore" d'un segment (ou d'un flux complet) s'avère déterminante. Chaque segment \mathcal{A}_i est associé à un **vecteur** $\mathbf{x}_i \in \mathbb{R}^d$, résultant par exemple d'une extraction MFCC ou d'un **embedding** audio. Le **Deep Synergy Learning** (DSL) doit ensuite définir une **synergie** $S(\mathcal{A}_i, \mathcal{A}_j)$ reflétant la proximité ou la similarité entre deux de ces représentations. Les mesures les plus courantes pour comparer deux vecteurs \mathbf{x}_i et \mathbf{x}_j incluent la **distance euclidienne** et la **similarité cosinus**.

A. Représentation Vecteur du Segment Audio

Chaque segment \mathcal{A}_i correspond à un bloc temporel ou un ensemble de frames audio (voir chap. 8.5.1.3). On peut construire un vecteur $\mathbf{x}_i \in \mathbb{R}^d$ en :

- Effectuant la **moyenne** ou la **concaténation** de MFCC sur plusieurs frames,
- Extrayant un **patch** du spectrogramme qu'on linéarise ou qu'on résume par un CNN,
- Employant un **embedding** profond (type Audio2Vec, autoencodeur, etc.).

D'un point de vue **mathématique**, le vecteur \mathbf{x}_i est donc l'unité essentielle pour caractériser la "signature" d'un segment audio \mathcal{A}_i .

Une fois ces vecteurs audio $\{\mathbf{x}_i\}$ définis, la **synergie** $S(\mathcal{A}_i, \mathcal{A}_j)$ se fonde sur une **distance** ou une **similarité**. Le DSL utilisera cette valeur pour décider, via la mise à jour $\omega_{i,j} \leftarrow \omega_{i,j} + \eta[S(i,j) - \tau \omega_{i,j}]$, si deux segments audios se connectent fortement ou non dans le **SCN**.

B. Définition d'une Distance ou d'une Similarité

Une approche directe consiste à employer la **distance** L^2 entre les deux vecteurs \mathbf{x}_i et \mathbf{x}_j . On définit :

$$d_{ij}^{\text{eucl}} = \|\mathbf{x}_i - \mathbf{x}_j\|_2 = \sqrt{\sum_{k=1}^d (x_{i,k} - x_{j,k})^2}.$$

Puis on convertit éventuellement cette distance en une **similarité** $S(i,j)$:

$$S(i,j) = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|^2) \quad \text{ou} \quad \frac{1}{1 + \|\mathbf{x}_i - \mathbf{x}_j\|^2}.$$

Une distance $\|\mathbf{x}_i - \mathbf{x}_j\|$ faible implique $S(i,j)$ grand, signifiant une **synergie** élevée entre les segments audio correspondants.

D'autres travaux préfèrent la **similarité cosinus** :

$$\cos(\mathbf{x}_i, \mathbf{x}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

Cette mesure met en avant l'**angle** entre \mathbf{x}_i et \mathbf{x}_j plutôt que leur écart en norme. Pour des signaux audio qui peuvent différer fortement en amplitude, la cosinus-sim fournit un indice de "forme" spectrale ou "orientation" dans l'espace des features, indépendamment du volume.

Dans le **Deep Synergy Learning**, si on emploie la cosinus-sim, on peut poser :

$$S(\mathcal{A}_i, \mathcal{A}_j) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|}.$$

Les segments ayant des signatures sonores analogues obtiennent un $S(\cdot, \cdot)$ proche de 1. La pondération $\omega_{i,j}$ grandit donc si la synergie $S(i,j)$ demeure > 0 , fidélisant la liaison entre ces deux entités audio et aboutissant à leur regroupement dans le **SCN**.

C. Considérations Mathématiques

Si on choisit la **distance euclidienne**, il peut s'avérer crucial de **normaliser** ou de **standardiser** les vecteurs \mathbf{x}_i (voir chap. 8.5.1.2). Une grande amplitude due à un enregistrement plus "fort" ou plus "long" ne doit pas se traduire en une distance faussée. Par contraste, la cosinus-sim s'accorde mieux des variations d'amplitude.

Le vecteur $\mathbf{x}_i \in \mathbb{R}^d$ peut être de taille réduite (ex. 13–39 MFCC) ou plus volumineux (embedding de 128 ou 512). Le **coût** de calcul pour comparer tous les couples (i, j) est alors $O(n^2 d)$. Dans un SCN volumineux, on recourt souvent à des techniques de **sparsification** (k-NN, etc.) pour limiter la croissance quadratique.

Une fois la **similarité** cos ou la distance RBF choisie, la **règle** DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

suit son cours. Si la ressemblance $S(i,j)$ dépasse un seuil, $\omega_{i,j}$ est renforcé, traduisant un **cluster** audio–audio se consolidant.

D. Applications

Lorsque plusieurs canaux enregistrent la même source sonore, ou lorsque des segments audio récurrents apparaissent, la **similarité** (distance faible, cosinus élevé) conduit le DSL à renforcer ω . On agrège ces segments, aboutissant à des super-nœuds représentant, par exemple, la même **voix**, le même **instrument** ou la même **séquence musicale**.

Dans le cadre d'une base de segments audio, la mise à jour DSL engendre des **clusters** d'entités audio partageant un timbre, un locuteur, ou une signature spectrale caractéristique. L'utilisateur peut ensuite repérer ces clusters (ex. “Segment audio 12 est relié aux segments 17, 45, 89… relevant du même événement sonore”).

Conclusion

Comparer deux segments ou flux audio $\mathcal{A}_i, \mathcal{A}_j$ dans un **SCN** repose sur une **mesure** $S(\mathcal{A}_i, \mathcal{A}_j)$ dérivée d'une distance euclidienne ou d'une similarité cosinus calculée dans un **espace de features** \mathbb{R}^d . Sur le plan **mathématique**, ce peut être :

- $d_{ij}^{\text{eucl}} = \|\mathbf{x}_i - \mathbf{x}_j\|$ transformé en $\exp(-\alpha d_{ij}^2)$,
- $\cos(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \cdot \mathbf{x}_j / \|\mathbf{x}_i\| \|\mathbf{x}_j\|$.

Le **DSL** se contente d'insérer $S(i,j)$ dans sa mise à jour $\omega_{i,j} \leftarrow \omega_{i,j} + \eta[S(i,j) - \tau \omega_{i,j}]$. Les segments $\mathbf{x}_i, \mathbf{x}_j$ se voient **liés** (pondération ω élevée) s'ils correspondent à des signatures sonores proches, révélant ainsi leur **cohérence** acoustique et formant des **clusters** qui aident à l'analyse et la recherche audio.

8.5.2.2. Détection d'Événements Communs (2 flux audio captant la même source ?)

Il est fréquent de disposer de **plusieurs flux audio** enregistrés en parallèle (plusieurs microphones, ou plusieurs canaux stéréo/5.1) qui couvrent la même scène ou le même espace sonore. L'un des objectifs est alors de **déteindre** si un **événement particulier** (une voix, un objet qui chute, un claquement) se produit et est enregistré **simultanément** par ces canaux. Le **Deep Synergy Learning** (DSL), au sein d'un **Synergistic Connection Network** (SCN), permet de modéliser un tel cas en définissant une **synergie** entre les trames (ou segments) de chaque flux, puis en laissant la dynamique ω révéler les correspondances.

A. Contexte : Fusion Audio dans le DSL

On se place dans la situation où l'on dispose de deux enregistrements :

$$\mathcal{A}_1 = \{\mathcal{E}_i^{(1)}\}_{i \in I_1}, \quad \mathcal{A}_2 = \{\mathcal{E}_j^{(2)}\}_{j \in I_2},$$

chaque $\mathcal{E}_i^{(1)}$ et $\mathcal{E}_j^{(2)}$ étant un segment ou une trame dans le temps (par exemple 20 ms de signal). Les flux peuvent être synchronisés temporellement (positions de micro différentes) ou légèrement décalés. L'idée est de comparer ces trames pour y repérer des “pics” de similarité au même instant.

Pour associer la trame $\mathcal{E}_i^{(1)}$ au flux 1 et la trame $\mathcal{E}_j^{(2)}$ au flux 2, on calcule une **similarité** ou un **score** $S(i,j)$ reflétant la correspondance acoustique :

$$S(\mathcal{E}_i^{(1)}, \mathcal{E}_j^{(2)}) = \rho(\mathbf{x}_i^{(1)}, \mathbf{x}_j^{(2)}),$$

où $\mathbf{x}_i^{(1)}$ et $\mathbf{x}_j^{(2)}$ sont, par exemple, des vecteurs MFCC, ou des vecteurs de spectrogramme, ou encore un embedding. La fonction ρ peut être une **similarité cosinus**, une **distance** inversée, ou même un **coefficient** de corrélation (Pearson) pour capturer la ressemblance temporelle.

Si la **dynamique DSL renforce** une liaison $\omega_{(i,j)}$ entre la trame $\mathcal{E}_i^{(1)}$ et $\mathcal{E}_j^{(2)}$, cela indique qu'ils "entendent" le **même** événement, un même son capté par les deux micros. Cette liaison devient un **témoin** d'un événement commun, éventuellement mis en évidence par un **seuil** θ ou par l'analyse de clusters dans le SCN.

B. Mise en Forme Mathématique

Dans le SCN, chaque **trame** $\mathcal{E}_i^{(1)}$ du flux 1 peut se lier à chaque trame $\mathcal{E}_j^{(2)}$ du flux 2. On définit donc une liaison $\omega_{(i,j)}(t)$. La **mise à jour** DSL s'écrit :

$$\omega_{(i,j)}(t+1) = \omega_{(i,j)}(t) + \eta[S(\mathcal{E}_i^{(1)}, \mathcal{E}_j^{(2)}) - \tau \omega_{(i,j)}(t)].$$

Une synergie $S(\cdot, \cdot)$ suffisamment élevée fait **croître** la pondération $\omega_{(i,j)}$. Sur le plan **analytique**, on aboutit (en régime stationnaire) à $\omega_{(i,j)} \approx S(i,j)/\tau$ si rien ne perturbe la dynamique.

Pour "comparer" utilement les segments $\mathcal{E}_i^{(1)}$ et $\mathcal{E}_j^{(2)}$, on peut exiger qu'ils se situent à peu près au **même** instant ($i \approx j$ si les flux sont synchronisés) ou introduire un paramètre de décalage δ . Cela évite de connecter $\mathcal{E}_i^{(1)}$ (temps t) à $\mathcal{E}_j^{(2)}$ (temps t') s'ils ne se chevauchent pas dans la dimension temporelle.

Une fois la **mise à jour** itérative de ω stabilisée, on peut poser un **seuil** afin que si $\omega_{(i,j)}(t)$ dépasse θ , on déclare qu'un **événement** est simultanément perçu dans les deux flux sur les trames $\mathcal{E}_i^{(1)}$ et $\mathcal{E}_j^{(2)}$. D'un point de vue **mathématique**, la formation d'un **cluster** regroupant $\{(i_1, j_1), (i_2, j_2), \dots\}$ rend compte d'un événement plus large ou plus prolongé.

C. Critères de Similarité Audio

Le plus simple est de comparer des vecteurs $\mathbf{x}_i^{(1)}$ et $\mathbf{x}_j^{(2)}$ (MFCC, spectral, embedding) via un **produit scalaire** (cosinus) :

$$S(\mathcal{E}_i^{(1)}, \mathcal{E}_j^{(2)}) = \frac{\mathbf{x}_i^{(1)} \cdot \mathbf{x}_j^{(2)}}{\|\mathbf{x}_i^{(1)}\| \|\mathbf{x}_j^{(2)}\|}.$$

Si ce score est élevé, on conclut que ces trames ont un contenu sonore proche. On peut aussi employer la **distance euclidienne**, ou un **coefficient** de corrélation de Pearson, notamment si la phase du signal (ou l'enveloppe) importe.

Dans le cas de bruits impulsifs ou de signaux brefs, on peut réaliser une **cross-corrélation** pour repérer le meilleur décalage temporel Δt . Une forte valeur de cette cross-corrélation indique que les deux segments captent la même source (ex. un claquement de mains arrivé au micro 1 puis au micro 2 avec un léger retard). On peut alors convertir la valeur de pic en un score $S(i,j)$.

D. Synchronisation et Structure du SCN

Dans le **DSL**, si un même son se produit, la synergie $S(i,j)$ devient forte pour les segments $\mathcal{E}_i^{(1)}$ et $\mathcal{E}_j^{(2)}$ englobant cet instant. La mise à jour $\omega_{(i,j)}(t+1) = \omega_{(i,j)}(t) + \eta[S(i,j) - \tau \omega_{(i,j)}(t)]$ amplifie la liaison $\omega_{(i,j)}$. On peut alors identifier le "cluster" de liens forts reliant flux 1 et flux 2 aux mêmes instants. Cela met en évidence l'existence et la durée de l'événement (une voix, une percussive...).

Cette logique s'étend si on possède plus de deux flux (ex. 3 ou 4 micros). Le **SCN** englobe alors de multiples couples $\omega_{(i,j,k,\dots)}$ ou s'appuie sur un graphe plus riche. Le concept demeure, la **synergy** S s'appuie sur la similarité ou la corrélation acoustique, et la **dynamique** d'auto-organisation fait émerger un "super-nœud" audio commun.

E. Conclusion

La détection d'un événement commun dans deux flux audio s'inscrit naturellement dans la logique du **DSL**. Sur le plan mathématique :

599. On divise chaque flux audio en **trames** $\{\mathcal{E}_i^{(1)}, \mathcal{E}_j^{(2)}\}$.

600. On calcule une synergie $S(\mathbf{x}_i^{(1)}, \mathbf{x}_j^{(2)})$ fondée sur une **similarité** acoustique (distance, cosinus, corrélation).

601. La **pondération** $\omega_{(i,j)}$ se met à jour via $\omega_{(i,j)}(t+1) = \omega_{(i,j)}(t) + \eta[S(i,j) - \tau \omega_{(i,j)}(t)]$, soulignant la correspondance audio entre flux 1 et flux 2.

602. Les liens ω élevés indiquent un **événement** simultané, capté par les deux micros.

Cette approche s'inscrit dans la **fusion** multimodale ou multi-canal du **Deep Synergy Learning**, permettant de **repérer** et **structurer** les phénomènes sonores communs, sans pipeline rigide, mais par **auto-organisation** locale des pondérations.

8.5.2.3. Inhibition si bruit ou signaux incohérents

Dans un contexte **multimodal** où plusieurs flux (audio, vision, texte) se combinent au sein d'un **SCN** (Synergistic Connection Network), il arrive qu'un ou plusieurs de ces flux soient **bruités** ou **incohérents**. L'un des enjeux est de **préserver** l'auto-organisation du **DSL** (Deep Synergy Learning) malgré ces perturbations, afin que les liens $\omega_{i,j}$ se focalisent sur les signaux fiables. L'**inhibition** joue alors un rôle de **contrôle** en limitant ou diminuant la pondération attachée aux canaux jugés douteux, empêchant ainsi la formation de clusters artificiels induits par le bruit.

A. Logique d'Inhibition vis-à-vis du Bruit

Il arrive qu'une **modalité** donnée (par exemple un micro défectueux, ou un flux vidéo recouvert de neige numérique) produise des **données** fortement dispersées ou contradictoires par rapport aux autres sources. La **mise à jour** du DSL, si elle reposait uniquement sur une synergie "brute" $S(i,j)$, risquerait de connecter abusivement ces entités au reste du réseau. Pour y remédier, on introduit un **terme d'inhibition** qui pénalise la pondération $\omega_{i,j}$ lorsqu'une entité \mathcal{E}_i est trop "bruyante".

On peut formaliser cela en définissant un **score** de bruit $B_i \geq 0$ pour l'entité \mathcal{E}_i . Si B_i est élevé, cela signale que la modalité de \mathcal{E}_i est peu fiable ou contaminée par du bruit. Dans la règle de mise à jour, on ajoute un terme proportionnel à B_i pour **diminuer** $\omega_{i,j}$. Par exemple :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma(\delta B_i + \delta B_j) \omega_{i,j}(t).$$

Ici, $\gamma > 0$ et $\delta > 0$ sont des coefficients de réglage. L'idée est que plus B_i ou B_j est grand, plus le lien $\omega_{i,j}$ se voit "poussé vers le bas" (inhibé).

Si un micro donne un **signal** très fluctuant ou saturé, $B_{\text{micro}} \gg 0$ et la dynamique DSL diminue $\omega_{\text{micro},\dots}$. Ce canal ne crée plus de liaisons parasites dans le **SCN**, et les clusters formés par les flux restants demeurent cohérents. De même, si un flux vidéo est en "panne" (images noires), on peut affecter B_{video} élevé pour neutraliser son influence.

B. Incohérence entre Modalités

Au-delà du bruit "aléatoire", il peut survenir des **contradictions** entre deux modalités. Par exemple, un flux textual indique un certain mot ("chat"), tandis qu'une image montre en réalité un "chien". Si la **similarité** brute $S(i,j)$ n'arrive pas à distinguer cette incohérence (peut-être parce que l'extraction de features visuelles est ambiguë), on souhaite **pénaliser** leur liaison $\omega_{i,j}$.

On peut introduire un "facteur d'incohérence" $C_{i,j} \geq 0$ dès qu'un superviseur (ou un module externe) repère une contradiction sémantique ou temporelle.

La mise à jour DSL standard :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

se voit complétée d'un **terme** d'inhibition :

$$-\gamma C_{i,j} \omega_{i,j}(t),$$

ce qui donne :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] - \gamma C_{i,j} \omega_{i,j}(t).$$

Quand $C_{i,j}$ est élevé, la pondération $\omega_{i,j}$ se trouve **freinée**, évitant un renforcement artificiel entre deux entités mal assorties.

C. Perspective Mathématique et Algorithmique

Dans la vision "énergie" (cf. chapitres analytiques), ajouter un terme de la forme $\gamma C_{i,j} \omega_{i,j}^2$ ou $\gamma C_{i,j} \omega_{i,j}$ fait croître l'énergie $J(\{\omega_{i,j}\})$ lorsque des liens s'établissent entre entités incohérentes. Cela provoque la **répulsion** ou la **dissuasion** de tels liens dans le schéma global.

Au niveau **itératif**, la formule "inhibition" agit à chaque pas et dépend de la pondération elle-même. Autrement dit, plus $\omega_{i,j}$ est élevé tout en ayant un $C_{i,j}$ non négligeable, plus on "soustrait" d'incrément dans la mise à jour. Cela évite qu'un bruit passager ou un désaccord modal ne provoque un cluster incorrect.

D. Conclusion (8.5.2.3)

L'**inhibition** appliquée aux **sources** bruyantes ou aux **incohérences** inter-modales confère au **DSL** une **robustesse** cruciale :

603. **Contrôle du bruit** : on diminue les liaisons issues de modalités jugées peu fiables.

604. **Gestion des contradictions** : si deux flux s'opposent sémantiquement ou temporellement, on empêche ω de croître en leur faveur.

605. **Convergence plus sûre** : la dynamique $\omega_{i,j}(t+1) = \dots$ ne s'égare pas dans la fusion "toxique" de signaux incohérents.

Matériellement, c'est un **terme** supplémentaire (terme d'inhibition) dans la **règle** de mise à jour ou dans la **fonction** d'énergie, modulé par un **score** de bruit ou d'incohérence $B_i, C_{i,j}$. Cela **assure** que la structure **auto-organisée** par le SCN reste **propre** et **significative**, même si certaines données sont altérées ou contradictoires.

8.5.3. Fusion Audio–Vision ou Audio–Texte

Dans de nombreux domaines multimédias (reconnaissance de scènes, sous-titrage automatique, visioconférence intelligente, etc.), la **fusion** de sources **audio** et **visuelles** (ou audio et textuelles) améliore considérablement la robustesse et la qualité d'interprétation. Le **DSL** (Deep Synergy Learning), par sa logique de $\omega_{i,j}$ adaptatifs et ses stratégies d'auto-organisation, offre un cadre pour relier les entités provenant de flux hétérogènes (ex. frames vidéo, segments audio, tokens textuels) en un **SCN** synergique. Cette section 8.5.3 détaille les deux grandes approches de fusion, l'**audio–vision** (8.5.3.1) et l'**audio–texte** (8.5.3.2).

8.5.3.1. Audio–Vision : reconnaissance vidéo-audio conjointe (lip sync, ambiance sonore)

Les applications conjuguant un **flux vidéo** et un **flux audio** sont nombreuses et recouvrent des scénarios variés, comme la **lip sync** (analyse de la cohérence entre lèvres et voix), la **détection** d'événements sonores (sirènes,

applaudissements), ou la **compréhension** d'une scène globale (apparition d'un véhicule visible accompagné d'un bruit de moteur). Le cadre du **DSL** (Deep Synergy Learning) et, en particulier, l'organisation du **SCN** (Synergistic Connection Network), facilite la **fusion** de ces deux modalités, car il autorise une représentation unifiée des **entités** (frames vidéo et segments audio) et un **calcul** itératif des **liens** $\omega_{i,j}$ reflétant leur **synergie**. L'objectif est de montrer comment cette synergie peut se définir et se mettre à jour pour aboutir à une **reconnaissance conjointe** plus robuste et plus fine.

A. Représentation et Synergie Audio–Vision

Une **séquence vidéo** peut être découpée en **frames** $\{\mathbf{v}_t\}_{t=1,\dots}$ s'échelonnant dans le temps selon un pas de capture (par exemple 25 images par seconde). Chaque frame \mathbf{v}_t est considérée comme une **entité** $\mathcal{E}_{\text{vis},t}$. De son côté, le **flux audio** associé, qui s'étend sur la même période temporelle, est segmenté en petits blocs ou trames $\{\mathbf{a}_u\}_{u=1,\dots}$, chacune représentant un fragment d'onde (voir 8.5.1.3 sur le traitement de séquences audio). Chaque segment audio \mathbf{a}_u devient une **entité** $\mathcal{E}_{\text{aud},u}$. Le **SCN** regroupe ainsi un ensemble mixte d'entités, comprenant des frames visuelles et des segments audio, prêtes à être reliées si elles présentent une **cohérence** temporelle ou sémantique.

Pour quantifier la **compatibilité** entre un frame vidéo \mathbf{v}_t et un segment audio \mathbf{a}_u , on définit une fonction de **synergie** $S(\mathbf{v}_t, \mathbf{a}_u)$. Sur le plan mathématique, on peut recourir à une transformation

$$\mathbf{z}_{\text{vis}}(t) = \Phi_{\text{vision}}(\mathbf{v}_t), \quad \mathbf{z}_{\text{aud}}(u) = \Phi_{\text{audio}}(\mathbf{a}_u),$$

où Φ_{vision} et Φ_{audio} sont des réseaux ou des modules extrayant des **embeddings**. La synergie s'exprime ensuite par

$$S(\mathbf{v}_t, \mathbf{a}_u) = \rho(\mathbf{z}_{\text{vis}}(t), \mathbf{z}_{\text{aud}}(u)),$$

où $\rho(\cdot, \cdot)$ peut être une **similarité cosinus**, un **kernel** Gaussien, ou encore un **coefficent** de corrélation temporel (si l'on cherche un alignement plus précis). Des considérations de **fenêtrage** (κ) peuvent s'ajouter si l'on estime devoir modéliser un léger décalage Δt entre l'événement visuel et le son correspondant.

B. Mécanisme d'Auto-Organisation dans le SCN

Le **DSL** actualise les liaisons $\omega_{(\text{vis},t),(\text{aud},u)}$ selon la **règle** habituelle :

$$\omega_{(t,u)}(k+1) = \omega_{(t,u)}(k) + \eta [S(\mathbf{v}_t, \mathbf{a}_u) - \tau \omega_{(t,u)}(k)],$$

où $\omega_{(t,u)} \equiv \omega_{(\text{vis},t),(\text{aud},u)}$. Si $S(\mathbf{v}_t, \mathbf{a}_u)$ se maintient à un niveau élevé (par exemple, forte corrélation lip sync, ou ambiance sonore parfaitement concordante avec ce que montre la vidéo), la pondération $\omega_{(t,u)}$ croît. Dans le cas contraire, elle retombe vers zéro. Les entités (frames, segments audio) se **regroupent** donc au sein d'un **cluster** si elles perçoivent le même phénomène (voix synchronisée, bruit d'objets visibles, etc.).

Le **cas** de la lip sync se formalise en associant par exemple un vecteur \mathbf{l}_t décrivant la **forme labiale** détectée dans \mathbf{v}_t , et un vecteur \mathbf{p}_u décrivant la **phonétique** principale de \mathbf{a}_u . La synergie s'écrit :

$$S(\mathbf{v}_t, \mathbf{a}_u) = \rho(\mathbf{l}_t, \mathbf{p}_u) \cdot \kappa(|t - u|),$$

indiquant qu'on pèse la similarité labiale–phonétique par la proximité temporelle $|t - u|$. Une fois insérée dans la dynamique $\omega_{(t,u)}(k+1) = \dots$, les paires (t, u) véritablement alignées voient leurs liaisons ω se consolider.

Un autre usage est la **reconnaissance d'événements** comme un “orchestre en train de jouer”, où le flux vidéo repère des instruments, des musiciens ; le flux audio capte le timbre musical correspondant. La synergie $S(\mathbf{v}_t, \mathbf{a}_u)$ prend alors une forme plus large, par exemple un **embedding** cross-modal dérivé d'un entraînement supervisé (Chap. 8.3.3.1 sur texte–image, transposable à audio–vision). Les frames associées à l'instrument musical se lient alors fortement avec les segments audio contenant des ondes musicales similaires.

C. Résultats et Bénéfices

En fin de convergence, le **SCN** engendre des **clusters** reliant plusieurs frames vidéo (t_1, t_2, \dots) aux segments audio (u_1, u_2, \dots) jugés “similaires” ou “synchrones”. Cette structure en **macro-nœuds** reflète un même événement ou *objet*

perçu simultanément dans les deux modalités. L'analyse de la scène s'en trouve enrichie, un vecteur "voix–visage" est plus robuste qu'une simple piste audio isolée ou qu'un visage muet.

Le fait de disposer de deux sources (audio et vision) permet de **compenser** les failles de l'une via l'autre. Si la qualité vidéo est dégradée, le son confirme ou infirme la scène ; si l'audio est parasité, la vidéo maintient un indice visuel. Le **DSL** gère ces divergences en modulant $\omega_{(t,u)}$ selon la valeur $S(\mathbf{v}_t, \mathbf{a}_u)$. Les saturations ou bruits (Chap. 8.5.2.3) peuvent être traités par un **terme d'inhibition** freinant les liaisons issues d'un flux peu fiable.

Le système peut s'étendre à un flux audio supplémentaire ou à une modalité textuelle, un **SCN** plus large comprendra des entités $\mathcal{E}_{\text{vis},t}$, $\mathcal{E}_{\text{aud1},u}$, $\mathcal{E}_{\text{aud2},v}$ voire $\mathcal{E}_{\text{txt},k}$ si l'on annexe des sous-titres ou légendes. La logique DSL reste la même : on calcule $S(i,j)$ et met à jour $\omega_{i,j}$ pour toute paire susceptible de présenter une synergie. Le graphe final illustre la **fusion** de multiples flux dans un réseau auto-organisé et localement cohérent.

Conclusion

La **fusion** audio–vision, dans un cadre DSL, s'appuie sur la représentation unifiée des **frames vidéo** et **segments audio** au sein d'un **SCN**, et sur une **fonction** de synergie $S(\mathbf{v}_t, \mathbf{a}_u)$ évaluant la **cohérence** (lip sync, ambiance, bruitage) entre ces deux modalités. Les liaisons $\omega_{(t,u)}$ se renforcent ou s'atténuent selon la correspondance détectée, menant à une **auto-organisation** en clusters multimodaux. Cette synergie est essentielle pour synchroniser la parole aux mouvements labiaux, pour repérer des bruits concordant avec des objets visibles, ou pour rehausser la reconnaissance d'événements complexes (un orchestre en scène, un klaxon d'automobile filmée, etc.). Sur le plan **mathématique**, la démarche consiste à définir un **embedding** ou un **score** cross-canal, puis à laisser la **dynamique** DSL itérer. Le résultat final procure une **vision** plus globale et plus robuste de la scène, où chaque source compense les lacunes de l'autre.

8.5.3.2. Audio–Texte : Alignement transcription–voix, synergie si corrélation forte

Un **DSL** (Deep Synergy Learning) multimodal peut se doter d'un **SCN** (Synergistic Connection Network) dédié à relier des **segments audio** (par ex. trames vocales) à des **éléments textuels** (tokens, morceaux de phrase) pour former un **alignement** plus ou moins explicite entre le **signal acoustique** et la **transcription** proposée. La notion de **corrélation** ou de **cohérence** audio–texte s'avère déterminante. Dès lors que la **synergie** entre un segment acoustique $\mathcal{E}_{\text{aud},m}$ et un segment textuel $\mathcal{E}_{\text{txt},n}$ dépasse un certain seuil (ou s'avère nettement plus grande que pour d'autres paires), la **pondération** $\omega_{(m,n)}$ se renforce, rendant l'alignement plus stable. Les paragraphes suivants détaillent les fondements mathématiques de cet alignement et son intégration dans la dynamique DSL.

A. Représentation Audio–Texte et Segments

Sur le plan **mathématique**, on découpe un **flux audio** continu en petites **trames** (ex. 20 ou 30 ms), chacune représentée par un **vecteur** $\mathbf{a}_m \in \mathbb{R}^d$ (ex. MFCC, spectrogramme résumé, ou embedding neuronal). On obtient alors un ensemble $\mathcal{A} = \{\mathbf{a}_1, \dots, \mathbf{a}_M\}$.

On dispose également d'une séquence textuelle $\mathcal{T} = \{t_1, \dots, t_N\}$, où chaque t_n est un **token** (mot, sous-mot, phonème, caractère), et l'on peut associer à chaque token t_n un vecteur $\mathbf{x}_n \in \mathbb{R}^{d'}$ (embedding lexical, projection phonémique, etc.).

Dans un **SCN** multimodal, chaque trame audio $\mathcal{E}_{\text{aud},m}$ et chaque token textuel $\mathcal{E}_{\text{txt},n}$ deviennent des **nœuds**. On cherche à quantifier une **synergie**

$$S(\mathcal{E}_{\text{aud},m}, \mathcal{E}_{\text{txt},n}) = S(\mathbf{a}_m, \mathbf{x}_n)$$

indicative de la **cohérence** entre \mathbf{a}_m (descripteur audio) et \mathbf{x}_n (descripteur textuel).

B. Définition de la Synergie Audio–Texte

Si l'on évalue la correspondance "voix" vs. "transcription", on peut modéliser chaque trame audio \mathbf{a}_m comme un **embedding phonétique**, tandis que chaque token t_n porte un embedding sémantique ou phonétique. On définit alors

$$\rho(\mathbf{a}_m, \mathbf{x}_n) = \exp(-\alpha \|\Phi(\mathbf{a}_m) - \Psi(\mathbf{x}_n)\|^2),$$

où $\Phi(\cdot)$ et $\Psi(\cdot)$ sont deux **projections** (réseaux ou mappings) transformant la trame audio et le token dans un espace *common* pour la comparaison. Plus $\rho(\mathbf{a}_m, \mathbf{x}_n)$ se rapproche de 1, plus la trame \mathbf{a}_m correspond à la prononciation du token t_n .

Le **DSL** met à jour la pondération $\omega_{(m,n)}$ reliant $\mathcal{E}_{\text{aud},m}$ et $\mathcal{E}_{\text{txt},n}$. La formulation standard est :

$$\omega_{(m,n)}(k+1) = \omega_{(m,n)}(k) + \eta [S(\mathbf{a}_m, \mathbf{x}_n) - \tau \omega_{(m,n)}(k)].$$

Ainsi, si $S(\mathbf{a}_m, \mathbf{x}_n) = \rho(\mathbf{a}_m, \mathbf{x}_n)$ se maintient à une valeur haute, la pondération $\omega_{(m,n)}$ croît au fil des itérations, marquant un **alignement** fort entre la trame audio m et le token textuel n . En revanche, si $\rho(\mathbf{a}_m, \mathbf{x}_n) \approx 0$, la liaison retombe vers zéro et la trame audio demeure non alignée à ce token textuel.

C. Alignement Temporel et Contrainte de Séquence

Dans un usage **classique** de speech-to-text, la prononciation des tokens suit l'ordre du texte. La trame audio \mathbf{a}_m ne saurait correspondre à un token t_n précédé par un autre token t_{n+1} aligné avec \mathbf{a}_m , où $m' < m$. Cette **monotonie** peut être intégrée dans le **SCN** en ajoutant un **terme** de pénalisation ou en restreignant les liaisons candidates $\omega_{(m,n)}$. Mathématiquement, on veille à ce que

$$m < m' \Rightarrow n \leq n',$$

limitant la configuration des liaisons $\{\omega_{m,n}\}$ dans le graphe.

Le **DSL** s'exécute toutefois localement et peut tolérer quelques violations ou retards si la corrélation le justifie (accents, sur-articulation). On obtient un alignement final plus **souple** qu'un forçage déterministe. Les liaisons fortes $\omega_{(m,n)}$ révèlent la séquence audio–texte la plus probable dans l'espace des correspondances.

D. Bénéfices et Applications

Dans un **pipeline** classique de speech-to-text, la liaison $\mathbf{a}_m \rightarrow t_n$ est déterminée par un modèle HMM ou un transformeur. Le **DSL** peut **améliorer** ou **réviser** cet alignement. Si un mot t_n semble mal placé (faible corrélation avec les trames audio en regard), la mise à jour $\omega_{(m,n)} \approx 0$ le met en évidence ; un mot plus approprié $t_{n'}$ pourra recevoir plus de pondération.

Si le **texte** n'est pas fidèle à la bande audio, les **valeurs** $\rho(\mathbf{a}_m, \mathbf{x}_n)$ restent basses, entraînant l'échec du renforcement $\omega_{(m,n)}$. On repère ainsi des divergences, ou on crée un "texte orphelin" (aucune trame audio ne se relie vraiment à lui).

Une fois l'**audio** aligné au **texte**, on peut **fusionner** un flux vidéo de la même scène. La dynamique DSL prend en charge **trois** modalités (audio, texte, vision), unifiant l'ensemble en un graphe plus riche. Chaque trame audio, token textuel et frame vidéo tente d'établir un lien de synergie, révélant des clusters tri-modaux (ex. "Cette personne vue à la caméra prononce ce mot, au moment T").

Conclusion

La **fusion audio–texte** dans un **DSL** (Deep Synergy Learning) vise à **aligner** (ou associer) chaque segment acoustique (trame audio) à chaque token textuel selon une **synergie** qui quantifie leur **corrélation** phonémico-lexicale. Sur le plan **mathématique**, on définit une **similarité** $\rho(\mathbf{a}_m, \mathbf{x}_n)$ reliant l'entité audio $\mathcal{E}_{\text{aud},m}$ et l'entité textuelle $\mathcal{E}_{\text{txt},n}$. La **mise à jour** $\omega_{(m,n)}$ s'effectue ensuite par la règle DSL :

$$\omega_{(m,n)}(k+1) = \omega_{(m,n)}(k) + \eta [S(\mathbf{a}_m, \mathbf{x}_n) - \tau \omega_{(m,n)}(k)].$$

Les paires (ou clusters) qui affichent une **corrélation forte** voient leur pondération augmenter, validant ainsi l'alignement entre la partie audio et le token textuel concerné. De manière incrémentale, le **SCN** converge vers une **structuration** où l'audio et le texte se combinent en segments cohérents. Cette stratégie :

606. **Facilite** la mise en évidence d'un alignement temporel plus souple qu'un algorithme classique,

607. **Déetecte** les segments mal transcrits (faible synergie audio–texte),
608. **Permet** l’extension multimodale (ajout d’un flux vidéo ou d’autres données) pour une **reconnaissance** d’autant plus robuste et intégrée de la scène.

8.5.3.3. Exemples : SCN reliant segments audio et frames vidéo pour une scène de film

Dans de nombreux scénarios multimédias, tels que l’analyse de scènes cinématographiques, la compréhension de flux vidéo-audio, ou encore la recherche d’événements sonores dans un film, l’association automatique entre des **segments audio** (voix, bruitages, musique) et des **plans vidéo** (images clés, frames) se révèle primordiale. Le **SCN** (*Synergistic Connection Network*), dans le cadre du **DSL** (*Deep Synergy Learning*), fournit un cadre mathématique pour **fusionner** ces deux modalités de manière adaptative et **auto-organisée**.

A. Structuration des Entités : Segments Audio et Frames Vidéo

On considère la piste sonore d’une **scène** de film, que l’on découpe en segments $\{A_1, A_2, \dots\}$. Chaque segment A_i est choisi selon des critères temporels (découpage en intervalles de quelques secondes) ou des changements détectés (début/fin d’un dialogue, rupture dans le niveau sonore, changement de musique, etc.). Sur le plan **mathématique**, chaque segment audio $\mathcal{E}_i^{(\text{audio})}$ peut être muni d’un **vecteur** $\mathbf{a}_i \in \mathbb{R}^{d_a}$, lequel encapsule des descripteurs (MFCC, spectrogrammes abrégés, ou embeddings neuronaux d’Audio2Vec). Selon la complexité, on peut également recourir à des indicateurs plus symboliques (par ex. “voix personnage A”, “bruit moteur”, etc.).

Côté vidéo, on prélève des **frames clés** ou on segmente la scène en **plans** plus longs $\{V_1, V_2, \dots\}$. Chaque entité $\mathcal{E}_j^{(\text{video})}$ correspond alors à une image (frame) ou un bloc d’images (plan), associé(e) à un **embedding** visuel $\mathbf{v}_j \in \mathbb{R}^{d_v}$ (issu, par exemple, d’un réseau type ResNet, VGG, ou d’un transformeur visuel).

Dans une approche plus sémantique, il est possible d’enrichir ce vecteur par des métadonnées. Des informations telles que “gros plan sur personnage X”, “scène de paysage” ou “mouvement de caméra rapide”, etc.

Le **SCN** vise à capturer la **synergie** $S(A_i, V_j)$ entre chaque segment audio A_i et chaque frame/plan vidéo V_j . Sur le plan **mathématique**, on peut l’exprimer comme

$$S(A_i, V_j) = \kappa(\mathbf{a}_i, \mathbf{v}_j),$$

où κ est une fonction de similarité adaptée :

- **Similarité temporelle.** Si A_i se situe sur la bande-son entre t_1 et t_2 , et V_j couvre le plan vidéo entre t_1' et t_2' , on peut définir un score fonction du recouvrement Δt .
- **Similarité sémantique.** On peut calculer un embedding “commun” pour l’audio et le visuel (type CLIP audio–vidéo) et mesurer la distance cosinus.
- **Alignement d’événement.** Un bruitage particulier (coup de feu) coïncide avec la vision d’une arme. κ alors prend en compte l’objet détecté dans l’image et le spectre caractéristique du son.

B. Mise en Place d’un SCN Audio–Vidéo

Dans le **SCN**, chaque entité $\mathcal{E}_i^{(\text{audio})}$ et $\mathcal{E}_j^{(\text{video})}$ est un **nœud** du réseau. La **pondération** $\omega_{(A_i, V_j)}$ reflète la “force” de lien, c’est-à-dire le **degré** de corrélation ou de compatibilité entre le segment audio A_i et le plan vidéo V_j .

La règle de **mise à jour** DSL repose sur :

$$\omega_{(A_i, V_j)}(t+1) = \omega_{(A_i, V_j)}(t) + \eta [S(A_i, V_j) - \tau \omega_{(A_i, V_j)}(t)].$$

Le **terme** $S(A_i, V_j)$ correspond à la **synergie** audio–vidéo (temporelle, sémantique, etc.), τ contrôle la décroissance, et η fixe le pas d'apprentissage.

Pour éviter qu'un segment audio A_i ne se “répande” en se liant moyennement à plusieurs plans vidéo $\{V_j\}$ à la fois, on peut inclure une **inhibition** latérale. *Mathématiquement*, cela se traduit par un **terme** supplémentaire (voir chap. 7.4) pénalisant la somme des $\omega_{(A_i, \cdot)}$ ou la création de trop nombreux liens modérés, forçant chaque segment audio à **sélectionner** les frames vidéo les plus corrélées.

À mesure que la **dynamique** s'opère, certains liens $\omega_{(A_i, V_j)}$ deviennent **forts** (forte corrélation), tandis que d'autres tombent vers 0 (peu ou pas de correspondance). Ainsi se **forment** des **clusters** multimodaux, regroupant un **ensemble** de segments audio $\{A_{i_1}, \dots, A_{i_k}\}$ et un **ensemble** de frames vidéo $\{V_{j_1}, \dots, V_{j_m}\}$. Sur le plan **conceptuel**, cela correspond à détecter qu'une “séquence” audio (voix, musique, bruit répétitif) s'aligne avec une “séquence” visuelle (plan fixe sur un personnage, travelling, etc.).

C. Avantages Mathématiques et Opérationnels

La mise à jour

$$\omega_{(A_i, V_j)}(t+1) = \omega_{(A_i, V_j)}(t) + \eta [S(A_i, V_j) - \tau \omega_{(A_i, V_j)}(t)]$$

permet, de manière **auto-organisée**, d'**assortir** chaque segment audio à son (ou ses) plan(s) vidéo correspondant(s). Cela évite un pipeline rigide de type “forçage temporel”. Les *clusters* émergent de la dynamique même du SCN, d'autant plus robustement que la **synergie** $S(A_i, V_j)$ est signifiante.

Lorsque la dynamique atteint un régime stable, on peut repérer des **macro-nœuds** (grands clusters) associant plusieurs segments audio liés à plusieurs frames vidéo. Cela permet :

- **Indexation** : retrouver instantanément les parties audio associées à une portion vidéo ;
- **Recherche de scènes** : par exemple, identifier tous les endroits où un *bruit* particulier (sirène) survient en même temps qu'un *élément visuel* (véhicule de police).
- **Structure** de la scène : on comprend qu'un certain “dialogue audio” correspond à un plan rapproché sur le personnage parlant, etc.

Les segments audio *parasites* (souffle, parasites) qui ne coïncident pas visuellement avec un événement et ne montrent aucune corrélation $S(A_i, V_j) \approx 0$ resteront faiblement connectés ($\omega \approx 0$) et ne perturberont pas la structure. De même, si un plan vidéo est vide (ou peu informatif), il n'entre pas en forte synergie avec un segment sonore précis. Ainsi, le SCN “filtre” les liaisons non contributives, augmentant la **lisibilité** des clusters.

D. Exemple Illustratif : Séquence de Dialogue + Action

Considérons un **exemple** où une scène de western où deux personnages discutent longuement, puis survient un coup de feu. La **bande-son** est scindée en segments :

Segment A : voix du personnage 1 (2 secondes),

Segment B : voix du personnage 2 (2 secondes),

Segment C : coup de feu (1 seconde).

La **vidéo**, quant à elle, est découpée en *plans* :

Plan X : gros plan sur le visage du personnage 1,

Plan Y : plan sur le personnage 2,

Plan Z : plan large dévoilant l'action (impact du coup de feu).

On crée des **nœuds** $\{\mathcal{E}_1^{(\text{audio})}, \mathcal{E}_2^{(\text{audio})}, \mathcal{E}_3^{(\text{audio})}\}$ pour les segments audio A, B, C, et $\{\mathcal{E}_1^{(\text{video})}, \mathcal{E}_2^{(\text{video})}, \mathcal{E}_3^{(\text{video})}\}$ pour les plans X, Y, Z. Les pondérations $\omega_{(A,X)}, \omega_{(B,X)}, \dots$ évoluent via la règle DSL.

- **Voix 1** (A) se synchronise temporellement avec le *plan X* (gros plan), donc $S(A, X) \approx \text{fort}$, conduisant à $\omega_{(A,X)}$ élevé.
- **Voix 2** (B) correspond en temps au plan Y, d'où renforcement de $\omega_{(B,Y)}$.
- **Coup de feu** (C) se cale sur le plan Z (où l'on voit l'action), entraînant un $\omega_{(C,Z)}$ dominant.

À l'issue, on perçoit **trois** micro-clusters :

$\{A, X\}$ (personnage 1),

$\{B, Y\}$ (personnage 2),

$\{C, Z\}$ (coup de feu + plan large).

Cette **organisation** permet de manipuler la scène de façon **sémantique**. On peut demander de “retrouver le segment audio de la voix de personnage 1 et les plans vidéo lui correspondant”, ou encore “analyser la partie action (coup de feu) pour extraire la musique ou les bruitages environnants”. Le DSL a thus “auto-appris” ces liens par la dynamique de ω , sans pipeline figé.

Conclusion

Un **SCN** reliant **segments audio** $\{A_i\}$ et **frames/ plans vidéo** $\{V_j\}$ constitue un **cadre d'auto-organisation** :

- Les **pondérations** $\omega_{(A_i, V_j)}$ se mettent à jour par la règle :

$$\omega_{(A_i, V_j)}(t+1) = \omega_{(A_i, V_j)}(t) + \eta \left[S(A_i, V_j) - \tau \omega_{(A_i, V_j)}(t) \right].$$

- Les **clusters** émergent quand la synergie audio–vidéo $S(A_i, V_j)$ se révèle forte (corrélation temporelle, sémantique, ou autre).
- Les **événements** ou les dialogues bruyants (faible synergie) ne perturbent pas la structure, leurs liens restant faibles.

Cette approche **fusionne** les dimensions audio et visuelle de façon **souple**, en évitant des algorithmes de forçage ou de pipeline rigide. *Mathématiquement*, on dispose d'un **réseau** dont les liens ω évoluent vers un **ensemble** de clusters représentant la **correspondance** audio–vidéo. On bénéficie alors d'une **compréhension** plus fine et **adaptable** des scènes filmiques, facilitant l'indexation, la recherche de moments-clés, ou l'analyse multi-canal de l'action.

8.6. Construction d'un SCN Multimodal Unique

Après avoir examiné comment le **DSL** (Deep Synergy Learning) s'articule dans différentes modalités (vision, langage, audio, etc.), l'étape cruciale consiste à **fusionner** l'ensemble de ces sources d'information dans un **SCN unique**. Il s'agit de placer dans le même réseau des **nœuds** représentant des objets visuels ($\{\text{image}_i\}$), des éléments textuels ($\{\text{texte}_j\}$), ou encore des extraits sonores ($\{\text{audio}_k\}$). Les **liens** entre ces nœuds, mesurés par la synergie S , peuvent alors dépasser le strict domaine intra-modal (image-image, texte-texte, etc.) pour tisser des **connexions** inter-modales (vision–texte, audio–vision, texte–audio).

8.6.1. Entités Hétérogènes dans un Même Réseau

Ce sous-chapitre se penche sur la façon de **coexister** dans un unique SCN des entités issues de différentes modalités. Sur le plan mathématique, la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ doit être **spécifiée** ou **adaptée** en fonction des types (vision, texte, audio) que revêtent \mathcal{E}_i et \mathcal{E}_j . Cela requiert une organisation modulaire dans le calcul de la synergie (chap. 5.4) ou dans la définition de S (chap. 3.3, 3.4) afin de gérer efficacement les **paires hétérogènes**.

8.6.1.1. $\{\text{image}_i\}, \{\text{texte}_j\}, \{\text{audio}_k\}$ coexistent comme nœuds dans le SCN

Lorsqu'un **SCN** (Synergistic Connection Network) prend en charge plusieurs **modalités** telles que la vision (images), le texte et l'audio, il est possible — et souvent très fructueux — de faire **coexister** ces entités dans le **même** graphe. Dans cette perspective, chaque nœud du SCN n'est plus nécessairement une entité unimodale ; il peut tout aussi bien représenter une image, un segment textuel ou un segment audio, le tout dans une architecture commune. Cette démarche constitue un pas décisif vers la **fusion** multimodale intégrale, car elle autorise des **liens** ω entre objets visuels, extraits sonores et fragments textuels, tout en conservant la logique d'**auto-organisation** inhérente au DSL (Deep Synergy Learning).

A. Définition d'un Ensemble de Nœuds Multimodaux

Considérons un ensemble d'**images** $\{\text{image}_1, \dots, \text{image}_{N_I}\}$, un **corpus de textes** $\{\text{texte}_1, \dots, \text{texte}_{N_T}\}$, et un **ensemble** de segments **audio** $\{\text{audio}_1, \dots, \text{audio}_{N_A}\}$. Il est possible de réunir toutes ces entités dans une **même** structure de graphe :

$$\mathcal{U} = \{\text{image}_i: i = 1, \dots, N_I\} \cup \{\text{texte}_j: j = 1, \dots, N_T\} \cup \{\text{audio}_k: k = 1, \dots, N_A\}.$$

Toutes les entités $\mathcal{E} \in \mathcal{U}$ sont alors traitées comme des **nœuds** du **SCN**, et pourront donc être reliées entre elles par des **pondérations** $\omega_{i,j}$. La différence tient au fait qu'on manipule désormais trois modalités distinctes (vision, texte, audio) au sein de la même dynamique DSL.

B. Intégration dans un Espace Fédérateur ou via Modules Distincts

Sur le plan **mathématique**, on peut chercher à **projeter** chaque modalité dans un **espace** vectoriel **unique** \mathbb{R}^d . Concrètement, il s'agit de définir des transformateurs

$$\Phi_{\text{image}}(\text{image}_i) \in \mathbb{R}^d, \quad \Phi_{\text{texte}}(\text{texte}_j) \in \mathbb{R}^d, \quad \Phi_{\text{audio}}(\text{audio}_k) \in \mathbb{R}^d,$$

puis de **comparer** deux entités (même ou différente modalité) via une **similarité** ou une **distance** dans cet espace commun :

$$S(\text{image}_i, \text{texte}_j) = \text{sim}(\Phi_{\text{image}}(\text{image}_i), \Phi_{\text{texte}}(\text{texte}_j)).$$

De même pour audio–vision, audio–texte, etc. Une **similarité cosinus** ou un kernel Gaussien peut servir de base, tant que l'on garantit que chaque modalité est correctement projetée. L'avantage est que le **calcul** de $S(\cdot, \cdot)$ devient homogène, sans nécessiter de règles complexes selon la paire de modalités.

À l'inverse, on peut conserver des **représentations** différencierées (images dans $\mathbb{R}^{d_{\text{img}}}$, textes dans $\mathbb{R}^{d_{\text{txt}}}$, audio dans $\mathbb{R}^{d_{\text{aud}}}$) et confier à des “**modules**” spécifiques (voir Chap. 5.4 sur le *Module Synergie*) le soin de calculer $S(\text{image}_i, \text{texte}_j)$, $S(\text{image}_i, \text{audio}_k)$, $S(\text{texte}_j, \text{audio}_k)$, etc. Le **SCN** fait alors appel, pour chaque paire (i, j) , au **module** de synergie correspondant au couple de modalités, qui renvoie un **score** de compatibilité.

Cette seconde approche se prête bien à la **diversité** des modalités (images, sons, texte), évitant de forcer un unique espace latent. Elle exige néanmoins un certain nombre de **fonctions** $S_{\text{img-txt}}$, $S_{\text{img-aud}}$, $S_{\text{txt-aud}}$, rendant potentiellement le système plus complexe à calibrer.

C. Cohabitation des Nœuds dans le Même SCN

Dans l'architecture **SCN**, tous les **nœuds** (images, textes, audios) cohabitent dans un **même** graphe. Les **liaisons** $\omega_{(i,j)}$ relient potentiellement :

- **Image–Image** : $\omega_{(\text{image}_i, \text{image}_j)}$
- **Texte–Texte** : $\omega_{(\text{texte}_i, \text{texte}_j)}$
- **Audio–Audio** : $\omega_{(\text{audio}_i, \text{audio}_j)}$
- **Cross** (Image–Texte, Texte–Audio, Audio–Image, etc.).

Chacun de ces liens se **met** à jour selon la règle DSL :

$$\omega_{(i,j)}(t+1) = \omega_{(i,j)}(t) + \eta [S(i,j) - \tau \omega_{(i,j)}(t)].$$

Où $S(i,j)$ se spécialise selon la **modalité** ou l'**embedding** commun. Par exemple, s'il s'agit de deux images, S peut être la similarité cosinus de leurs vecteurs CNN ; s'il s'agit d'un texte et d'un audio, S peut être un score de correspondance phonétoco-sémantique ; etc.

À mesure que les liens $\omega_{(i,j)}$ évoluent, de **clusters** émergent rassemblant un **ensemble** d'entités hétérogènes (images + textes + audios). Par exemple, on peut avoir un cluster qui regroupe :

- Une **série** d'images montrant un chat,
- Un **texte** mentionnant “chat” ou décrivant la scène,
- Un **enregistrement** audio d'un miaulement.

Le **SCN** consolide ces associations si la synergie S est élevée, aboutissant à un **macro-nœud** suggérant un concept commun (“chat qui miaule, décrit textuellement”). C'est précisément cette **auto-organisation** tri-modale qui fait la force du DSL dans un environnement riche.

D. Avantages et Points d'Attention

En autorisant toutes les **paires** (image–texte, texte–audio, audio–image) à **exister** dans le **même** SCN, on laisse la dynamique DSL **découvrir** (et non imposer) quelles correspondances méritent d'être renforcées. Il peut émerger des clusters inattendus, comme un certain son associé à une description textuelle spécifique, ou un même thème (ex. “sport”) reliant plusieurs images, articles de presse (texte) et extraits sonores (clameurs de foule).

Si le nombre d'entités $N_I + N_T + N_A$ est important, le graphe peut atteindre $O(N^2)$ liens potentiels. Dans ce cas, il devient **nécessaire** de recourir à des techniques de **réduction** ou de **sparsification** (k-NN, rayon ϵ , filtrage adaptatif) pour éviter un coût trop élevé en calcul. Chap. 7.2.3 évoque ces stratégies.

Il est impératif que les diverses fonctions de **synergie** (image–image, image–texte, audio–texte, etc.) soient **cohérentes** en termes d'échelle. Si l'une renvoie des scores typiquement dans $[0.7, 1]$ tandis qu'une autre fluctue dans $[0, 0.3]$, la

dynamique DSL peut se trouver déséquilibrée. Une **normalisation** (chap. 8.5.1.2 sur l'amplitude/log-scale) peut s'imposer pour aligner les distributions de S .

Conclusion

Le fait de **coexister** dans un **même** SCN des ensembles d'**images** $\{\text{image}_i\}$, de **textes** $\{\text{texte}_j\}$ et de **segments audio** $\{\text{audio}_k\}$ installe le **fondement** d'un **DSL** réellement multimodal :

- 609.**Un seul** graphe réunit tous les nœuds $\mathcal{E} \in \mathcal{U}$,
- 610.**Des pondérations** $\omega_{(i,j)}$ traitant indifféremment la modality pair (image–texte, texte–audio, etc.),
- 611.**Un ensemble** de fonctions $S(\cdot, \cdot)$ ou un embedding commun pour comparer entités hétérogènes,
- 612.**Une auto-organisation** tri- (voire multi-) modale où des **clusters** rassemblent simultanément images, textes, audio autour d'un même concept ou d'un même événement.

Sur le plan **mathématique**, cela signifie étendre la mise à jour $\omega(t+1) = \omega(t) + \eta[S - \tau \omega(t)]$ à tous les couples (i,j) possibles, en veillant toutefois à limiter la **combinatoire** (via inhibition, k-NN). Il en résulte un **SCN** qui incorpore véritablement la pluralité des flux dans un **réseau unique**, favorisant la **synergie** entre modalités et l'émergence de **clusters** multimodaux plus riches et plus signifiants.

8.6.1.2. Chaque nœud a son type, le module synergie identifie la fonction $S_{\text{vision}, \text{texte}}, S_{\text{audio}, \text{vision}}, \dots$

Dans un **DSL** (Deep Synergy Learning) réellement multimodal, les **entités** qui composent le réseau peuvent appartenir à différentes modalités, qu'il s'agisse de vision, de texte, d'audio ou encore de données issues de capteurs. L'un des piliers de cette approche consiste à **typer** chaque nœud pour renseigner sa modalité (image, texte, segment audio...) et à sélectionner la **fonction** de synergie S appropriée lors de la mise à jour des pondérations $\omega_{i,j}$. Le **module Synergie** (chap. 5.4, 5.5) incarne précisément cette logique. Il détecte le couple de **types** (m_1, m_2) auquel appartiennent les nœuds \mathcal{E}_i et \mathcal{E}_j , puis calcule $S_{m_1, m_2}(\mathbf{x}_i, \mathbf{y}_j)$ selon la nature des deux modalités.

A. Notion de "Type" de Nœud et Sélection de la Fonction de Synergie

Chaque **nœud** \mathcal{E}_k du SCN se voit attribuer un **type** (ou label modal) :

$$\text{type}(\mathcal{E}_k) \in \{\text{Vision, Texte, Audio, ...}\}.$$

Sur le plan **mathématique**, on peut considérer différents espaces $\mathcal{X}_{\text{vision}}, \mathcal{X}_{\text{texte}}, \mathcal{X}_{\text{audio}}$, etc. Chaque entité \mathcal{E}_k appartient à l'un de ces espaces. Si $\text{type}(\mathcal{E}_k) = \text{Vision}$, alors \mathcal{E}_k est un **embedding** visuel $\mathbf{x}_k \in \mathcal{X}_{\text{vision}}$, etc.

Pour deux nœuds \mathcal{E}_i et \mathcal{E}_j , il convient de distinguer **plusieurs** fonctions de synergie :

$$S_{m_1, m_2}: \mathcal{X}_{m_1} \times \mathcal{X}_{m_2} \rightarrow \mathbb{R},$$

où $m_1 = \text{type}(\mathcal{E}_i)$ et $m_2 = \text{type}(\mathcal{E}_j)$. Autrement dit, si \mathcal{E}_i et \mathcal{E}_j sont tous deux des images (vision–vision), on utilise $S_{\text{vision}, \text{vision}}$. Si c'est un segment audio \mathcal{E}_i et une entité textuelle \mathcal{E}_j , on appelle $S_{\text{audio}, \text{texte}}$, etc.

De cette manière, on ne confond pas la **mesure** de ressemblance audio–audio (spectre, MFCC) avec celle de ressemblance texte–texte (similarité cosinus sur embeddings linguistiques). Le **module Synergie**, lors de la mise à jour $\omega_{i,j}$, identifie automatiquement $m_1 = \text{type}(i)$ et $m_2 = \text{type}(j)$, puis appelle S_{m_1, m_2} avec les représentations $\mathbf{x}_i, \mathbf{y}_j$ appropriées.

B. Illustration : Exemples de Fonctions S

Si \mathcal{E}_i et \mathcal{E}_j sont deux **images** (même modalité “Vision”), on peut définir

$$S_{\text{vision}, \text{vision}}(\mathbf{v}_i, \mathbf{v}_j) = \cos(\mathbf{v}_i, \mathbf{v}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|},$$

où $\mathbf{v}_i \in \mathbb{R}^{d_{\text{img}}}$ et $\mathbf{v}_j \in \mathbb{R}^{d_{\text{img}}}$ sont des embeddings CNN. On peut aussi recourir à un RBF kernel, ou à une distance euclidienne inversée.

Lorsque $\text{type}(\mathcal{E}_i) = \text{Vision}$ et $\text{type}(\mathcal{E}_j) = \text{Texte}$, il arrive qu'on dispose d'un **espace latent commun** (ex. CLIP) où $\mathbf{v}_i \in \mathbb{R}^d$ (projection de l'image) et $\mathbf{t}_j \in \mathbb{R}^d$ (projection du texte). On définit alors :

$$S_{\text{vision}, \text{texte}}(\mathbf{v}_i, \mathbf{t}_j) = \cos(\mathbf{v}_i, \mathbf{t}_j).$$

Cette fonction renvoie un **score** de correspondance cross-modal image–texte.

De la même manière, on peut configurer :

$S_{\text{audio}, \text{vision}}$: modèle la **corrélation** ou la **coïncidence** spatio-temporelle entre un signal audio et un flux vidéo (cf. lip-sync, ambiance sonore).

$S_{\text{audio}, \text{texte}}$: compare un extrait audio (caractéristiques phonétiques) à une transcription textuelle (caractéristiques lexicales ou phonémiques).

C. Mise en Œuvre dans la Règle DSL

Au cours de la **mise à jour** DSL, la pondération $\omega_{i,j}(t+1)$ est calculée par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Le **module Synergie** se charge de renvoyer la valeur de $S(\cdot, \cdot)$ en examinant :

$$\text{type}(\mathcal{E}_i) = m_1, \quad \text{type}(\mathcal{E}_j) = m_2,$$

puis appelle :

$$S(\mathcal{E}_i, \mathcal{E}_j) = S_{m_1, m_2}(\mathbf{x}_i, \mathbf{y}_j).$$

Où \mathbf{x}_i et \mathbf{y}_j sont les **représentations** (vecteurs, frames, embeddings) de \mathcal{E}_i et \mathcal{E}_j . Le reste du **DSL** ne se soucie pas de la modalité. Il applique la règle d'apprentissage de ω comme si $S(\cdot, \cdot)$ était unique.

La **clarté** du système permet de différencier efficacement vision–vision, qui regroupe des images similaires, de vision–texte, qui associe une image à sa légende. L'**évolutivité** est assurée, car l'ajout d'une **nouvelle** modalité, telle qu'un capteur ou un EEG, ne nécessite que la définition des fonctions de similarité $S_{\text{nouvelle}, \text{vision}}$ et $S_{\text{nouvelle}, \text{texte}}$, sans qu'il soit nécessaire de modifier le cœur du **DSL**.

D. Exemples d'Alignement Multi-Type

Image–Image : Permet un **clustering** d'images proches (ex. toutes les photos de paysages).

Texte–Texte : Regroupe des documents ou tokens sémantiquement similaires, formant des thèmes.

Audio–Audio : Classe des segments sonores (voix semblables, bruitages similaires).

Vision–Texte : Associe un objet visuel à sa description, facilitant l'annotation ou la recherche inversée (requête textuelle → images).

Audio–Texte : Aigne un flux vocal avec sa transcription ou des mots clés (reconnaissance).

Vision–Audio : Déetecte la correspondance (lip sync, ambiance sonore corrélée aux actions vidéo).

E. Conclusion

Le fait que **chaque nœud** possède un **type** (Vision, Texte, Audio...) et que le **module Synergie** sélectionne la **fonction** $S_{\text{type}(i), \text{type}(j)}$ appropriée constitue le fondement du **DSL multimodal**. Cette organisation formalise la comparaison entre modalités hétérogènes en utilisant un embedding commun ou des fonctions spécialisées, ce qui permet d'établir des **liens** (pondérations ω) entre nœuds de différentes **natures** sans mélange arbitraire. Elle renforce également la **cohérence** de l'auto-organisation, puisque chaque couple (image–image, image–texte, texte–audio, etc.) est évalué selon la **bonne** fonction de similarité. Enfin, elle facilite l'émergence de **clusters** véritablement multimodaux, unifiant vision, texte et audio autour d'un concept ou d'une situation commune.

En ce sens, le **module Synergie** agit comme un “router”. Selon $\text{type}(i)$ et $\text{type}(j)$, on applique S_{m_1, m_2} . Le **SCN** demeure ainsi **unifié** au niveau de la dynamique ω , tout en s'adaptant finement à la **spécificité** de chaque modalité.

8.6.1.3. Exemples de définitions multiples de S selon les paires de modalités

Dans un **DSL** (Deep Synergy Learning) véritablement multimodal, chaque entité \mathcal{E}_i peut appartenir à l'une de plusieurs **modalités**. Images, textes, sons, données de capteurs, etc. Pour que le **SCN** (Synergistic Connection Network) sache relier ces entités de manière significative, il est indispensable de **définir** des **fonctions** de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ adaptées à chaque **couple** de modalités. Ainsi, la comparaison image \leftrightarrow image ne se fera pas de la même façon que image \leftrightarrow texte ou audio \leftrightarrow texte. La présente section illustre comment on peut **multiplier** les définitions de S en fonction de ces paires de modalités, tout en conservant la **même** dynamique DSL sur la matrice de liaisons $\omega_{i,j}$.

A. Synergie Image–Image : mesures de similarité visuelle

Lorsqu'on compare deux **images** (ou frames vidéo), on peut représenter chaque image \mathcal{I}_i par un vecteur $\mathbf{v}_i \in \mathbb{R}^d$. Ce vecteur peut provenir :

- d'un **embedding** d'un réseau de neurones (CNN, ViT),
- de **descripteurs** locaux (SIFT, SURF) regroupés ou agrégés,
- d'un histogramme global (couleurs, textures).

Ensuite, la **synergie** $S_{\text{img}, \text{img}}$ se formule souvent via une **similarité cosinus**,

$$S_{\text{img}, \text{img}}(\mathcal{I}_i, \mathcal{I}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\|\mathbf{v}_i\| \|\mathbf{v}_j\|},$$

ou bien via une **distance euclidienne** inversée ou un **RBF** (radial basis function),

$$S_{\text{img}, \text{img}}(\mathcal{I}_i, \mathcal{I}_j) = \exp(-\alpha \|\mathbf{v}_i - \mathbf{v}_j\|^2).$$

Un score élevé indique des images jugées **proches** (même objet, même catégorie).

Dans des cas plus fins, on peut calculer des correspondances **locales** (patchs, keypoints) et tirer un score global de matching, comme :

$$S_{\text{img}, \text{img}}(\mathcal{I}_i, \mathcal{I}_j) = \frac{|\mathcal{M}(\mathbf{k}_i, \mathbf{k}_j)|}{\max(|\mathbf{k}_i|, |\mathbf{k}_j|)},$$

où \mathbf{k}_i et \mathbf{k}_j désignent des ensembles de points-clés (SIFT) et \mathcal{M} représente l'ensemble des appariements trouvés. Cela donne une mesure plus robuste aux changements d'échelle, rotations, etc.

B. Synergie Texte–Texte : similarité sémantique ou distributionnelle

Chaque **texte** (ou segment, paragraphe, document) \mathcal{T}_i peut être converti en un vecteur $\mathbf{w}_i \in \mathbb{R}^m$ via un modèle de langage (Word2Vec, GloVe, BERT...). La **synergie** s'écrit alors :

$$S_{\text{text},\text{text}}(\mathcal{T}_i, \mathcal{T}_j) = \cos(\mathbf{w}_i, \mathbf{w}_j),$$

pour refléter la **proximité sémantique**. On peut également recourir à un **kernel** exponentiel sur la distance $\|\mathbf{w}_i - \mathbf{w}_j\|$, etc.

De façon plus “classique”, on peut considérer un **vector** TF-IDF tfidf_i pour le texte \mathcal{T}_i . Alors, la similarité cosinus sur ces vecteurs

$$S_{\text{text},\text{text}}(\mathcal{T}_i, \mathcal{T}_j) = \frac{\text{tfidf}_i \cdot \text{tfidf}_j}{\|\text{tfidf}_i\| \|\text{tfidf}_j\|}$$

indique si les textes partagent les mêmes mots, avec pondération de leur fréquence et de leur rareté. On peut aussi définir un coefficient de Jaccard (chevauchement de tokens).

C. Synergie Audio–Audio : correspondances de signatures acoustiques

Pour deux segments audio $\mathcal{A}_i, \mathcal{A}_j$, on extrait des **features** (ex. MFCC, spectrogrammes résumés) sous forme de vecteurs $\mathbf{z}_i, \mathbf{z}_j$. On applique une **distance** ou une **similarité** :

$$S_{\text{aud},\text{aud}}(\mathcal{A}_i, \mathcal{A}_j) = \cos(\mathbf{z}_i, \mathbf{z}_j) \quad \text{ou} \quad \exp(-\|\mathbf{z}_i - \mathbf{z}_j\|^2).$$

Des segments audio comparables (même locuteur, même bruit, même instrument) obtiennent un score $S_{\text{aud},\text{aud}}$ élevé. Cela permet de **regrouper** (clustering) ou de **détecter** des répétitions sonores.

D. Synergie Inter-Modale : image–texte, audio–texte, image–audio, etc.

Lorsque \mathcal{E}_i est une image, \mathcal{E}_j un segment textuel, on définit souvent un **espace latent** commun (modèles de type CLIP) ou un simple **réseau** de correspondance :

$$S_{\text{img},\text{txt}}(\mathcal{I}, \mathcal{T}) = \cos(\Phi_{\text{img}}(\mathbf{v}_i), \Phi_{\text{txt}}(\mathbf{w}_j)).$$

Cela reflète la **compatibilité sémantique** entre l'**embedding** visuel et l'**embedding** textuel.

Pour aligner un **segment audio** avec un **texte** (transcription, mot-clef), on peut projeter \mathbf{a}_i (vecteur phonétique) et \mathbf{t}_j (embedding lexical) dans un espace cross-modal, ou recourir à la **transcription** auto-supervisée (ASR). On obtient un score

$$S_{\text{aud},\text{txt}}(\mathbf{a}_i, \mathbf{t}_j) = \rho(\Phi_{\text{audio}}(\mathbf{a}_i), \Phi_{\text{txt}}(\mathbf{t}_j)),$$

qui indique la **corrélation** entre la voix enregistrée et le contenu textuel présumé.

Plus rare, cette comparaison peut survenir dans un contexte vidéo (lip sync, bruits associés à des actions visuelles). On peut soit définir directement un **embedding** image–audio (modèle cross-modal), soit passer par un “hub” textuel (ex. l’audio transcrit en mots, l’image décrite par tags), puis comparer les mots. Le DSL autorise cependant un lien direct $\omega_{(\text{img}),(\text{audio})}$ dès lors qu’on dispose d’une **fonction** $S_{\text{img},\text{aud}}$.

E. Synthèse : une Famille de S pour chaque paire de Modalités

Sur un plan purement **mathématique**, la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ n’est pas une unique fonction universelle, mais une **famille** de fonctions :

$$S_{(\text{modality}_i, \text{modality}_j)},$$

adaptées à la modalité (ou au type) de \mathcal{E}_i et \mathcal{E}_j . Dans un **DSL multimodal**, le **module Synergie** (chap. 5) joue un rôle central dans l'évaluation des relations entre entités. Il **identifie** les types $\text{type}(\mathcal{E}_i)$ et $\text{type}(\mathcal{E}_j)$ afin de sélectionner la fonction de similarité appropriée. Ensuite, il **appelle** la fonction $S_{(\text{mod}(i), \text{mod}(j))}(\mathbf{x}_i, \mathbf{y}_j)$ correspondant aux modalités des deux entités concernées. Enfin, il **retourne** un score réel, intégrant si nécessaire un offset ou une fenêtre temporelle pour capturer la synchronisation entre signaux multimodaux.

Ce score est alors inséré dans la **mise à jour** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

De cette façon, un **SCN** unique gère la **même** matrice ω , tout en recourant à différentes **formules** de similarité/distances selon la nature des entités.

Conclusion

L'existence de **définitions multiples** pour la **synergie** S (image–image, texte–texte, audio–audio, image–texte, etc.) illustre la **souplesse** du **DSL**. Un **même** réseau $\{\omega_{i,j}\}$ peut accueillir diverses **modalités** de données, dès lors qu'on fournit au **module Synergie** une **famille** de fonctions $S_{\alpha,\beta}$ adaptée à chaque couple de modalités α, β .

Sur le **plan opérationnel**, cela permet de **fusionner** des canaux hétérogènes dans un **SCN** unique, où les liens (pondérations $\omega_{i,j}$) se **renforcent** ou se **relâchent** en fonction de la synergie réelle entre les entités.

Sur le **plan mathématique**, on obtient une **auto-organisation** cross-modale où un **cluster** peut inclure des **images** (liées entre elles), des **textes** (liés entre eux) et des **audios**, tout en reliant image–texte, texte–audio, image–audio lorsque leurs synergies inter-modales sont jugées fortes.

C'est la **philosophie** fondamentale d'un **SCN** multimodal où la dynamique ω exploite différentes mesures $S_{(\alpha,\beta)}$ pour **révéler** les correspondances et faire émerger des **clusters** réellement multidimensionnels (image + texte + audio + ...) autour d'un même **concept** ou **événement**.

8.6.2. Densité et Parcimonie

Dans un contexte **multimodal**, où l'on fusionne potentiellement plusieurs ensembles d'entités (images, textes, sons...), le **SCN** (Synergistic Connection Network) peut se retrouver envahi par un **trop grand** nombre de liaisons. La synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ calculée entre deux entités multimodales (ex. image vs. texte) n'est pas toujours nulle, et l'on risque de multiplier les liens $\omega_{i,j}$ dès lors qu'il existe un soupçon de corrélation. D'où l'importance de **contrôler** la densité du réseau et de préserver une forme de **parcimonie** qui maintient la cohérence et la scalabilité.

8.6.2.1. Risque de prolifération de liens si toutes les images se lient à tous les textes

Dans un **SCN** (Synergistic Connection Network) dédié à la fusion multimodale, il est naturel de vouloir relier des **images** à des **textes**, par exemple pour établir des correspondances entre le contenu visuel et des descriptions ou des mots-clés. Cependant, si l'on se contente de calculer une **synergie** $S(i, j)$ pour **toutes** les paires ($i \in \{\text{images}\}, j \in \{\text{textes}\}$) sans appliquer de mécanisme de filtrage ou de sélection, on peut rapidement faire face à une **explosion** du nombre de liens $\omega_{i,j}$. Cette situation nuit autant à la **lisibilité** du réseau qu'à son **efficacité** computationnelle et à sa capacité à faire émerger des **clusters** clairement distincts.

A. Explosion du nombre de liaisons et coût $O(N_{\text{img}} N_{\text{txt}})$

Considérons un ensemble de N_{img} images et un ensemble de N_{txt} textes. Dans un **SCN** où chaque image img_i est reliée à chaque texte txt_j via une pondération $\omega_{i,j}$, on obtient au total $O(N_{\text{img}} \times N_{\text{txt}})$ liaisons potentielles.

Sur le plan **mathématique**, la mise à jour du SCN (voir chapitre 4) impose, à chaque itération, de calculer la synergie

$$S(\text{img}_i, \text{txt}_j)$$

puis de mettre à jour la pondération

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\text{img}_i, \text{txt}_j) - \tau \omega_{i,j}(t)].$$

Lorsque N_{img} et N_{txt} deviennent significatifs (des milliers, voire plus), cette itération globale requiert un nombre de calculs de l'ordre de $N_{\text{img}} \times N_{\text{txt}}$. Le **coût** peut alors s'avérer prohibitif et ralentir fortement la convergence du réseau.

B. Risque de densification et perte de discrimination

Lorsqu'une **image** se relie à un trop grand nombre de **textes**, ou qu'un **texte** se retrouve faiblement corrélé avec de nombreuses images, le SCN tend à devenir **très dense**. Cela engendre deux problèmes majeurs :

Manque de sélectivité des liaisons. Un lien $\omega_{i,j}$ qui n'est que moyennement élevé (et qui ne redescend jamais à un niveau quasi nul) peut persister artificiellement. On se retrouve alors avec un grand nombre de liaisons "moyennes" ou "intermédiaires", sans que le réseau n'affiche un réel contraste entre liens "très pertinents" et liens "faiblement pertinents".

Difficulté à faire émerger des clusters nets. Sur un plan **conceptuel**, on aimerait voir des **clusters** "Images de chats + mots-clés félin", ou "Images de paysages + vocables sur la nature". Si l'immense majorité des textes entretient un degré de similarité (même léger) avec la plupart des images, la structure du réseau devient confuse, car rien ne s'oppose à ce que liaisons "moyennes" se multiplient. Les macro-nœuds deviennent moins identifiables, le SCN ressemblant plus à un vaste nuage de liaisons qu'à une organisation claire.

C. Perte de lisibilité et charge mémoire

D'un **point de vue pratique**, stocker $\omega_{i,j}$ pour $O(N_{\text{img}} \times N_{\text{txt}})$ liens peut occuper une **grande quantité** de mémoire si les matrices de pondérations sont denses, en particulier quand N_{img} et N_{txt} se chiffrent en dizaines ou centaines de milliers.

Par ailleurs, la **visualisation** ou l'**inspection** du réseau devient ardue. Un SCN trop dense rend difficile la lecture de la structure. Même si, théoriquement, la mise à jour DSL pourrait faire baisser certains $\omega_{i,j}$, la dynamique risque de prendre beaucoup d'itérations pour "nettoyer" efficacement les liaisons moyennes ou faibles.

D. Perte d'efficacité d'auto-organisation

Le **DSL** (chap. 4) et ses mécanismes annexes (recuit, inhibition, etc.) ont précisément pour vocation de faire **émerger** des **clusters** (sous-groupes d'entités fortement connectées), tout en abaissant la plupart des pondérations hors cluster. Si, dès le départ, le réseau se retrouve avec trop de liaisons, le **temps** nécessaire pour que ces liaisons s'affaiblissent peut s'allonger considérablement.

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)].$$

Il faut de nombreuses itérations pour faire **redescendre** $\omega_{i,j}$ à un niveau quasi nul quand le réseau est $O(N_{\text{img}} N_{\text{txt}})$.

E. Conclusion

Si **toutes** les images se relient à **tous** les textes sans **filtrage**, on assiste rapidement à une **prolifération** des liaisons $\omega_{i,j}$. Les conséquences sont multiples :

613. **Complexité** de calcul accrue, requérant $O(N_{\text{img}} \times N_{\text{txt}})$ évaluations de synergie S et de mises à jour par itération.

614. **Risque** de densification, rendant le **SCN** difficile à interpréter et masquant l'émergence de clusters clairs.

615. **Convergence** ralentie, le DSL doit "se battre" pour faire baisser la majorité des liaisons moyennes ou inutiles.

Pour éviter cette situation, il est souvent indispensable de restreindre, dès le départ ou dynamiquement, le **nombre** de liaisons potentiellement actives :

- par des **heuristiques** de parcimonie (k plus proches voisins, ϵ -radius) ;
- par un **filtrage** initial (seules les paires image–texte ayant un certain score minimal sont considérées) ;
- par des **mécanismes** de compétition ou d'inhibition (chapitre 7), qui limitent le nombre de connexions admissibles pour chaque entité.

Ainsi, la structure du SCN multimodal demeure gérable, tant du point de vue **computationnel** que de celui de la **lisibilité** et de la qualité de l'**auto-organisation**.

8.6.2.2. Heuristiques : k plus proches voisins multimodaux, ϵ -rayon, ou recuit (Chap. 7.3)

Lorsqu'un **DSL** (Deep Synergy Learning) traite simultanément plusieurs **modalités** (vision, audio, texte, capteurs, etc.), le risque de prolifération de liaisons $\omega_{i,j}$ devient particulièrement aigu. En effet, si le SCN tente d'évaluer la **synergie** $S(i,j)$ pour **toutes** les paires (i,j) issues d'un large ensemble d'entités multimodales, la complexité peut s'élèver en $O(n^2)$. Au-delà du coût de calcul, une telle densité compromet la *lisibilité* des clusters et la *qualité* de l'auto-organisation. Diverses **heuristiques** se révèlent alors essentielles pour limiter le **nombre** de liens effectivement considérés ou pour maintenir la dynamique DSL hors de minima locaux mal configurés. Parmi ces heuristiques, on trouve notamment :

- Le **k plus proches voisins** (k -NN) multimodal,
- Le **ϵ -rayon** (ou ϵ -ball) limitant les connexions à un certain périmètre de distance,
- Le **recuit simulé** (réexpliqué au *Chap. 7.3* sur les méthodes d'optimisation stochastiques).

A. k plus proches voisins multimodaux

Le **k -NN** (k plus proches voisins) propose de **restreindre** la création (ou la mise à jour) de liaisons $\omega_{i,j}$ aux seuls "voisins les plus proches" selon une certaine **métrique** multimodale. Concrètement, pour chaque entité \mathcal{E}_i (qu'elle soit image, texte ou segment audio), on ne retient que les k entités \mathcal{E}_j minimisant une distance $d(\mathcal{E}_i, \mathcal{E}_j)$. Les liaisons hors de ce petit voisinage sont fixées à 0 ou négligées.

Le **gain** est double. D'une part, la complexité de calcul est réduite de $O(n^2)$ à $O(nk)$, chaque entité ne maintenant un lien qu'avec k voisins, ce qui optimise les ressources de traitement. D'autre part, le réseau devient **plus sparse**, avec une densité de liaisons sensiblement diminuée, ce qui facilite la **mise à jour** DSL et améliore la **lisibilité** des clusters, permettant une structuration plus claire des connexions.

L'approche **multimodale** suppose qu'une entité \mathcal{E}_i puisse porter des composantes $\mathbf{x}_i^{(\text{vision})}, \mathbf{x}_i^{(\text{texte})}, \mathbf{x}_i^{(\text{audio})}$, etc. On définit alors une **distance** globale

$$d(\mathcal{E}_i, \mathcal{E}_j) = \alpha_1 d_{\text{vision}}(\mathbf{x}_i^{(\text{vis})}, \mathbf{x}_j^{(\text{vis})}) + \alpha_2 d_{\text{texte}}(\mathbf{x}_i^{(\text{txt})}, \mathbf{x}_j^{(\text{txt})}) + \dots$$

où $\alpha_1, \alpha_2, \dots$ pondèrent l'importance relative de chaque modalité. Une fois cette distance définie, pour chaque \mathcal{E}_i on recherche les k entités \mathcal{E}_j les "moins distantes", et on autorise uniquement $\omega_{i,j}$ pour ces paires.

La **sparsification** induite par le k -NN **soulage** fortement le SCN. La mise à jour $\omega_{i,j}$ ne s'effectue plus sur l'entièreté des couples (i,j) , mais seulement sur $O(nk)$. De plus, le réseau évite de se noyer dans des liaisons moyennes.

Toutefois, la **pertinence** de k -NN dépend de la **cohérence** de la distance d . Si celle-ci n'est pas adaptée (mauvaises projections, déséquilibres entre modalités), on peut *oublier* des liens synergiques rares mais importants. D'où la nécessité de paramétrier soigneusement $\alpha_1, \alpha_2, \dots$ et d'ajuster la dimension des embeddings.

B. ϵ -rayon

Une alternative au k-NN consiste à définir, pour chaque entité \mathcal{E}_i , un **voisinage** $N_\epsilon(i)$ composé des entités \mathcal{E}_j satisfaisant

$$d(\mathcal{E}_i, \mathcal{E}_j) \leq \epsilon,$$

où $\epsilon > 0$ est un **paramètre** fixant le “rayon” de connectivité. Seules les paires (i, j) pour lesquelles $d(\mathcal{E}_i, \mathcal{E}_j) \leq \epsilon$ sont susceptibles de voir leur liaison $\omega_{i,j}$ mise à jour et s’élèver. Au contraire, si la distance est trop grande, on met $\omega_{i,j} = 0$.

Les **avantages** de cette approche sont multiples. La **sélectivité** de l’entourage permet de limiter les connexions aux entités réellement **proches** dans l’espace multimodal, ce qui évite de calculer la synergie pour des paires trop éloignées, à condition que le paramètre ϵ soit bien calibré. L’**interprétation géométrique** offre une visualisation intuitive du réseau. Chaque entité, représentée dans un espace vectoriel après projection, définit une **boule** de rayon ϵ , et seules les entités situées à l’intérieur de cette hypersphère sont considérées comme des voisines potentielles. Enfin, le **pilotage de la densité** du graphe est directement influencé par la valeur de ϵ . Un rayon plus grand favorise un réseau plus dense, tandis qu’un ϵ plus restreint renforce la **sparsification**, optimisant ainsi l’**auto-organisation** du DSL en régulant le nombre de connexions actives.

Avec un rayon ϵ , le **nombre** de voisins peut varier d’une entité à l’autre en fonction de la distribution des points. k-NN, au contraire, fixe un **nombre** k identique pour chaque entité, ce qui peut entraîner un déséquilibre.

Une entité située dans une zone de forte densité risque d’accumuler plusieurs voisins très proches, tandis qu’une autre en zone de faible densité pourrait en avoir très peu. Pour concilier ces méthodes, on peut combiner les approches en utilisant un rayon ϵ afin d’éviter les distances excessives, tout en imposant une limite sur le nombre de voisins afin de prévenir une explosion de connexions dans les régions les plus denses.

C. Recuit (Chap. 7.3) pour surmonter les minima locaux

Le **recuit simulé** consiste à **injecter** un **bruit** stochastique dans la mise à jour $\omega_{i,j}$. À chaque itération t , on ajoute par exemple un terme $\xi_{i,j}(t)$ de variance contrôlée par une “température” $T(t)$. La mise à jour prend la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t).$$

où $\sigma(t) = T(t)$ décroît au fil du temps (refroidissement).

Même en utilisant un voisinage restreint (k-NN, ϵ -rayon), le **DSL** peut se retrouver bloqué dans un **minimum local** où certains liens $\omega_{i,j}$ se stabilisent alors qu’il existerait une configuration plus globale, plus cohérente, si l’on acceptait de “briser” quelques associations établies au profit d’autres. L’injection de bruit incite le système à **explorer** des configurations “moins immédiates” et à franchir des barrières locales.

Il est tout à fait possible de limiter la dynamique aux paires (i, j) tombant dans le rayon ϵ (ou dans les k plus proches voisins) *et* d’appliquer un recuit sur ces paires sélectionnées. Cela réduit considérablement le **nombre** de liaisons actives, tout en permettant à la structure d’évoluer par perturbations aléatoires quand la synergie reste dans une zone ambivalente.

Conclusion

Dans la construction d’un **SCN** multimodal, il est souvent vital de **réduire** ou **perturber** la dynamique DSL pour empêcher la prolifération des liaisons, améliorer la sparsification et sortir de configurations imparfaites. Trois approches majeures apparaissent :

616. **k plus proches voisins multimodaux.**

En conservant uniquement les k entités jugées les plus proches (toutes modalités confondues), on **limite** la connectivité à $O(nk)$. C’est une façon efficace d’avoir un **graphe** plus économique et plus rapide à mettre à jour.

617. ϵ -rayon.

Au lieu de k , on fixe un rayon ϵ . Toute entité \mathcal{E}_i ne se connecte qu'aux entités situées dans une **balle** de rayon ϵ . Ce mécanisme assure une “cohésion” locale, naturellement interprétable d'un point de vue géométrique dans l'espace des features multimodales.

618. Recuit (chap. 7.3).

Même avec un voisinage restreint, le DSL peut se figer dans un arrangement localement stable mais sous-optimal. Le **recuit simulé** — c'est-à-dire l'ajout d'un bruit stochastique décroissant au fil du temps — permet au réseau de franchir certaines “barrières” d'énergie, révélant des **configurations** plus globalement cohérentes.

Ces heuristiques, souvent **combinées** dans un système, offrent un **cadre** permettant de garder la dynamique DSL sous contrôle (pas de gaspillage de calcul) et assez **souple** (via le bruit ou la limitation de distance) pour laisser **émerger** des clusters multimodaux nets, tout en **éitant** l'explosion combinatoire.

8.6.2.3. Surveiller la scalabilité

Dans un **DSL** (Deep Synergy Learning) orienté multimodal, la question de la **scalabilité** se révèle déterminante dès lors que le nombre n d'entités — qu'il s'agisse d'images, de segments audio, de textes ou d'autres flux — devient important. Chaque entité \mathcal{E}_i est susceptible de se lier à de nombreuses autres entités \mathcal{E}_j , et la **logique** du SCN (Synergistic Connection Network) peut entraîner un nombre de liaisons $\omega_{i,j}$ potentiellement en $O(n^2)$. Sans mesures de régulation, les ressources de calcul et de stockage sont rapidement saturées, rendant l'auto-organisation du DSL non viable à grande échelle. La présente section 8.6.2.3 met en relief l'importance de **surveiller** et de **maîtriser** la complexité dès lors que plusieurs modalités coexistent, et expose quelques indicateurs et approches pour y parvenir.

A. Complexité Algorithmique et Nombre de Liaisons

En théorie, si l'on cherche à évaluer la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pour *toutes* les paires (i, j) d'entités, on effectue $O(n^2)$ comparaisons. Dans un système multimodal, un même objet (par ex. une image) peut se comparer non seulement à d'autres images, mais aussi à tous les segments audio et textuels, multipliant les synergies à calculer. On se retrouve ainsi avec :

$$\text{nombre total de paires} \approx N_{\text{img}}^2 + N_{\text{txt}}^2 + N_{\text{aud}}^2 + 2N_{\text{img}}N_{\text{txt}} + 2N_{\text{img}}N_{\text{aud}} + 2N_{\text{txt}}N_{\text{aud}},$$

ce qui peut grandement dépasser la capacité d'un simple algorithme en $O(n^2)$ si les flux sont massifs.

En l'absence de mécanismes de **sparsification**, la dynamique DSL peut maintenir un volume **conséquent** de liaisons $\omega_{i,j}$ d'intensité moyenne, ne décroissant pas suffisamment vite. Cela nuit à la **lisibilité**, car un réseau ultra-dense ne met plus en évidence de *clusters* nets, une même entité se connectant à un très grand nombre d'autres. De plus, la **charge de mise à jour** augmente, puisque maintenir et actualiser chaque $\omega_{i,j}$ requiert un temps de calcul important à chaque itération, ce qui ralentit la convergence du système.

Un **DSL** multimodal est souvent **dynamique**, ce qui signifie que de nouveaux objets tels que des images, des textes ou des extraits audio peuvent arriver en flux continu. Si, à chaque insertion, on compare le nouvel objet à *toutes* les entités existantes, la charge croît de façon quasi-cubique en régime continu $O(n)$ vérifications à chaque arrivée, répétées plusieurs fois). La complexité peut rapidement **s'emballer**, rendant nécessaire une surveillance attentive du **temps** moyen d'une itération, du **nombre** de liaisons actives $\omega_{i,j}$, ainsi que de la **densité** du graphe et de la **taille** totale de la matrice ω .

B. Stratégies pour Gérer la Scalabilité

Comme développé en 8.6.2.2, il est possible de restreindre les connexions à un **voisinage** calculé a priori. Le **k-NN multimodal** consiste à ne conserver, pour chaque entité \mathcal{E}_i , que ses k plus proches voisins selon une distance

multimodale. Une autre approche repose sur le **ϵ -rayon**, où la connectivité est limitée aux entités \mathcal{E}_j satisfaisant la condition

$$d(\mathcal{E}_i, \mathcal{E}_j) \leq \epsilon.$$

Ces approches ont l'avantage de convertir un possible $O(n^2)$ en $O(n k)$ ou $O(n \log n)$ si l'on emploie des structures d'indexation (k-d trees, etc.). Sur le **plan mathématique**, cette sparsification entraîne un graphe plus léger, tout en préservant la dynamique DSL dans le voisinage local.

Lorsqu'il devient impossible de gérer un grand $\{\omega_{i,j}\}$ sur une seule machine, il est possible de **distribuer** la matrice ou la collection d'entités sur plusieurs nœuds HPC (High Performance Computing). Une première approche consiste à diviser l'ensemble $\{1, \dots, n\}$ en **sous-SCN**, chaque bloc étant traité indépendamment par un sous-réseau (voir Chapitre 5.7 sur la distribution). Ces sous-SCN effectuent ensuite des **synchronisations périodiques**, où les pondérations inter-blocs sont échangées à intervalles réguliers pour assurer la cohérence globale. Enfin, un **équilibrage** peut être nécessaire. Si un bloc devient trop lourd, soit en raison d'un trop grand nombre d'entités, soit par accumulation de connexions, il est scindé ou redistribué pour optimiser la charge de calcul.

Cette approche est très efficace si l'on structure les entités en clusters “relativement indépendants”, faisant que peu de liaisons inter-blocs doivent être maintenues.

Pour un **flux continu** d'entités, une **stratégie** incrémentale peut être adoptée. À l'arrivée d'une nouvelle entité \mathcal{E}_{new} , on ne compare celle-ci qu'à un **sous-ensemble** (k plus proches voisins, un échantillon aléatoire, un bloc HPC spécifique). Puis la dynamique DSL localise \mathcal{E}_{new} dans le graphe déjà existant sans relancer une phase exhaustive. Cette **sélectivité** évite des réinitialisations coûteuses et étend la **scalabilité** à des environnements en flux.

C. Indicateurs de Survie : Densité et Temps d'Itération

Pour “surveiller” la scalabilité, on peut définir quelques **indicateurs** clés au fil des itérations :

Densité Effective δ . On calcule

$$\delta(t) = \frac{1}{n(n-1)} \sum_{i \neq j} \mathbf{1}\{\omega_{i,j}(t) \neq 0\},$$

c'est-à-dire la proportion de liaisons non nulles. Si $\delta(t)$ reste, disons, en dessous de 1 % (ou 5 %, etc.), le réseau est très clairsemé, ce qui est souhaitable pour un traitement large-échelle.

Temps par Époque / Itération. À chaque “tour” de mise à jour, on note le **temps** d'exécution. Si ce temps grandit trop vite avec n , on sait que l'approche n'est pas scalable.

Qualité du Clustering ou de la Représentation. On peut aussi mesurer la **modularité** ou l'**indice d'homogénéité** des clusters. Si une sparsification trop sévère ruine la qualité, on fait un compromis en augmentant k ou ϵ .

D. Exemples Concrets de Contrôle

Dans un **SCN** multimodal, plusieurs mécanismes permettent de contrôler la structure du graphe et d'optimiser son évolution.

Un premier exemple repose sur un **ajustement dynamique** de k ou ϵ . Au début du processus, on fixe des valeurs relativement larges afin de permettre la formation de connexions pertinentes. Puis, à mesure que le réseau atteint un état plus stable, on réduit progressivement k ou ϵ pour favoriser la sparsification et ne conserver que les liens les plus significatifs.

Une seconde approche combine **recuit simulé et inhibition**. La température du recuit facilite l'exploration de nouvelles configurations, tandis que l'inhibition régule la compétition entre liaisons. On surveille la densité δ et, si celle-ci devient trop élevée, on intensifie l'inhibition ou on accélère la baisse de température pour éviter un surchargement du réseau.

Une autre solution consiste à organiser le système en **sous-SCN spécialisés** par modalité ou par blocs thématiques. Par exemple, un SCN peut être dédié aux entités visuelles, un autre aux entités textuelles, un troisième à l'audio, et un mét SCN se charge de relier ces différents sous-ensembles via des super-nœuds allégés. Cette structuration réduit la charge de calcul tout en préservant la connectivité globale du réseau.

Conclusion : Surveiller et Gérer la Scalabilité

Dans un **DSL** multimodal, le nombre d'entités \mathcal{E}_i peut croître très vite, entraînant une **densification** du réseau $\{\omega_{i,j}\}$. **Surveiller la scalabilité** signifie :

Observer la **densité** du graphe, le **temps** de mise à jour, la **taille** mémoire,

Limiter les liaisons actives (via k-NN, ϵ -rayon, distribution HPC, etc.),

Assurer que la qualité des clusters reste satisfaisante (ou du moins “assez bonne”) malgré la reduction du nombre de liaisons.

Cette approche garantit que le **SCN** demeure exploitable pour des volumes massifs ou des flux en continu, préservant ainsi la **philosophie** d'auto-organisation du DSL sans compromettre les ressources ni l'efficacité.

8.6.3. Synergie Tri-Modal (Vision–Langage–Audio)

Dans les systèmes **multimodaux**, il arrive souvent que l'on souhaite exploiter **trois flux** simultanément. Une **image** (ou une séquence vidéo), un **segment audio** (enregistrement sonore, musique, paroles) et un **texte** (sous-titres, transcription, métadonnées). Le **DSL** (Deep Synergy Learning) peut alors dépasser la simple synergie *binaire* (image–texte, texte–audio, ou audio–image) pour calculer un **score “triple”** lorsqu'il existe une convergence entre les **trois modalités**. Cette synergie tri-modale permet d'identifier des **coïncidences** plus riches (ex. un passage vidéo, une phrase correspondante, et un signal audio concordant) et de renforcer la **cohérence** du réseau face à des phénomènes multimédias complexes (ex. un film sous-titré, un concert enregistré avec commentaire textuel, etc.).

8.6.3.1. Possibilité de calculer un “score triple” si un segment audio, un embedding image et un texte concordent

Lorsqu'un **DSL** (Deep Synergy Learning) vise à gérer **plus de deux** modalités simultanément, par exemple **audio**, **image** et **texte**, il peut être pertinent de dépasser la seule logique binaire $S(i,j)$ pour définir, dans certains cas, un **score de synergie à trois**. On introduit alors $S^{(3)}(\mathcal{E}_a, \mathcal{E}_v, \mathcal{E}_t)$ qui quantifie la **concordance** entre un segment audio, un extrait visuel et un passage textuel.

Cette possibilité ouvre la voie à une **coopération** tri-modale permettant de repérer — dans le même laps de temps ou la même scène — l'**harmonie** entre ces trois sources (par exemple un objet filmé + son bruit spécifique + la légende textuelle décrivant l'objet).

A. Formalisation Mathématique d'un Score Triple

Dans le **DSL** traditionnel, on traite la synergie **par paires** $S(i,j)$. Pour un système tri-modal, on pourrait commencer par combiner les **scores binaires** déjà définis entre paires :

$$S^{(2)}(\text{audio}, \text{vision}), \quad S^{(2)}(\text{audio}, \text{texte}), \quad S^{(2)}(\text{vision}, \text{texte}).$$

Une **fonction** F prend alors ces trois valeurs en entrée et en déduit un **score global** :

$$S^{(3)}(a, v, t) = F(S^{(2)}(a, v), S^{(2)}(a, t), S^{(2)}(v, t)).$$

On peut imaginer plusieurs formes de F . Par exemple :

Additive :

$$S^{(3)}(a, v, t) = \alpha_1 S^{(2)}(a, v) \alpha_2 S^{(2)}(a, t) \alpha_3 S^{(2)}(v, t)$$

où $\alpha_1, \alpha_2, \alpha_3$ sont des poids.

Multiplicative :

$$S^{(3)}(a, v, t) = [S^{(2)}(a, v)] \times [S^{(2)}(a, t)] \times [S^{(2)}(v, t)],$$

un score élevé n'apparaissant que si **toutes** les synergies binaires sont elles-mêmes fortes.

Min ou Médiane :

$$S^{(3)}(a, v, t) = \min\{S^{(2)}(a, v), S^{(2)}(a, t), S^{(2)}(v, t)\} \text{ ou } \text{median}\{\dots\},$$

traduisant la contrainte qu'il faut un “socle commun” de toutes les paires pour afficher une bonne synergie tri-modale.

Cette **combinaison** binaire—>tri-modale s'avère assez **simple** à mettre en œuvre, car on suppose déjà avoir calculé $S^{(2)}$ entre chaque paires de modalités. Toutefois, elle peut rater certaines nuances si la “tri-synergie” ne se résume pas au simple produit ou à la simple somme des synergies binaires (certains signaux n'apparaissent qu'en conjonction).

Une autre approche, plus proche de la **théorie de l'information**, considère que l'on dispose de variables aléatoires $X_{\text{audio}}, X_{\text{image}}, X_{\text{texte}}$. La **co-information** à trois variables, inspirée du formalisme de l'information mutuelle, se définit en combinant entropies et entropies conjointes :

$$\begin{aligned} & \text{CoInfo}(X_{\text{aud}}, X_{\text{img}}, X_{\text{txt}}) \\ &= H(X_{\text{aud}}) + H(X_{\text{img}}) + H(X_{\text{txt}}) - [H(X_{\text{aud}}, X_{\text{img}})H(X_{\text{aud}}, X_{\text{txt}})H(X_{\text{img}}, X_{\text{txt}})] \\ & \quad + H(X_{\text{aud}}, X_{\text{img}}, X_{\text{txt}}) \end{aligned}$$

où H désigne l'entropie (ou entropie conjointe) selon la définition de la théorie de l'information.

- Si $\text{CoInfo} > 0$, on considère qu'il existe une **redondance** partagée par les trois variables ; si elle est négative, on parle d'**interaction synergique** plus complexe.
- D'un point de vue **DSL**, on peut alors définir un “score triple” $S^{(3)}$ proportionnel à $\max\{0, \text{CoInfo}\}$ ou toute autre dérivation, afin d'implémenter un mécanisme où la synergie tri-modale est élevée lorsque l'audio, l'image et le texte s'avèrent très fortement corrélés.

Une fois un “score triple” $S^{(3)}(a, v, t)$ établi, on peut imaginer étendre la **dynamique** DSL à des **hyper-liens** (voir plus loin dans l'ouvrage, ou Chap. 12 si l'on traite la synergie n -aire). Par exemple, on ne manipule plus seulement des pondérations $\omega_{i,j}$ pour deux entités, mais aussi des coefficients $\omega_{a,v,t}$ pour un triplet (a, v, t) . La mise à jour :

$$\omega_{(a,v,t)}(t+1) = \omega_{(a,v,t)}(t) + \eta [S^{(3)}(a, v, t) - \tau \omega_{(a,v,t)}(t)]$$

renforcerait la liaison “tri-angulaire” si la synergie tri-modale est forte, ouvrant la possibilité de **clusters** n -aires (ex. un hyper-nœud englobant la portion audio, l'extrait visuel et la phrase textuelle qui décrivent un même évènement).

B. Exemples et Bénéfices Concrets

Prenons le cas d'un **documentaire** où :

- L'**audio** correspond à la voix off expliquant un concept,
- Les **images** (vidéo) montrent un plan rapproché de ce concept,
- Les **sous-titres** (ou résumé textuel) explicitent le contenu.

Si l'on définit un score $S^{(3)}(\text{audio}, \text{visuel}, \text{texte})$ important lorsque les trois coïncident à la fois dans la dimension temporelle (même séquence) et dans la dimension sémantique (même concept), le DSL pourra créer un **hyper-lien** $\omega_{(\text{audio}, \text{visuel}, \text{texte})}$ renforcé. Cela facilite l'indexation — on sait que cet ensemble tri-modal décrit un *même moment clé* du documentaire.

Dans un **clip publicitaire** :

- La piste **audio** (chanson ou jingle),
- Le composant **vidéo** (images dynamiques de produits),
- Le **slogan textuel** (affiché à l'écran à un instant donné).

Un “score triple” détecterait la concordance de trois signaux. La portion musicale, le plan vidéo illustrant le produit et le texte du slogan ou la phrase clé. L'**auto-organisation** du SCN, en exploitant ce score, permet de repérer (ou faire émerger) le moment où tous ces éléments coïncident parfaitement, et ainsi de “tagger” cette scène comme un *macro-événement* tri-modal.

Si l'on veut **retrouver** un évènement spécifique dans un enregistrement massif (spectacle, concert, conférence), on peut rechercher un certain *mot* prononcé sous forme de texte, une *image* ou un plan vidéo avec un cadrage particulier, ou encore un *son* distinct correspondant à une ambiance ou un instrument.

Le score tri-modal peut servir de *filtre* fort ; on ne considère un évènement détecté que lorsque l'audio, la vision et la transcription textuelle s'avèrent mutuellement cohérents. Cela réduit les faux positifs si un canal est bruité ou ambigu.

C. Limites et Considérations

Le **score triple** $S^{(3)}(a, v, t)$ suppose une évaluation pour **chaque** triplet (a, v, t) . Si l'on a N_{aud} segments audio, N_{vid} frames/segments vidéo, N_{txt} blocs de texte, le total peut atteindre $O(N_{\text{aud}} \times N_{\text{vid}} \times N_{\text{txt}})$. Sans sparsification ni heuristiques, le coût peut devenir prohibitif.

S'il existe déjà une **dynamique** DSL sur :

- $\omega_{(\text{audio}, \text{vision})}$,
- $\omega_{(\text{audio}, \text{texte})}$,
- $\omega_{(\text{vision}, \text{texte})}$,

on peut juger parfois superflu d'introduire un **score tri-modal** explicite, car l'**émergence** d'un cluster reliant (audio, vision, texte) “naturellement” provient du renforcement de ces trois liaisons binaires. Cependant, dans certains cas, la prise en compte d'une **cohérence globale** (parfois non réductible à la seule addition/produit binaires) apporte des capacités plus expressives (ex. on ne veut un cluster qu'en cas de triple validation stricte).

Pour **implémenter** la tri-synergie de manière organique au sein d'un SCN, on peut envisager un **hypergraphe** où un hyper-lien connecte simultanément (a, v, t) . La **mise à jour** se fait selon :

$$\omega_{(a, v, t)}(t + 1) = \omega_{(a, v, t)}(t) + \eta [S^{(3)}(a, v, t) - \tau \omega_{(a, v, t)}(t)].$$

Cette approche, certes plus complexe à coder, peut **révéler** des schémas tri-modaux plus riches (voir Chap. 12 si l'ouvrage discute de la synergie *n*-aire et des hyper-liens).

Conclusion

Le **score triple** $S^{(3)}(a, v, t)$ représente une **extension** naturelle de la logique DSL au-delà des simples paires d'entités. Sur le **plan mathématique**, on peut :

Combiner les synergies binaires $S^{(2)}$ pour construire un **score** tri-modal ;

Recourir à une mesure de **co-information** ou d'entropie conjointe à trois variables, permettant d'évaluer la **corrélation** simultanée des trois canaux (audio, vision, texte) ;

Implémenter un **hyper-lien** $\omega_{(a,v,t)}$ mis à jour par une règle DSL adaptant ω selon $S^{(3)}(a, v, t)$.

Cette possibilité de **tri-synergie** s'avère décisive pour des scénarios où le **contexte** n'apparaît clairement qu'en présence de **trois** flux synchronisés. Par exemple, *une image filmée, un son audible et une phrase prononcée ou décrite textuellement* qu'il convient d'associer comme un **même** événement. Ainsi, le DSL peut aller au-delà de l'**agrégation binaire** et promouvoir une **coopération** véritablement tri- (voire multi-) modale, créant des **clusters** ou **hyper-clusters** plus expressifs et plus sélectifs.

8.6.3.2. Approches additive, multiplicative ou min(…) pour combiner les synergies binaires

Lorsque l'on souhaite mesurer la **synergie** d'un **groupe** de plusieurs entités, par exemple trois modalités audio–image–texte, à partir des **scores binaires** déjà définis $S(i, j)$, se pose la question de savoir comment **fusionner** ces valeurs binaires pour obtenir un **score global**. Trois grandes approches s'imposent souvent, l'**additive**, la **multiplicative** et la **min** (ou d'autres “règles de consensus”). Le choix de l'une ou l'autre dépend de la **sémantique** recherchée et du **comportement** souhaité lorsque l'on combine plusieurs synergies.

A. Approche Additive

Dans l'approche **additive**, on définit la **synergie n-aire** (ou tri-modale, etc.) d'un sous-ensemble $\{i_1, i_2, \dots, i_m\}$ comme la **somme** (ou la moyenne) des synergies **binaires** qui relient les paires (i_a, i_b) à l'intérieur du groupe :

$$S_{\text{add}}(\{i_1, \dots, i_m\}) = \sum_{1 \leq a < b \leq m} S(i_a, i_b) \quad (\text{somme simple})$$

ou

$$S_{\text{add}}(\{i_1, \dots, i_m\}) = \frac{2}{m(m-1)} \sum_{1 \leq a < b \leq m} S(i_a, i_b) \quad (\text{moyenne, normalisée par le nombre de paires}).$$

Cette approche **pondère** toutes les paires (i_a, i_b) . Si **toutes** les paires sont fortes, alors la somme (ou moyenne) est grande ; si l'on a du bon et du moyen, on obtient un score intermédiaire, car tout se **cumule**. On peut l'interpréter comme une mesure de la “**cohésion globale**” du groupe où quelques paires faibles peuvent être **compensées** par un grand nombre de paires fortes.

L'un des principaux **avantages** réside dans la mise en œuvre simple et la **lecture** intuitive, basée sur la somme ou la moyenne des synergies binaires. Cette approche reflète l'idée qu'un groupe peut rester **globalement** synergique si la majorité des paires présentent une forte cohésion, même si certaines connexions restent moyennes.

La **limite** de cette méthode est qu'elle est insensible à la faiblesse d'une **unique** paire. Une connexion particulièrement faible peut être noyée dans la somme globale, ce qui rend cette approche moins stricte lorsque l'on souhaite garantir une “cohérence parfaite” entre toutes les paires.

B. Approche Multiplicative

Dans l'approche **multiplicative**, la synergie n-aire s'obtient par le **produit** (ou le **produit transformé**) des synergies binaires :

$$S_{\text{mult}}(\{i_1, \dots, i_m\}) = \prod_{1 \leq a < b \leq m} S(i_a, i_b).$$

Pour des raisons pratiques (éviter sous-flux ou sur-flux), on procède souvent en **logarithmes** :

$$\ln S_{\text{mult}}(\{i_1, \dots, i_m\}) = \sum_{1 \leq a < b \leq m} \ln [S(i_a, i_b)].$$

Le **produit** exprime un **principe** de “tout ou rien” où une seule paire $S(i_a, i_b)$ proche de zéro fait chuter rapidement le **produit total**, indiquant ainsi que chaque **paire** doit être suffisamment forte pour garantir un score élevé. C'est une stricte exigence où *toutes* les binaires doivent coopérer.

L'un des principaux **avantages** de cette approche est qu'elle satisfait un **principe fondamental** affirmant que si un groupe est réellement synergique, *toutes* les paires doivent l'être. Elle se révèle particulièrement adaptée aux scénarios nécessitant une **cohérence stricte**, où l'on exige que *chaque* lien binaire maintienne un niveau de synergie élevé.

Toutefois, cette méthode présente une **limite importante**, car elle se montre extrêmement **sensible** au maillon le plus faible. Si une seule paire $((a, b))$ possède une synergie très faible, alors le produit global s'effondre. Cette rigidité peut conduire à ignorer des groupes qui, bien qu'étant bons en moyenne, comportent *une* connexion moins optimale.

C. Approche min (ou “min-like”)

Avec la règle $\min(\dots)$, la synergie n-aire vaut la plus faible des synergies binaires du sous-groupe :

$$S_{\min}(\{i_1, \dots, i_m\}) = \min_{1 \leq a < b \leq m} S(i_a, i_b).$$

Cela veut dire que, pour considérer un ensemble comme **synergique**, *toutes* les paires doivent être bonnes au même niveau, sinon la min tombe sur la paire la plus faible.

On généralise encore l'idée de **tout ou rien** en imposant que la synergie n-aire ne puisse dépasser la plus faible synergie binaire interne. Cette approche constitue la variante la plus stricte en matière de cohésion des liens. Une seule paire franchement faible *neutralise* la synergie globale.

L'un des principaux **avantages** de cette approche réside dans sa **lecture explicite**, puisqu'elle garantit que le score global ne dépasse jamais la **moins bonne** synergie interne. Elle se montre idéale dans les cas où il est impératif que *toutes* les connexions soient parfaitement alignées, sans possibilité de compromis ou de tolérance aux écarts.

Cependant, cette méthode présente une **limite majeure**, car dans un groupe comportant un grand nombre de paires, une seule dysharmonie suffit à faire chuter l'ensemble du score. Elle ne permet pas de prendre en compte une éventuelle **compensation** par les autres paires, ce qui peut être problématique dans des structures complexes où des ajustements locaux existent sans altérer la cohérence générale.

D. Choix Pratique et Contexte

La **nature** de la fonction de combinaison dépend du **contexte** :

- **Additif** (“moyenne”) : on veut un ressenti **global**. Un groupe peut être jugé positif même si certaines paires sont juste correctes.
- **Multiplicatif** : une **exigence** forte d'homogénéité entre toutes les paires. Un seul maillon faible *pénalise* tout le groupe.
- **min** (plus drastique) : on interdit qu'*une seule* paire soit faiblement synergique.

En **fusion multimodale**, on peut préférer un **produit** (ou un **min**) si l'on veut exiger l'accord simultané de tous les canaux (par ex. texte-image-audio doivent tous confirmer le même concept). Si un canal est en désaccord, le groupe n-aire est jugé inconsistant.

Pour un sous-ensemble $\{i_1, \dots, i_m\}$, il faut **accéder** aux synergies binaires $S(i_a, i_b)$ pour $\binom{m}{2}$ paires. Ensuite :

- **Additif** : on **somme** ou **moyenne**.
- **Multiplicatif** : on **multiplie** ou on **somme** les logs.

- min : on prend la plus petite valeur.

Ces opérations sont assez peu coûteuses si $\binom{m}{2}$ reste modéré.

E. Conclusion

En **DSL** n-aire (ou tri-modale), où l'on souhaite vérifier la **cohérence** d'un sous-groupe $\{i_1, \dots, i_m\}$ à partir de **scores binaires** $S(i_a, i_b)$, trois grandes règles s'imposent pour combiner ces synergies binaires :

619. Additif :

- $S_{\text{add}} = \sum S(i_a, i_b)$ ou la moyenne,
- Traduit une **vision globale** qui tolère quelques paires moins fortes.

• Multiplicatif :

- $S_{\text{mult}} = \prod S(i_a, i_b)$ ou $\exp[\sum \ln(S)]$,
- Exige l'**accord** de toutes les paires (un maillon faible compromet tout).

• min :

- $S_{\text{min}} = \min S(i_a, i_b)$,
- Principe très **strict** : le score n-aire reflète la **plus petite** liaison binaire interne.

Leur choix dépend de la **philosophie** recherchée. Il s'agit de déterminer si l'on privilégie un **score moyen**, une approche basée sur un **principe tout ou rien**, ou encore une **borne minimale commune** garantissant une cohésion stricte au sein des associations établies. Cela peut s'intégrer soit dans une **extension** du DSL aux hyper-liens (avec une règle $\omega_{(i_1, \dots, i_m)} \leftarrow \dots$), soit comme un **critère** d'évaluation de groupe dans un algorithme de clusterisation n-aire. Dans tous les cas, ces fonctionnels additive, multiplicative ou min(\dots) donnent des **comportements** et des **résultats** sensiblement différents, chacun adapté à un type de **fusion** sémantique.

8.6.3.3. Étude d'exemples (ex. scène d'un concert, documentaire avec sous-titres)

Les principes du **DSL** (Deep Synergy Learning), dans un **SCN** (Synergistic Connection Network) où cohabitent plusieurs modalités (audio, vidéo, texte...), se concrétisent aisément lorsqu'on considère des situations **réelles**. Par exemple, la captation d'un **concert** (audio + vidéo) ou la production d'un **documentaire** (vidéo + audio + sous-titres textuels). L'idée est de montrer de manière **opérationnelle** comment les entités issues de multiples flux se **fusionnent** dans un même réseau, et comment la **dynamique** $\{\omega_{i,j}\}$ fait émerger des **clusters** multimodaux cohérents.

A. Exemple : Scène d'un Concert (Audio + Vidéo)

Considérons une **scène de concert** filmée dans laquelle deux flux principaux interviennent.

Le **flux vidéo** est composé d'une suite d'images, ou plus fréquemment de *frames clés* sélectionnées à partir de l'enregistrement original. Ces frames peuvent être segmentées en *patchs* ou *plans*, puis transformées en **descripteurs** sous forme de vecteurs obtenus par des réseaux neuronaux convolutionnels (CNN).

Le **flux audio** capture la musique, incluant les instruments, les voix et les applaudissements. Ce signal sonore est souvent découpé en *trames* ou *segments* de quelques secondes, ensuite convertis en **features** telles que les coefficients cepstraux en fréquence de Mel (MFCC), les spectrogrammes ou encore des embeddings issus de modèles d'apprentissage profond spécialisés dans l'analyse acoustique.

Dans un **SCN** multimodal, chaque segment \mathcal{E}_v (vidéo) et \mathcal{E}_a (audio) est **intégré** comme un **nœud**. Dès lors, la **synchronicité** ou la **pertinence** qu'ont certains segments à coïncider sur un même moment (ou un même événement musical) se traduit par une **synergie** $S(\mathcal{E}_v, \mathcal{E}_a)$.

Une **fonction** audio–vidéo peut prendre en compte :

$$S(\mathcal{E}_v, \mathcal{E}_a) = \alpha \text{correlation_temporelle}(\mathcal{E}_v, \mathcal{E}_a) + \beta \text{similarite_features}(\mathcal{E}_v, \mathcal{E}_a),$$

où la composante `correlation_temporelle` vérifie si le segment audio coïncide en temps avec le plan vidéo (ex. le solo de guitare a lieu de 1:50 à 2:05, la caméra montre le guitariste à la même période), tandis que `similarite_features` évalue la *complémentarité* plus *sémantique* (par exemple, détection d'un musicien + analyse de la présence d'une guitare dans l'audio).

Le **DSL** met à jour chaque liaison $\omega_{(v,a)}$ suivant la formule usuelle :

$$\omega_{(v,a)}(t+1) = \omega_{(v,a)}(t) + \eta [S(\mathcal{E}_v, \mathcal{E}_a) - \tau \omega_{(v,a)}(t)].$$

Si le segment \mathcal{E}_v (ex. gros plan sur le guitariste) *coïncide* régulièrement avec l'extrait \mathcal{E}_a (riff de guitare), la synergie demeure élevée et la pondération $\omega_{(v,a)}$ se stabilise à un niveau fort, **forçant** le DSL à **associer** ces deux entités.

Au **fil** des itérations, une organisation structurelle se met en place.

Les **plans vidéo** capturant le batteur ainsi que l'audio associé à un “batterie solo” convergeront vers un **cluster** commun. De même, les **plans** montrant le chanteur seront étroitement reliés à la **voix** correspondante, établissant ainsi des connexions fortes entre les segments visuels et sonores pertinents.

Cette **auto-organisation** permet au **SCN** de différencier divers segments d'un concert (intro, solos, transitions, applaudissements), clarifiant la **structure** de la vidéo musicale. Dans la pratique, on peut associer chaque cluster à un “**morceau**” ou un “**passage**” du concert, facilitant la *recherche* (“retrouve-moi la partie où le guitariste fait son solo”) ou le *montage* (on isole d'emblée les segments vidéo–audio les plus synchrones).

B. Exemple : Documentaire avec Sous-Titres (Vidéo + Audio + Texte)

Dans un **documentaire** complet, on trouve :

- Un **flux vidéo** (plans d'images),
- Un **flux audio** (voix off, interventions, musique d'ambiance),
- Des **sous-titres** ou un script texte ajoutés (souvent un fichier *.srt* contenant $\langle t_{\text{start}}, t_{\text{end}}, \text{texte} \rangle$).

Le **SCN** se dote alors de nœuds vidéo $\{\mathcal{E}_i^{(\text{vid})}\}$, nœuds audio $\{\mathcal{E}_j^{(\text{aud})}\}$ et nœuds texte $\{\mathcal{E}_k^{(\text{txt})}\}$.

On peut définir :

$$S_{\text{vid,aud}}(\mathcal{E}_i^{(\text{vid})}, \mathcal{E}_j^{(\text{aud})}), \quad S_{\text{aud,txt}}(\mathcal{E}_j^{(\text{aud})}, \mathcal{E}_k^{(\text{txt})}), \quad S_{\text{vid,txt}}(\mathcal{E}_i^{(\text{vid})}, \mathcal{E}_k^{(\text{txt})}).$$

Chacune reflète, par exemple :

- **Vidéo–Audio** : alignement temporel d'une scène + bruitage ou dialogue correspondant,
- **Audio–Texte** : transcription plus ou moins exacte de la parole,
- **Vidéo–Texte** : une légende/sous-titre décrivant l'image.

Si le **Chap. 8.6.3.1** évoque la possibilité d'un *score triple* $S^{(3)}(v, a, t)$, la pratique la plus simple demeure de **combiner** ces binaires (ex. additive, multiplicative, min) ou de se contenter de *paires*, puis de laisser la mise à jour DSL *interpénétrer* les trois flux. Ainsi, lorsque qu'un **texte sous-titre** $\mathcal{E}_k^{(\text{txt})}$ coïncide à la fois **temporellement** et

sémantiquement avec un segment **audio** $\mathcal{E}_j^{(\text{aud})}$, la pondération $\omega_{k,j}$ se renforce naturellement. Simultanément, si la **vision** capture la personne en train de parler, la pondération $\omega_{i,j}$ augmente lorsque $\mathcal{E}_i^{(\text{vid})}$ correspond à la même plage temporelle et / ou sémantique, que ce soit par détection du locuteur ou par reconnaissance faciale. Finalement, ces ajustements conduisent à la formation de **clusters multimodaux**, associant des segments de **vidéo**, des extraits **audio** et des **sous-titres**, chacun tourné autour d'un *même événement ou même contexte*.

Dans un processus de **chapitrage**, le réseau de synergie **SCN** peut faire émerger, à travers les pondérations ω , des **macro-nœuds** correspondant à des segments narratifs distincts. Par exemple, dans un documentaire, des chapitres clairement délimités peuvent apparaître, tels que “*présentation de la géologie*” ou “*interview de tel scientifique*”. L'**homogénéité** des données audio, textuelles et visuelles dans chaque bloc renforce cette structuration, facilitant l'interprétation de la *structure narrative globale*.

Dans le cas d'une **correction d'alignement**, un sous-titre $\mathcal{E}_k^{(\text{txt})}$ peut être légèrement *décalé* par rapport à son segment audio associé $\mathcal{E}_j^{(\text{aud})}$. Si ce décalage altère la **synergie audio–texte**, la pondération $\omega_{k,j}$ s'en trouve diminuée. Un **recuit simulé** (Chapitre 7.3) ou un ajustement local peut alors permettre de **réaligner** les segments et de *retrouver* la meilleure *fenêtre de correspondance*, optimisant ainsi la cohérence du lien et augmentant $\omega_{k,j}$.

La **fusion fiable** des canaux multimodaux permet de pallier les défaillances individuelles. Lorsqu'un canal est dégradé, comme une caméra de mauvaise qualité ou un micro qui grésille, les autres modalités peuvent compenser cette perte d'information. Le **SCN** s'auto-organise en renforçant les **clusters** où la somme ou le produit des synergies intermodales demeure élevée, garantissant ainsi une robustesse accrue du réseau.

La **structure multimodale explicite** offerte par le **SCN** met en évidence des **macro-nœuds**, révélant des “moments” ou “thèmes” multimédias. Des entités telles que *Concert — Morceau 1, Documentaire — Séquence interview* ou encore *Chat vidéo + Son miaulement + Sous-titre* se distinguent naturellement grâce à la densité des connexions ω . Cette organisation permet une représentation plus intuitive et exploitable du contenu analysé.

Les **applications** sont nombreuses. Dans le cas d'un **concert**, le système peut segmenter les morceaux, détecter les solos et identifier les musiciens à partir des liens renforcés entre les flux audio et vidéo. Dans un **documentaire**, l'indexation automatique permet un chapitrage précis, un alignement optimisé des sous-titres et une recherche avancée. Par exemple, il devient possible d'extraire une séquence où la voix off prononce un terme spécifique tout en affichant un lieu correspondant à l'écran.

Conclusion

L'examen de cas concrets — **concert** (audio + vidéo) et **documentaire** (audio + vidéo + texte) — montre comment un **SCN** multimodal exploite la **dynamique** du DSL pour :

- **Relier** segments audio, frames vidéo, éventuels textes (sous-titres),
- **Renforcer** ω quand ces flux s'alignent spatio-temporellement ou sémantiquement,
- **Former des clusters** ou **macro-nœuds** correspondant à des événements, scènes ou chapitres cohérents.

Sur le plan **mathématique**, on gère un **graph** où les nœuds sont répartis sur plusieurs **modalités**. Les synergies S peuvent être binaires ou tri-modales (voir 8.6.3.1), et la **mise à jour** ω (avec inhibition ou recuit) permet d'**extraire** les **coïncidences** et **corrélations** fortes. L'**apprentissage** en découle, révélant les portions audio–vidéo–texte associées à des thèmes, sous-séquences ou instants clés, et facilitant ainsi la **compréhension** et le **chapitrage** de contenus multimédia complexes.

8.7. Dynamique d'Auto-Organisation en Contexte Multimodal

Lorsque les entités manipulées (images, textes, sons, etc.) diffèrent grandement par leur nature, la **dynamique** d'auto-organisation du **DSL** (Deep Synergy Learning) doit néanmoins conserver sa **base** théorique. L'actualisation des pondérations $\omega_{i,j}$ suit la **même** formule qu'en contexte monomodal, mais en veillant à employer la fonction de **synergie** appropriée pour chaque paire d'entités. Les sections suivantes (8.7.2, 8.7.3) montreront comment cette mise à jour s'applique à un **SCN** (Synergistic Connection Network) où coexistent plusieurs modalités, et quelles **difficultés** spécifiques surgissent (oscillations, confusion sémantique, etc.).

8.7.1. Mise à Jour des Pondérations

8.7.1.1. Toujours la formule DSL classique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Dans la dynamique du **DSL** (Deep Synergy Learning), cette **règle de mise à jour** reste le socle principal, indépendamment du fait que l'on manipule un unique canal (texte, image...) ou des canaux multiples (multimodal). La présente section (8.7.1.1) rappelle :

- **La nature** du terme de synergie $S(i,j)$,
- **Le rôle** simultané du renforcement $\eta S(i,j)$ et de l'amortissement $\eta \tau \omega_{i,j}(t)$,
- **Comment** cette même équation s'applique, sans être modifiée en profondeur, même dans un contexte multimodal complexe.

A. Rappel mathématique de la règle

La mise à jour d'une liaison $\omega_{i,j}(t)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j dans un Synergistic Connection Network (SCN) procède selon une relation discrète qui s'apparente à une équation différentielle en temps discret. On part d'une règle fondamentale

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Cette équation peut se lire comme la somme d'un terme d'**incrément** positif lié à la synergie $S(i,j)$ et d'un terme de **décroissance** proportionnel à $\omega_{i,j}(t)$. Le facteur η représente un taux d'apprentissage, tandis que τ contrôle la part d'**amortissement** imposée à chaque liaison. On peut également réécrire la règle de mise à jour sous la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta S(i,j) - \eta \tau \omega_{i,j}(t).$$

La différence $\omega_{i,j}(t+1) - \omega_{i,j}(t)$ reflète la dérivée discrète de $\omega_{i,j}(t)$. Cette dérivée comporte un ajout, $\eta S(i,j)$, qui renforce la liaison si la synergie est positive, et un retrait, $\eta \tau \omega_{i,j}(t)$, qui la ramène vers zéro en l'absence de stimulation. Lorsqu'un couple (i,j) affiche un score de synergie élevé, le poids $\omega_{i,j}(t)$ croît jusqu'à l'équilibre $\omega_{i,j}^* \approx S(i,j)/\tau$. Cette valeur d'équilibre est approximative, car d'autres mécanismes, tels que l'inhibition ou le recuit, peuvent se surajouter. Quand la synergie devient faible ou nulle, le terme de décroissance prédomine et $\omega_{i,j}(t)$ chute jusqu'à s'annuler.

L'algorithme DSL consiste alors à itérer cette mise à jour pour toutes les liaisons $\omega_{i,j}$. L'ordre de mise à jour peut varier, mais la formule reste la même pour chaque couple (i,j) . Les itérations se poursuivent jusqu'à une forme de convergence, où chaque liaison se stabilise à un niveau dicté par la balance entre la synergie $S(i,j)$ et la dissipation $\tau \omega_{i,j}(t)$. D'un point de vue algorithmique, il est possible de traiter toutes les paires (i,j) en parallèle à chaque itération ou par lot (batch) successifs. On peut aussi envisager un mode streaming, où les pondérations se mettent à jour au fur et à mesure que les données arrivent.

Sur le plan énergétique, on peut décrire la configuration globale du réseau par une fonction

$$\mathcal{J}(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} (\omega_{i,j})^2,$$

où Ω regroupe l'ensemble des liaisons $\omega_{i,j}$. Cette fonction comprend un terme négatif $-\omega_{i,j} S(i,j)$ qui pousse chaque liaison vers une valeur élevée si la synergie est positive, et un terme de régularisation $\tau/2 \omega_{i,j}^2$ qui empêche les poids de croître indéfiniment. L'évolution de $\omega_{i,j}(t)$ selon la règle ci-dessus équivaut à opérer une descente sur \mathcal{J} , dans un style local et distribué. Chaque liaison s'efforce de minimiser sa contribution à \mathcal{J} , stabilisant $\omega_{i,j}$ là où la dérivée partielle par rapport à $\omega_{i,j}$ s'annule.

Cette dynamique aboutit, en pratique, à la formation de **clusters** où les entités ayant de fortes synergies se retrouvent interconnectées par des liaisons $\omega_{i,j}$ élevées, tandis que les entités moins compatibles voient leurs liens mutuels s'étioler. Dans un contexte multimodal, il devient possible de faire émerger des regroupements associant images, texte et audio autour d'un même concept. Le résultat final illustre la logique auto-organisée du DSL où chaque liaison croît ou décroît en fonction d'informations purement locales, jusqu'à produire un état global structurant un réseau de clusters pertinents.

B. Application inchangée pour le multimodal

La puissance du Deep Synergy Learning (DSL) réside dans le fait que la règle de mise à jour des liaisons $\omega_{i,j}$ reste strictement la même, même dans un cadre multimodal intégrant différents types d'entités. Concrètement, la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

ne subit aucune modification de forme. Cette invariance garantit que le noyau du DSL (son mécanisme auto-organisé de renforcement et de décroissance) demeure identique, quelle que soit la nature des paires d'entités mises en correspondance. L'élément qui devient plus sophistiqué concerne la définition de la synergie $S(i,j)$, qui doit être capable de gérer un large éventail de modalités.

Lorsque deux entités \mathcal{E}_i et \mathcal{E}_j appartiennent à la même modalité, on applique une fonction de similarité cohérente pour ce flux (par exemple un cosinus ou une distance exponentielle si les deux embeddings proviennent d'un CNN d'images). Lorsque les entités proviennent de modalités différentes, on applique un modèle ou un mapping spécifique correspondant à chaque paire, qu'il s'agisse de texte-texte, image-texte ou audio-texte.

Dans chaque cas, la logique interne du DSL (renforcer la liaison si la synergie se révèle élevée et appliquer une dissipation proportionnelle à $\omega_{i,j}$) ne change pas. La partie qui exige un travail supplémentaire réside dans l'établissement d'un module ou d'une fonction de similarité adéquate pour chacune des combinaisons de modalités.

Le SCN (Synergistic Connection Network) doit être informé du type de modalité en jeu lorsqu'il calcule la synergie $S(i,j)$. Différentes formules apparaissent selon la nature des paires :

- $S_{\text{img,img}}$. Lorsque les deux entités sont visuelles, on peut recourir à un cosinus entre leurs embeddings CNN ou à une distance RBF sur ces vecteurs.
- $S_{\text{txt,txt}}$. Quand il s'agit de texte vs. texte, il est fréquent d'utiliser un cosinus sur les embeddings textuels (Word2Vec, GloVe, BERT) ou bien une distance TF-IDF.
- $S_{\text{img,txt}}$. Pour relier l'image et le texte, on peut s'inspirer de mécanismes à la CLIP, en projetant l'embedding image et l'embedding texte dans un espace commun, puis en calculant un score de correspondance.
- $S_{\text{aud,txt}}$. Dans le cas audio vs. texte, on peut employer un alignement phonème-mot, des embeddings audio-texte préappris ou un modèle combinant Wav2Vec pour l'audio et BERT pour le texte.

Dès que la synergie associée à un couple (i,j) s'avère élevée à travers plusieurs itérations, la liaison $\omega_{i,j}(t)$ se renforce progressivement, traduisant une cohérence inter-modale repérée de façon répétée.

La même équation de base sert pour tous les flux, ce qui veut dire qu'on peut conserver deux constantes globales η et τ pour l'ensemble du réseau, ou ajuster le taux d'apprentissage η localement. Par exemple, si la similarité audio–audio tend à afficher des valeurs plus faibles que la similarité image–image, on peut accroître le coefficient $\eta_{\text{aud},\text{aud}}$ afin de compenser et de donner un poids similaire aux informations audio. On peut aussi normaliser, à l'avance, la fonction de synergie S pour chaque couple de modalités de sorte à la ramener dans un intervalle uniforme (souvent $[0,1]$). Cette uniformisation évite qu'une seule modalité domine systématiquement la construction du SCN.

Au final, le DSL applique toujours la même règle de renforcement/décroissance pour chaque pondération $\omega_{i,j}$. Le fait de brancher de nouveaux canaux (comme l'audio, la vidéo, le texte) n'oblige pas à repenser le cœur algorithmique du DSL. On se contente de définir ou de calibrer un module de similarité $S_{\text{mod}_a,\text{mod}_b}$ pour chaque paire de modalités mod_a et mod_b . On obtient ainsi une architecture cohérente où tous les flux cohabitent, partageant la même boucle de mise à jour tout en s'enrichissant mutuellement par la formation de clusters réellement multimodaux.

C. Structuration émergente et clusters multimodaux

Le même mécanisme de mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ s'applique de manière uniforme, quel que soit le type des entités \mathcal{E}_i ou \mathcal{E}_j . Les seules différences résident dans la façon de calculer la synergie $S(i,j)$ en fonction des modalités mises en jeu, comme les images, l'audio, le texte ou leur combinaison. Cette approche permet à un Synergistic Connection Network (SCN) de bâtir des regroupements internes tant au sein d'une seule modalité que dans un cadre inter-modal. Par exemple, des entités purement visuelles se rassemblent si leur similarité d'embeddings est jugée élevée, et des triplets comprenant des frames vidéo, du son et un sous-titre textuel émergent si les fonctions S inter-modalités soutiennent cette concordance.

Lorsque l'on compare deux images, la synergie s'appuie sur un score de ressemblance visuelle. Lorsque l'on compare une image et un mot, la synergie se fonde sur un module d'alignement comme CLIP. Lorsque l'on compare un segment audio et un label textuel, un mécanisme audio–texte dédié s'emploie pour produire la valeur de synergie. Le SCN procède ensuite à la mise à jour de toutes les liaisons ω . Les liaisons associées à une forte synergie se renforcent et celles qui manquent de cohérence décroissent. Au bout de plusieurs itérations, le réseau donne naissance à des clusters multimodaux, illustrant l'association spontanée d'éléments issus de flux différents.

La robustesse de la fusion tient au fait que le SCN applique simultanément un renforcement au sein de chaque modalité et un renforcement entre différentes modalités lorsque la similarité est confirmée. Un concept se trouve représenté par plusieurs canaux qui partagent une synergie soutenue entre eux. Un exemple simplifie la compréhension. Une image de chat peut se relier fortement à un mot “cat” et un segment audio correspondant à un miaulement, tandis qu'une image de chien reste faiblement connectée au mot “cat”. Les pondérations évoluent pour consolider ces liens robustes, et l'on obtient un noyau d'entités qui convergent autour de l'idée d'un “chat”, par exemple des images de félin, des sons de miaulement et des mots ou expressions décrivant ce champ sémantique.

Cependant, quand le SCN gère plusieurs types de flux, la taille du problème peut croître rapidement, puisque le nombre total de liaisons $\omega_{i,j}$ est en ordre de n^2 . Il devient donc utile de recourir à des heuristiques comme k-NN pour ne calculer la synergie que sur un voisinage réduit, ou à un algorithme d'inhibition pour maîtriser la densité des liens. Le chapitre 7 discute de ces dispositifs qui contribuent à la scalabilité du processus.

La formule de base $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$ reste identique en multimodal. Les modifications se situent plutôt dans le calcul de $S(i,j)$ pour chaque couple de modalités et dans le recours à des méthodes de filtrage ou d'inhibition. Cette constance de la règle autorise une architecture où toutes les entités, quels que soient leurs types, se soumettent au même mode d'actualisation. Les clusters émergents deviennent alors réellement multimodaux. Les éléments image, texte et audio se retrouvent au sein d'une même région du réseau si leurs synergies croisées l'emportent sur l'effet dissipatif. Les niveaux d'organisation qui en résultent sont bien plus riches qu'en traitement monomodal et rendent possible la révélation de concepts ou d'événements qui impliquent plusieurs flux de données de façon simultanée.

8.7.1.2. Adaptation si $\text{type}(i) \neq \text{type}(j)$: application de la synergie correspondante (vision, texte, etc.)

Lorsque le **DSL** (Deep Synergy Learning) prend en charge plusieurs **modalités** — par exemple la **vision**, le **texte** et l'**audio** — la fonction $S(\mathcal{E}_i, \mathcal{E}_j)$ doit impérativement s'**adapter** au **type** respectif des entités \mathcal{E}_i et \mathcal{E}_j . C'est-à-dire que, dans la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)],$$

le $S(\cdot, \cdot)$ utilisé dépendra de la **modalité** de \mathcal{E}_i et de celle de \mathcal{E}_j . Ainsi, le **même** schéma DSL s'applique, mais on **change** la **formule** de la synergie selon les cas (vision, vision), (vision, texte), (audio, texte),

A. Logique générale : un “switch” sur les types

Lorsqu'on étend le Deep Synergy Learning (DSL) à un ensemble multimodal, il devient essentiel de distinguer les différentes **modalités** en jeu. Le mécanisme de mise à jour des pondérations $\omega_{i,j}$ reste fondamentalement le même, mais la façon de calculer la synergie $S(i, j)$ doit s'adapter à la nature des entités \mathcal{E}_i et \mathcal{E}_j . Pour ce faire, chaque entité \mathcal{E}_i dispose d'un champ $\text{type}(\mathcal{E}_i)$ qui spécifie sa modalité (vision, texte, audio, etc.). Dès lors qu'il s'agit de comparer \mathcal{E}_i et \mathcal{E}_j , on se réfère à $\text{type}(i)$ et $\text{type}(j)$ pour déterminer la fonction de synergie adéquate :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \begin{cases} S_{\text{vision,vision}}(\mathbf{v}_i, \mathbf{v}_j), & \text{si } (\text{type}(i) = \text{vision} \text{ et } \text{type}(j) = \text{vision}), \\ S_{\text{vision,texte}}(\mathbf{v}_i, \mathbf{t}_j), & \text{si } (\text{type}(i) = \text{vision} \text{ et } \text{type}(j) = \text{texte}), \\ S_{\text{audio,texte}}(\mathbf{a}_i, \mathbf{t}_j), & \text{si } (\text{type}(i) = \text{audio} \text{ et } \text{type}(j) = \text{texte}), \\ \dots & \dots \end{cases}$$

Cette organisation s'apparente à un “switch” (ou un “dispatch”) sur les types de modalité. Les entités visuelles, textuelles et audio peuvent ainsi cohabiter au sein du même Synergistic Connection Network (SCN), tout en assurant qu'à chaque comparaison on emploie la bonne mesure de similarité, qu'elle soit intra-modale ou inter-modale.

Un exemple récurrent est l'association entre une image et un texte. Si $\text{type}(\mathcal{E}_i) = \text{vision}$ et $\text{type}(\mathcal{E}_j) = \text{texte}$, alors on invoque $S_{\text{vision,texte}}$. Ce dernier peut être un cosinus entre deux vecteurs appris par un système du type CLIP, qui projette les embeddings de l'image et du texte dans un espace latent commun. On peut également imaginer un module plus sophistiqué, comme un mini-réseau de co-attention traitant le patch visuel face à une séquence de tokens.

De même, si $\text{type}(\mathcal{E}_i) = \text{texte}$ et $\text{type}(\mathcal{E}_j) = \text{texte}$, on applique $S_{\text{txt,txt}}$. Par exemple, si les entités textuelles sont des phrases, on peut en tirer des vecteurs BERT, puis calculer un cosinus. Si ce sont des mots, on peut recourir à des embeddings statiques (Word2Vec, GloVe) et employer une similarité TF-IDF ou une distance cosinus sur ces vecteurs.

Lorsqu'on évalue un segment audio face à un autre segment audio, on utilise $S_{\text{audio,audio}}$. Cela peut être une distance spectrale ou un produit scalaire entre embeddings acoustiques fournis par Wav2Vec ou un modèle d'autoencodeur sur spectrogrammes.

Cette différenciation entre synergies intra-modales (image–image, texte–texte, audio–audio) et synergies inter-modales (image–texte, audio–texte, etc.) constitue un pivot central dans un SCN multimodal. Sans cette adaptativité, on risquerait de comparer arbitrairement un vecteur d'image à un vecteur de texte sans passer par une fonction appropriée, ce qui invaliderait la convergence du DSL.

Après avoir déterminé la synergie $S(i, j)$ adaptée, la mise à jour de la pondération $\omega_{i,j}$ se déroule selon la règle standard :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)].$$

Cette uniformité du cœur DSL, associée à la diversité dans les modules S , permet une fusion multimodale cohérente et flexible. Les liaisons $\omega_{i,j}$ entre entités potentiellement issues de flux variés se renforcent ou se réduisent en fonction du score produit par la fonction correspondante.

B. Mise à jour $\omega_{i,j}(t + 1)$ inchangée

Dans un contexte multimodal, la **même** règle de mise à jour du Deep Synergy Learning (DSL) s'applique sans qu'il soit nécessaire de modifier sa forme fondamentale. Chacune des pondérations $\omega_{i,j}(t)$ entre deux entités \mathcal{E}_i et \mathcal{E}_j suit toujours la relation

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

Cette uniformité est ce qui confère au Synergistic Connection Network (SCN) une grande souplesse et une grande cohérence. D'un point de vue algorithmique, nul besoin d'introduire des heuristiques particulières ou de scinder la dynamique en plusieurs branches spécialisées lorsque l'on souhaite inclure des données audio, visuelles ou textuelles. Le $\omega_{i,j}$ se met à jour selon le même schéma, qu'il s'agisse de comparer une image et une autre image, ou un segment audio et un texte.

La **seule** évolution, dans le cas multimodal, concerne la définition de la **synergie** $S(i,j)$. Lorsque les modalités de \mathcal{E}_i et \mathcal{E}_j concordent (par exemple image–image), on utilise un module de similarité ajusté à ces embeddings visuels. Lorsqu'il s'agit d'un couple audio–texte, on se réfère à un module de correspondance audio–texte qui sait interpréter un vecteur audio et un embedding lexical. La mise à jour de $\omega_{i,j}$ se fonde alors sur la valeur de $S(i,j)$ fournie par ce module. Autrement dit, le réseau *appelle* la fonction $S_{\text{type}(i),\text{type}(j)}$ adaptée, mais la structure de la règle de descente ne varie pas.

Lorsque la synergie demeure suffisante au fil des itérations, le poids $\omega_{i,j}$ s'élève progressivement. Cet effet se produit autant pour des entités de même modalité (deux patches d'image similaires ou deux extraits sonores proches) que pour des entités inter-modales (vision–texte, audio–texte, etc.). Les liens se consolident exactement de la même manière :

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta S(i,j) - \eta \tau \omega_{i,j}(t).$$

On constate que le terme $\eta S(i,j)$ est un incrément de renforcement lorsque la correspondance détectée dans S est positive, tandis que le terme $\eta \tau \omega_{i,j}(t)$ joue le rôle d'amortissement ou de dissipation, poussant à réduire les poids en l'absence de similarité réaffirmée.

L'**avantage** majeur d'une telle invariant dans la dynamique du DSL réside dans la possibilité d'agréger toutes les entités (vision, texte, audio, etc.) dans une **unique** matrice ω . Il n'y a pas besoin de structurer le réseau en modules de descente distincts ou de superviser différemment la pondération en fonction des flux. On définit simplement une fonction de similarité adaptée pour chaque paire de types, ce qui enclenche la **même** descente pour toute liaison $\omega_{i,j}$. La force du SCN se concrétise par un **unique graphe** regroupant tous les noeuds, qu'il s'agisse d'images, de textes ou de pistes audio. Le calcul de $S(i,j)$ varie en fonction de $\text{type}(i)$ et $\text{type}(j)$, mais la mise à jour de ω s'effectue de manière uniforme. Chaque liaison, indépendamment de la modalité d'entrée, augmente en amplitude si la synergie persiste et décroît si la cohérence diminue.

Cette structure unifiée est particulièrement intéressante pour l'**évolutivité**. On peut introduire à n'importe quel moment de nouvelles modalités (par exemple un canal de données sensorielles ou biométriques) en ajoutant un module de calcul de synergie approprié, mais **sans** devoir revoir le mécanisme de descente, qui reste

$$\omega_{i,j}(t + 1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)].$$

L'**auto-organisation** multimodale s'opère alors par ce **processus** local et homogène, permettant de détecter aussi bien des clusters unimodaux (un ensemble de documents textuels très proches) que des clusters inter-modaux (un segment audio, une image et un texte décrivant la même scène). Chaque paire d'entités s'actualise de la même façon, gage de cohérence globale du SCN et de facilité d'implémentation.

C. Résultat : structure unifiée avec clusters mixtes

En appliquant la même règle de mise à jour à tous les couples (i,j) , qu'ils soient issus d'une modalité unique ou de deux flux différents, le Synergistic Connection Network (SCN) produit une **structure** finale où les **clusters** émergent spontanément. Certaines de ces formations correspondent à des regroupements internes dans la même modalité. Un ensemble d'images semblables peut constituer un cluster « purement visuel », tandis que des mots ou des segments de

texte fortement corrélés se rassemblent en un cluster purement textuel. Par ailleurs, grâce à la synergie inter-modale, des **clusters** plus complexes unissant plusieurs canaux apparaissent lorsque leurs pondérations se renforcent suffisamment de manière conjointe.

Un exemple typique illustre cette intégration. Supposez qu'un cluster "chat" prenne forme en rassemblant plusieurs entités. Un certain nombre d'images de chat, diverses expressions textuelles telles que "cat" ou "petit félin domestique" et un enregistrement audio portant sur le miaulement d'un chat. Dans cette configuration, chaque liaison $\omega_{i,j}$ reliant une image et un mot aura été renforcée, car la fonction de similarité image–texte aura perçu une correspondance (par exemple via un embedding de type CLIP). De même, si l'on dispose d'un module audio–image capable de reconnaître la coïncidence entre un miaulement et l'apparition visuelle d'un chat, les pondérations ω reliant l'entité audio au patch d'image auront crû de façon similaire. C'est donc la **même** dynamique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

qui a permis, localement et au fil des itérations, d'accumuler des liaisons élevées vers un **cluster** cohérent, à la fois visuel, textuel et audio. Ce "co-assemblage" reflète la puissance auto-organisée du DSL.

Dans un usage pratique, un utilisateur ou une application peut alors **interroger** le SCN de diverses manières. En partant d'un simple mot ("cat"), on retrouvera aisément les images et les sons associés dans le cluster, car les pondérations ω reliant "cat" à ces entités s'avèrent particulièrement élevées. Inversement, on peut partir d'une image et remonter vers le segment audio ou le label textuel qui lui correspond le mieux dans le réseau. À l'échelle plus large, ce même mécanisme enrichit les possibilités d'**indexation** et d'**annotation** multimédia, ainsi que de **clustering** sémantique intégrant plusieurs canaux. Autrement dit, un seul et même **SCN** exploite l'auto-organisation du DSL pour faire émerger des sous-groupes thématiques ou conceptuels très riches, couvrant vision, texte et audio au sein d'une structure unifiée.

Conclusion

Lorsque $\text{type}(i) \neq \text{type}(j)$, le **DSL** applique la fonction de **synergie** $S_{\text{mod}_i, \text{mod}_j}$ adéquate, qu'il s'agisse de vision–texte, audio–texte ou d'une autre combinaison. La règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$$

reste inchangée. Seule la définition de $S(\mathcal{E}_i, \mathcal{E}_j)$ varie en fonction des modalités, ce qui procure une cohérence dans les liaisons ω , en garantissant que chaque couple d'entités est évalué selon la mesure appropriée, qu'il s'agisse d'un embedding cross-modal ou d'une similarité sémantique. L'intégration des entités se fait de manière fluide, permettant à toutes les modalités, qu'il s'agisse d'images, de textes ou de sons, de cohabiter dans un même **SCN**. La structure ainsi formée favorise l'émergence de **clusters mixtes** où vision, texte et audio se retrouvent interconnectés, ce qui enrichit la représentation et améliore l'interprétabilité du système auto-organisé.

8.7.1.3. Contrôle de la Compétition (Inhibition, Saturation) pour Éviter la Saturation en Liens

Dans un **SCN** (Synergistic Connection Network) conçu pour la multimodalité (cf. §8.7.1.1 et §8.7.1.2), l'abondance de flux et d'entités — images, audio, textes, etc. — peut conduire à une **profusion** de connexions $\omega_{i,j}$ si l'on ne restreint pas la formation ou la croissance des liens moyens. Cette **surabondance** nuit à la fois à la **lisibilité** du réseau et à la **qualité** des clusters formés, tout en alourdisant le **coût** algorithmique. Pour y remédier, on recourt à des **mécanismes de compétition** où l'**inhibition** limite la somme des liaisons sortantes d'une entité et la **saturation** borne la valeur maximale de chaque liaison.

A. Inhibition Dynamique et Limitation de la Somme des Liens Sortants

La règle de mise à jour initiale du Deep Synergy Learning (DSL) définit la croissance d'une liaison $\omega_{i,j}(t)$ en fonction de la synergie $\eta S(i,j)$ et d'un amortissement $\eta \tau \omega_{i,j}(t)$. Cette équation peut être étendue pour inclure un mécanisme d'inhibition qui maintient la somme des liens sortants d'un nœud à un niveau modéré. On ajoute alors un terme supplémentaire au moment de l'incrément de $\omega_{i,j}(t)$. Cette forme modifiée de la règle d'actualisation est décrite par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Le coefficient positif γ introduit une compétition parmi les liaisons sortant d'une même entité \mathcal{E}_i . À chaque itération, la somme $\sum_{k \neq j} \omega_{i,k}(t)$ reflète la quantité de ressources que \mathcal{E}_i mobilise déjà pour d'autres liens. La présence de ce terme dans la mise à jour de $\omega_{i,j}(t)$ freine la croissance simultanée d'un trop grand nombre de liaisons de force comparable. Une entité trop dispersée dans ses connexions se voit ainsi freinée chaque fois qu'elle tente d'augmenter encore un lien sortant.

Cette pénalisation se rapproche d'une contrainte de type L1 sur la somme des pondérations sortantes, car plus la somme des liens $\omega_{i,k}$ augmente, plus la dérivée instantanée de $\omega_{i,j}(t)$ se retrouve négative pour les nouveaux liens qui cherchent à se renforcer. Un nœud doit donc "choisir" les connexions les plus significatives afin de contrer la décroissance liée à l'inhibition. Il s'agit d'une forme de compétition latérale qui rend le réseau plus épars et plus lisible. Les liens qui ne compensent pas assez la pénalisation disparaissent ou se réduisent, tandis que les liaisons vraiment pertinentes survivent en maintenant un score de synergie suffisamment fort pour surpasser la dissipation et l'inhibition.

La conséquence de ce phénomène est l'apparition de clusters plus marqués dans le Synergistic Connection Network, car on évite la situation où un nœud consolide un grand nombre de liaisons moyennes. La dynamique reposant sur une équation localement inchangée, on préserve la nature auto-organisée du DSL. Le coefficient γ se règle selon le degré de parcimonie recherché. Une valeur plus grande force une séparation plus franche en poussant les nœuds à concentrer leurs pondérations sur un petit ensemble de voisins, alors qu'une valeur modérée autorise un certain niveau de dispersion dans les liaisons sortantes. Dans tous les cas, on obtient un contrôle plus fin de la topologie du réseau et de la clarté des communautés qui émergent, sans remettre en cause la base du mécanisme de renforcement et d'amortissement déjà en place.

B. Saturation (Clipping) : Borne Supérieure sur Chaque Liaison

L'inhibition dans un Synergistic Connection Network (SCN) restreint déjà la somme totale des liaisons sortantes d'un nœud, mais il demeure possible qu'une seule liaison $\omega_{i,j}$ prenne une valeur trop élevée si la synergie $S(i,j)$ reste très grande. Afin d'éviter qu'un poids ne devienne démesuré, il est judicieux d'introduire un mécanisme de **clipping** qui impose une borne maximale ω_{\max} . Concrètement, après avoir mis à jour la pondération selon la règle du DSL,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

on applique la saturation

$$\omega_{i,j}(t+1) \leftarrow \min(\omega_{i,j}(t+1), \omega_{\max}).$$

Cette borne supérieure limite la croissance d'un poids, même si le terme $\eta S(i,j)$ encourageait une valeur potentiellement bien plus grande. On empêche ainsi un lien isolé de dominer le réseau à cause d'une synergie exceptionnellement forte. La saturation présente aussi l'avantage de stabiliser plus rapidement la dynamique si une liaison tend à atteindre un équilibre très élevé. Une fois que $\omega_{i,j}$ sature, la mise à jour du DSL ne la fait plus grimper, ce qui allège la pression pour d'autres liaisons et contribue à un ajustement global plus rapide.

Il est en outre possible de rendre la borne ω_{\max} temporellement variable. On peut par exemple choisir une fonction $\omega_{\max}(t) = \omega_{\max}^{(0)} + \beta t$ afin de démarquer un seuil initial bas, imposant une plus grande parcimonie, puis d'autoriser une expansion progressive des valeurs de ω à mesure que le réseau se structure. Cette forme d'adaptation temporelle peut s'avérer utile si la phase initiale de l'apprentissage exige un tri rapide des liens alors que la phase avancée autorise un raffinement plus généreux.

En définitive, le clipping n'affecte pas la forme essentielle de la règle de mise à jour. Il agit après coup, comme un correctif local sur la valeur de $\omega_{i,j}(t+1)$. On maintient ainsi l'**équation** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)],$$

puis on applique $\omega_{i,j}(t+1) = \min(\omega_{i,j}(t+1), \omega_{\max})$. L'objectif est de restreindre chaque liaison au-delà d'une certaine valeur, évitant la surreprésentation d'un lien unique et contribuant à un SCN plus stable et plus lisible.

C. Combinaison des Deux Mécanismes

La version la plus complète de la mise à jour associe l'**inhibition** et la **saturation** en un seul schéma. La règle du DSL devient

$$\omega_{i,j}(t+1) = \{\omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)] - \gamma \sum_{k \neq j} \omega_{i,k}(t)\}_{\text{clip à } \omega_{\max}},$$

où l'on borne la valeur obtenue au maximum ω_{\max} . On cumule ainsi deux principes où la compétition latérale imposée par $\gamma \sum_{k \neq j} \omega_{i,k}(t)$ et la restriction de tout poids $\omega_{i,j}$ à la valeur ω_{\max} . L'inhibition contraint une entité E_i à distribuer ses ressources, tandis que la saturation empêche une liaison unique de grimper sans limite.

D. Éviter la Saturation en Liens pour un SCN Multimodal

Dans un SCN volumineux, où cohabitent de multiples modalités (images, textes, audio, etc.), il arrive qu'une entité se connecte à un très grand nombre d'autres entités avec un niveau moyen de pondération. Un graphe trop dense en résulte, ce qui complique la convergence et la lecture de la structure. L'inhibition force une entité à sélectionner plutôt que de maintenir un large éventail de liaisons non spécialisées. Simultanément, la saturation borne les liaisons dominantes, ce qui évite qu'une poignée de poids monopolise la dynamique.

Ce double contrôle se révèle précieux dans un contexte multimodal. Au lieu d'observer un nœud qui entretient "un peu" de synergie avec des dizaines ou des centaines d'éléments, le SCN fait émerger des grappes plus resserrées où le lien reste fortement justifié par la fonction S . Chacun des deux mécanismes agit à un stade différent où l'inhibition pénalise la somme des pondérations sortantes, alors que la saturation limite la valeur maximale d'une liaison individuelle. Une entité ne peut plus étendre ses connexions de manière indifférenciée et ne peut pas non plus pousser un lien particulier au-delà de ω_{\max} .

Cette approche encourage un réseau plus parcimonieux et plus stable. L'algorithme DSL, identique dans sa structure fondamentale, voit simplement chacun de ses incrément modifié par l'inhibition et finalisé par un éventuel clipping. Le résultat, sur le plan expérimental, est un SCN plus lisible, dont les clusters multimodaux se démarquent clairement, sans que des liens moyens ou disproportionnés ne viennent obscurcir la structure globale. Sur le plan algorithmique, on définit les paramètres γ et ω_{\max} selon les caractéristiques du problème, en tenant compte de la densité visée et du niveau de compétition qu'on souhaite introduire.

Conclusion

Dans un SCN multimodal de grande dimension, la **prolifération** de liaisons $\omega_{i,j}$ représente un risque majeur, tant pour la **lisibilité** du réseau que pour la **charge** de calcul. Les **mécanismes de compétition** (inhibition dynamique) et de **saturation** (clipping) offrent une réponse simple et puissante :

- L'**inhibition** ($-\gamma \sum_{k \neq j} \omega_{i,k}(t)$) limite la somme des liens sortants, poussant chaque entité à un **choix** parcimonieux,
- La **saturation** borne la **valeur** de chaque lien, empêchant une connexion de "monopoliser" la structure.

Cette combinaison garantit une meilleure **sélectivité** dans la formation des clusters multimodaux, favorisant un SCN aux liens moins nombreux mais plus **cohérents**, et accélérant la **convergence** en évitant la stagnation dans un graphe trop dense. C'est donc un mécanisme crucial pour assurer la **qualité** et la **scalabilité** d'un DSL multimodal gérant plusieurs flux (images, sons, textes) simultanément.

8.7.2. Émergence de Clusters Multimodaux

Au sein d'un SCN (Synergistic Connection Network) gérant des **entités multimodales** (images, textes, sons, etc.), la dynamique DSL (Deep Synergy Learning) peut donner naissance à des **clusters** réunissant des contenus de **natures différentes** autour d'un **thème** ou d'un **concept** commun. L'auto-organisation s'applique alors non seulement à l'intérieur de chaque modalité (groupement de documents textuels, similarité entre images, etc.) mais aussi **entre**

modalités, via des mesures de synergie “cross-modales” (texte–image, son–image, etc.). Ce phénomène, particulièrement visible lorsque l’on considère la **fusion** de plusieurs flux (8.7.1), se manifeste ici par la **cohésion** de données variées en un **cluster** multimodal.

8.7.2.1. Cas d'un cluster contenant images, textes et sons liés par un thème commun

Les données multimédias se prêtent particulièrement bien à la logique du **Deep Synergy Learning (DSL)**, dès lors que l'on peut construire une **synergie** multimodale entre des entités hétérogènes (images, textes, sons). Dans ce **cas d'usage**, on envisage la formation d'un **cluster** \mathcal{C} qui réunisse des éléments de plusieurs modalités autour d'un **thème** commun où les **pondérations** entre des images, des extraits sonores et des textes **se renforcent** dès lors que ces documents véhiculent une même sémantique sous-jacente, comme un style musical, un genre cinématographique ou un concept scientifique.

A. Situation de Base

Considérons une **base** de données multimédia composée de trois **ensembles** :

$$\mathcal{I} = \{\mathcal{E}_i^{(\text{img})}\} : \text{un ensemble d'images},$$

$$\mathcal{T} = \{\mathcal{E}_j^{(\text{txt})}\} : \text{un ensemble de textes (articles, résumés, descriptions)},$$

$$\mathcal{A} = \{\mathcal{E}_k^{(\text{aud})}\} : \text{un ensemble d'extraits audio (fragments musicaux, enregistrements de parole, bruitages, etc.)}.$$

Chaque **entité** \mathcal{E}_u appartient donc à l'une de ces **modalités**. Dans l'optique du DSL, on dispose d'une **fonction de synergie** $S(\mathcal{E}_u, \mathcal{E}_v)$, notée aussi $S(u, v)$, qui **mesure** la pertinence ou l'**affinité** entre deux entités, **quelle que** soit leur modalité d'origine. Ainsi, \mathcal{E}_u peut être une image, \mathcal{E}_v un texte, et la **synergie** $S(\mathcal{E}_u, \mathcal{E}_v)$ exprime le **score** de “cohérence” entre l'image et le texte (voir la section 2.2.1.2 pour les grandes familles de *similarités*). L'**objectif** est de laisser le **DSL** faire émerger un **cluster** combinant des $\mathcal{E}_i^{(\text{img})}$, des $\mathcal{E}_j^{(\text{txt})}$ et des $\mathcal{E}_k^{(\text{aud})}$ relatifs à un même **thème**.

B. Fonction de Synergie Cross-Modale

Le Deep Synergy Learning (DSL) s'applique à un Synergistic Connection Network (SCN) dans lequel les entités peuvent appartenir à plusieurs modalités (image, texte, audio, etc.). La seule manière pour le réseau de **fusionner** ces sources hétérogènes est de définir une **mesure** de synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ entre chaque paire, y compris lorsque leurs modalités diffèrent. Pour cela, on procède à la construction de **fonctions partielles** spécialisées selon la nature des entités, puis on les assemble de manière cohérente afin de couvrir l'ensemble des combinaisons (image–image, image–texte, texte–audio, etc.). Sur le plan mathématique, cela débouche sur une “table” de fonctions $S_{\text{mod1,mod2}}$ qui, pour chaque couple (i, j) , fournit un score de similarité ou de correspondance.

Synergie image–texte

Lorsque l'on compare une entité $\mathcal{E}_i^{(\text{img})}$ (image) à une entité $\mathcal{E}_j^{(\text{txt})}$ (texte), on part généralement d'un embedding visuel $\mathbf{v}_i^{(\text{img})} \in \mathbb{R}^{d_{\text{img}}}$ et d'un embedding textuel $\mathbf{v}_j^{(\text{txt})} \in \mathbb{R}^{d_{\text{txt}}}$. On définit ensuite une **fonction** $S_{\text{img-txt}}(\cdot, \cdot)$ permettant d'estimer la compatibilité sémantique entre ces deux vecteurs. Par exemple, on peut recourir à un modèle de type CLIP, où l'on projette image et texte dans un espace latent partagé grâce à deux encodeurs Φ_{img} et Φ_{txt} , puis on compare le résultat via une similarité cosinus ou exponentielle :

$$S(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_j^{(\text{txt})}) = \exp(-\alpha \|\Phi_{\text{img}}(\mathbf{v}_i^{(\text{img})}) - \Phi_{\text{txt}}(\mathbf{v}_j^{(\text{txt})})\|).$$

Cette approche évalue à quel point le concept véhiculé par l'image s'aligne sur le concept véhiculé par le texte, condition nécessaire pour que la liaison $\omega_{i,j}$ se renforce dans le SCN.

Le lien entre un extrait audio $\mathcal{E}_k^{(\text{aud})}$ et un texte $\mathcal{E}_j^{(\text{txt})}$ exige lui aussi un “pont” multimodal. Chaque entité audio est décrite par un vecteur $\mathbf{v}_k^{(\text{aud})} \in \mathbb{R}^{d_{\text{aud}}}$ (par exemple un embedding généré par Wav2Vec), tandis que le texte $\mathbf{v}_j^{(\text{txt})} \in \mathbb{R}^{d_{\text{txt}}}$ peut provenir d’un encodeur BERT. On conçoit alors une fonction

$$S(\mathcal{E}_k^{(\text{aud})}, \mathcal{E}_j^{(\text{txt})}) = \text{sim}\left(\psi_{\text{aud}}(\mathbf{v}_k^{(\text{aud})}), \psi_{\text{txt}}(\mathbf{v}_j^{(\text{txt})})\right),$$

où sim s’appuie sur un cosinus ou un score dérivé d’un alignement sémantique audio–texte. Dans ce cas, ψ_{aud} et ψ_{txt} sont des mappings ou des encodeurs entraînés à rapprocher le contenu sonore et le contenu textuel équivalent.

Bien qu’elle soit moins fréquente que l’appariement image–texte, la comparaison d’une image et d’un signal audio peut également se rencontrer. Des travaux émergent pour aligner un son (bruit d’océan, rugissement d’animal, etc.) et une image représentative (vue de plage, photo d’un lion). La fonction de synergie

$$S(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_k^{(\text{aud})})$$

peut faire usage d’un encodeur visuel et d’un encodeur audio projetant les deux vecteurs dans un espace latent, puis comparer le résultat par un cosinus ou une distance gaussienne. Dans un SCN, cela donne la possibilité de lier, par exemple, un patch d’image de voiture et un segment audio de moteur.

Il ne faut pas négliger la synergie au sein de la même modalité. Deux images $\mathbf{v}_i^{(\text{img})}$ et $\mathbf{v}_j^{(\text{img})}$ se comparent par une similarité cosinus ou une distance RBF pour établir à quel point elles représentent des scènes similaires. Deux morceaux de texte $\mathbf{v}_u^{(\text{txt})}$ et $\mathbf{v}_v^{(\text{txt})}$ font l’objet d’un cosinus sur leurs embeddings BERT ou Word2Vec, tandis que deux segments audio $\mathbf{v}_p^{(\text{aud})}$ et $\mathbf{v}_q^{(\text{aud})}$ se rapprochent si leurs signatures spectrales se recoupent. Ce sont souvent ces synergies intra-modales qui forment la base des clusters unimodaux.

Le SCN devient un graphe dont les nœuds renvoient à des entités variées (images, extraits audio, textes, etc.), et dont les arêtes correspondent aux pondérations $\omega_{i,j}$. À chaque paire $(\mathcal{E}_i, \mathcal{E}_j)$, on associe un score $S(i, j)$ défini par la fonction de similarité ou de correspondance multimodale appropriée. Mathématiquement, on bâtit plusieurs **fonctions** de synergie partielles (image–image, texte–texte, image–texte, audio–texte, etc.) et on les “**colle**” pour en obtenir une version globale cohérente sur l’union des espaces. Les principes de “collage” discutés en section 2.2.1.1 (théorème de consistance) assurent la régularité de S lorsque plusieurs blocs d’entités se rencontrent.

Au moment d’activer la dynamique du DSL, la mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i, j) - \tau \omega_{i,j}(t)]$$

se base sur cette fonction globale S . Les paires jugées cohérentes à travers leur modalité (ou entre différentes modalités) voient leurs liaisons renforcées ; celles qui ne révèlent pas de correspondance s’affaiblissent jusqu’à s’annuler, favorisant l’émergence naturelle de clusters mixtes ou purement unimodaux selon les scénarios.

C. Émergence d’un Cluster Multimodal

Une fois la fonction de **synergie** $S(u, v)$ définie pour chaque couple (u, v) , y compris dans un cadre multimodal, le Deep Synergy Learning (DSL) met en branle la **règle** de mise à jour des pondérations $\omega_{u,v}(t)$. Cette règle, rappelée en **section 2.2.2**, se présente typiquement sous la forme

$$\omega_{u,v}(t+1) = \omega_{u,v}(t) + \eta[S(u, v) - \tau \omega_{u,v}(t)].$$

L’algorithme effectue cette mise à jour à chaque itération ou “tick” du réseau, de sorte que les liaisons $\omega_{u,v}$ se renforcent si la synergie $S(u, v)$ est jugée importante, tout en subissant une décroissance $\tau \omega_{u,v}(t)$ si cette synergie n’est pas régulièrement réaffirmée. Lorsque $S(u, v)$ reste élevé sur plusieurs itérations, $\omega_{u,v}$ se fixe approximativement vers $S(u, v)/\tau$. À l’inverse, deux entités n’ayant pas de correspondance prononcée voient leur liaison décroître vers zéro.

Le **processus** auto-organisé du DSL, via cette règle de mise à jour, conduit naturellement à la **constitution** de clusters lorsque différents canaux (image, texte, audio, etc.) décrivent un même **thème** sous des angles complémentaires. Dans un exemple, un sujet “Jazz” peut rassembler :

Des **entités visuelles** : Plusieurs images ou photos de musiciens, d’instruments (saxophone, trompette), ou de pochettes d’album associées au Jazz.

Des **entités textuelles** : Fragments de texte ou mots clés tels que “Miles Davis”, “Duke Ellington”, “improvisation”, “swing”, “blue note”, etc.

Des **entités audio** : Extraits sonores présentant la signature harmonique jazz (rythmique swing, modes, instruments spécifiques) reconnus par un classifieur audio ou un embedding Wav2Vec.

Chacune de ces entités, prise individuellement, relève de sa propre modalité (vision, texte, audio). Mais le SCN peut faire correspondre une image spécifique (un musicien en concert) à un extrait textuel (mentionnant l’improvisation jazz) et à un segment audio (solo de saxophone). Le fait que la **synergie** conserve des scores élevés (image–texte, texte–audio, image–audio) sert de moteur à la consolidation de liaisons $\omega_{i,a}$, $\omega_{a,p}$, $\omega_{i,p}$ dans le DSL.

À mesure que les pondérations se stabilisent, on voit ainsi **émerger** un **cluster cohérent**, noté $\mathcal{C}_{\text{jazz}}$, qui regroupe ces images, textes et extraits audio autour de la même thématique. On parle alors de **cluster multimodal**, car il ne se limite pas à une catégorie de données, il embrasse des éléments différents unis par une idée commune.

Exemple concret

On suppose que certains nœuds $\{\mathcal{E}_1^{(\text{img})}, \dots, \mathcal{E}_m^{(\text{img})}\}$ correspondent à des photographies de concerts de Jazz (musiciens, instruments, ambiance de club). D’autres nœuds $\{\mathcal{E}_a^{(\text{txt})}, \dots\}$ sont des passages de texte citant “Duke Ellington”, “Miles Davis”, “improvisation”, “swing”, et ainsi de suite. Enfin, d’autres nœuds $\{\mathcal{E}_p^{(\text{aud})}, \dots\}$ renvoient à des enregistrements où un modèle audio détecte un tempo, une harmonie ou un style caractéristiques du Jazz.

La fonction de synergie inter-modale identifie qu’une image exhibant un saxophone $\mathbf{v}_i^{(\text{img})}$ se rapproche d’un mot “saxophone” $\mathbf{v}_a^{(\text{txt})}$, ou qu’un extrait sonore $\mathbf{v}_p^{(\text{aud})}$ présente une corrélation forte avec ce concept musical. À chaque itération DSL, les liaisons $\omega_{i,a}$, $\omega_{a,p}$, $\omega_{i,p}$ se **renforcent** car la valeur $S(i, a)$, $S(a, p)$, $S(i, p)$ demeure élevée. Les entités finissent par “coaguler” en un **cluster** $\mathcal{C}_{\text{jazz}}$ unissant images, textes et audio référant à la thématique Jazz.

Conclusion

À l’issue de la convergence, les **clusters** reflètent des zones du réseau où la synergie, intramodale ou inter-modale, se maintient à un niveau suffisant pour entretenir les liens $\omega_{i,j}$. Les concepts multimodaux (ici “Jazz”) s’expriment ainsi par l'**auto-organisation**, plutôt que par un pipeline supervisé imposant la réunion d’images, de textes et de sons. Un utilisateur peut alors exploiter ce cluster pour naviguer entre images associées au Jazz, textes décrivant l’histoire de ce style, ou morceaux audio illustrant son ambiance. Cette approche illustre la **flexibilité** du DSL puisque la constitution de clusters se fait simplement en laissant la **même** règle de descente s’appliquer, couplée à une synergie S adaptée pour chaque paires d’entités (image–texte, audio–image, texte–audio, etc.).

D. Vision Mathématique : Liens dans un Graphe Hétérogène

On peut formaliser l’ensemble des entités images–textes–audio dans un **graphe** \mathcal{G} (ou hypergraphe), dont les **nœuds** u représentent les entités \mathcal{E}_u . À chaque couple (u, v) est associée une **pondération** $\omega_{u,v}(t)$, qui évolue selon la **dynamique** DSL. Avec un **terme de parsimonie** ω_{\min} (section 2.2.3), on peut élaguer les liens trop faibles. Au terme d’un certain nombre d’itérations, un **sous-graphe** cohésif apparaît là où la **synergie** est notable, formant ainsi un **cluster** transcendant la diversité des modalités.

Si l’on se limite à la **règle additive**, on a :

$$\omega_{u,v}(t+1) = (1 - \eta \tau) \omega_{u,v}(t) + \eta S(u, v),$$

et la convergence se produit quand $\omega_{u,v}(t+1) \approx \omega_{u,v}(t) \approx \frac{S(u,v)}{\tau}$.

E. Rôle dans la Découverte de Thèmes

Le Deep Synergy Learning (DSL), lorsqu'il est appliqué à un réseau d'entités multimodales (images, texte, audio, voire plus), autorise l'**auto-organisation** de ces données en **clusters** sans supervision préalable. Les pondérations $\omega_{i,j}$ se mettent à jour selon la règle de base, et la **synergie** $S(i,j)$ détermine dans quelle mesure deux entités de différentes modalités se révèlent pertinentes l'une pour l'autre. Cette logique permet la **découverte** de thèmes transversaux qui associent librement des informations diverses, sans qu'on impose de catégories nommées comme "Jazz", "Classique" ou "Pop".

Lorsque le SCN opère en mode non supervisé, aucune étiquette n'est imposée pour guider la constitution des clusters. Les images, le texte et l'audio cohabitent dans le même graphe, et chaque pondération $\omega_{i,j}(t)$ grandit ou décroît au fil des itérations en fonction du score de synergie $S(i,j)$. Peu à peu, le réseau forme des **groupements** stables, chacun reflétant un **thème** ou un **concept** partagé. Par exemple, un groupe peut réunir des images de saxophones, des extraits sonores de jazz et des références textuelles à "Miles Davis" ou "improvisation", même si le système n'a jamais reçu explicitement le label "Jazz". L'émergence de ce cluster s'opère simplement parce que la synergie image–texte–audio demeure élevée entre les entités concernées. Cette capacité à s'**auto-organiser** sans label explicite est ce qui distingue le DSL d'architectures strictement supervisées.

Le **principe** multimodal fournit une **complémentarité** entre les flux. Quand une entité visuelle (une photo de saxophone) affiche déjà une cohérence avec un extrait textuel (mention d'un instrument "saxophone"), et que l'audio lié présente un spectre caractéristique du jazz, l'association se trouve renforcée simultanément sur trois axes (image–texte, texte–audio, image–audio). Les indices se **soutiennent** mutuellement puisqu'une image isolée de saxophone pourrait prêter à confusion si elle n'est pas identifiée, mais le texte "saxophone" et l'extrait audio "solo de sax" confirment la cohérence. Le résultat est un **cluster** plus fiable et précis que s'il n'y avait qu'une seule modalité pour juger de la similarité.

Cette **philosophie** multimodale s'étend aisément à de nouvelles sources de données. Il suffit de définir une fonction de synergie S adéquate pour la paire (nouvelle modalité, modalité existante). Par exemple, pour de la **vidéo**, on conçoit un encodeur vidéo (CNN 3D, transformeur spatio-temporel) permettant d'extraire un embedding, puis on définit un score de correspondance vidéo–image ou vidéo–texte, etc. De même, l'ajout de **capteurs** IoT (température, pression, localisation GPS) se gère en construisant un module de similarité entre ces mesures et les éventuels flux sémantiques. Au niveau énergétique, la somme

$$\sum_{u,v} \omega_{u,v} S(u, v)$$

s'enrichit de contributions supplémentaires, reflétant la synergie que de nouveaux canaux peuvent apporter. Le SCN élargit alors la possibilité de faire **émerger** des **communautés** ou des thèmes encore plus vastes, associant différents types d'informations et conservant la logique habituelle de renforcement et de dissipation ($\tau \omega_{u,v}$).

La conséquence finale est un **réseau** dont la topologie s'ajuste de manière distribuée pour faire ressortir des **clusters** multimodaux. Ils peuvent être observés comme des "macro-nœuds" qui regroupent un ensemble d'éléments disparates (visuels, textuels, audio, etc.) tous liés par des pondérations ω suffisamment fortes. Cette dynamique exploite pleinement la **complémentarité** des flux pour découvrir des **thèmes** qui n'étaient pas explicitement nommés au départ, rendant le DSL particulièrement adapté aux tâches de fouille de données et d'exploration non supervisée dans des environnements riches et diversifiés.

F. Conclusion

Un **cluster** multimodal naît lorsque, dans un **réseau** hétérogène (images–textes–sons), les liens $\omega_{u,v}$ ont été **renforcés** par une **synergie** suffisamment élevée pour **rassembler** des entités autour d'un même **thème**. Du point de vue mathématique, tout se joue dans :

620. La **fonction** $S(\cdot, \cdot)$ qui, par alignement cross-modal, **assigne** un score positif à des paires hétérogènes cohérentes,

621. La **dynamique** DSL (sections 2.2.2 et 2.2.3), qui **met à jour** $\omega_{u,v}$ et **stabilise** les liens forts en un **sous-graphe** homogène,

622. La **parsimonie** (éventuelle) qui **supprime** les connexions trop faibles, mettant en relief des micro-communautés nettement connectées.

Ce **mécanisme de fusion** d'entités multimodales **facilite la découverte** de **thèmes** sans avoir besoin de labels explicites. Les **clusters** formés servent alors de **structures d'indexation**, de **recherche**, ou de **classification** conceptuelle, permettant de naviguer dans des bases multimédia complexes en identifiant automatiquement les **groupes** d'items partageant une **unité** sémantique (ex. "Jazz", "Bande Dessinée", "Recettes de cuisine italienne", etc.). Le **DSL** devient ainsi un outil pertinent pour l'**organisation** et la **compréhension** non supervisées de grandes collections de données multi-modales.

8.7.2.2. Outils pour repérer ces groupes de nœuds hétérogènes

Dans un **SCN** (Synergistic Connection Network) où coexistent plusieurs **modalités** (images, textes, sons, etc.), il arrive fréquemment qu'un **cluster** rassemble des nœuds aux caractéristiques très **dissemblables** tout en entretenant entre eux des **liaisons** $\omega_{i,j}$ significatives. Ces *groupes hétérogènes* peuvent révéler des **patterns** transversaux (thèmes ou processus communs) ou, au contraire, signaler des **anomalies** (brassage inhabituel de modalités). La présente section dresse un **panorama** des méthodes et **indicateurs** permettant d'**identifier** ces **regroupements** multimodaux dans le cadre d'un **DSL**.

A. Analyse des Entropies et Indicateurs de Diversité

Une **approche** classique pour quantifier la **multimodalité** d'un sous-groupe $\mathcal{C} \subseteq \{\text{nœuds}\}$ consiste à mesurer l'**entropie** de sa **composition**. En supposant que \mathcal{C} renferme des entités issues de plusieurs **modalités** $m \in \{\text{img, txt, aud, ...}\}$, on note

$$p_m(\mathcal{C}) = \frac{|\{i \in \mathcal{C} \mid \text{mod}(i) = m\}|}{|\mathcal{C}|} \quad (\text{pour chaque modalité } m).$$

L'**entropie** de Shannon se définit alors par

$$H(\mathcal{C}) = - \sum_m p_m(\mathcal{C}) \ln(p_m(\mathcal{C})).$$

Plus $H(\mathcal{C})$ est élevée, plus la **composition** en modalités est **diversifiée**, signifiant que \mathcal{C} "mélange" fortement des nœuds de natures variées. À l'inverse, un **cluster** quasi-monolithique (ex. presque uniquement des images) aura un $H(\mathcal{C})$ faible. On peut alors décider de **repérer** les clusters où $H(\mathcal{C}) > H_{\min}$ pour ne retenir que les sous-groupes vraiment **hétérogènes**.

Au-delà de l'entropie, divers **indices de diversité** (Simpson, Gini, Hill numbers, etc.) sont employés en écologie ou en classification pour rendre compte de la **répartition** des classes. Par exemple, l'**indice de Simpson** sur la distribution $\{p_m\}$ se définit par

$$D_{\text{Simpson}}(\mathcal{C}) = 1 - \sum_m (p_m(\mathcal{C}))^2.$$

Il atteint son maximum lorsque les p_m sont **uniformes**. Ainsi, pour la **détection** de clusters "vraiment multimodaux", on peut filtrer les valeurs supérieures à un seuil D_{\min} . D'un point de vue **mathématique**, ces indices donnent un **chiffre synthétique** de l'**hétérogénéité** interne, ce qui facilite la **comparaison** et le **rang** des clusters trouvés par un algorithme DSL ou de partition.

B. Cartographie du Degré ou des Forces de Liaison Inter-modales

Dans le cadre d'un **SCN multimodal**, on peut examiner les **liaisons** $\omega_{i,j}$ en **stratifiant** par **paires** de modalités (m_1, m_2). Soit $\mathbf{W}^{(m_1, m_2)}$ la **sous-matrice** contenant les poids $\omega_{i,j}$ pour $\text{mod}(i) = m_1$ et $\text{mod}(j) = m_2$. Les **densités** de ces blocs peuvent révéler l'**intensité** des connexions inter-modales :

$$d(m_1, m_2) = \frac{1}{|m_1| \cdot |m_2|} \sum_{\substack{i \in m_1 \\ j \in m_2}} \omega_{i,j}.$$

Si $d(m_1, m_2)$ est **beaucoup** plus élevé que d'autres combinaisons, on suspecte une **affinité** exceptionnelle entre ces deux modalités. Une étude plus détaillée de la structure (ex. un bloc dans la matrice $\mathbf{W}^{(m_1, m_2)}$ dense) peut alors faire émerger un **cluster** mixte associant m_1 et m_2 .

Pour chaque nœud i , on peut décomposer son **degré** (ou sa **force**) en **catégories** selon la modalité de ses voisins :

$$\mathbf{deg}_i = \left(\sum_{j : \text{mod}(j)=\text{img}} \omega_{i,j}, \sum_{j : \text{mod}(j)=\text{txt}} \omega_{i,j}, \sum_{j : \text{mod}(j)=\text{aud}} \omega_{i,j}, \dots \right).$$

Ainsi, si un nœud i présente un vecteur \mathbf{deg}_i dont **tous** les composants sont significatifs (forte connexion image, forte connexion texte, etc.), on soupçonne un "**hub** multimodal". Un regroupement de tels nœuds "hubs" (avec liens transverses) peut être un **sous-ensemble** hétérogène à forte interconnexion inter-modale.

C. Méthodes de Détection de Communautés Mixtes

Lorsque le Deep Synergy Learning (DSL) a produit un réseau dont les liaisons $\omega_{i,j}$ unissent des entités potentiellement issues de plusieurs modalités (images, textes, audio, etc.), il devient utile de recourir à des **méthodes** de partitionnement afin d'identifier des **communautés** ou sous-réseaux thématiques. De nombreux algorithmes de détection de communautés dans les graphes pondérés ont été développés, comme Girvan–Newman, Louvain, Infomap, ou le clustering spectral. La dimension multimodale amène quelques ajustements, car on peut souhaiter repérer des groupes hétérogènes englobant plusieurs flux de données.

Un premier choix consiste à appliquer la **modularité** pondérée telle que définie par Louvain ou Infomap, mais en l'adaptant aux spécificités des flux. On peut insérer un **facteur** qui valorise ou pénalise les liaisons inter-modales, par exemple en majorant les poids $\omega_{i,j}$ lorsque les entités \mathcal{E}_i et \mathcal{E}_j appartiennent à des modalités différentes. Cette astuce oriente l'algorithme vers une communauté plus variée. Inversement, si l'on cherche à privilégier des regroupements unimodaux, on diminue ou on neutralise les poids inter-modaux. De manière plus technique, il est possible de définir une fonction de **modularité** Q qui inclut un terme de "compatibilité modale" $\chi(\text{mod}(i), \text{mod}(j))$. Ainsi, une liaison entre un nœud visuel et un nœud textuel reçoit un coefficient plus élevé lorsqu'on vise à constituer des groupes multimodaux.

Une deuxième approche consiste à employer un algorithme de détection de communautés **standard** sur la matrice \mathbf{W} issue du SCN, puis à **filtrer** ex post les clusters identifiés pour n'en garder que ceux qui révèlent une entropie modale ou une diversité modale satisfaisante. Ce filtrage se base sur la distribution des types de nœuds dans chaque communauté. Un cluster composé uniquement d'images se voit écarté si l'on recherche absolument un mélange, alors qu'un cluster contenant plusieurs images, quelques extraits audio et des segments textuels est considéré comme "mixte" ou "hétérogène".

Dans certains scénarios, l'on souhaite extraire un **sous-graphe** restreint mais riche en hétérogénéité, plutôt qu'une partition globale. Cette recherche de motifs ou de hubs multimodaux fait appel à d'autres techniques. On peut définir une **fonction** $\mathcal{H}(\mathcal{C})$ mesurant la force interne du sous-ensemble \mathcal{C} (somme des pondérations $\omega_{i,j}$ pour $i, j \in \mathcal{C}$) tout en tenant compte de la répartition modale. Ce critère \mathcal{H} peut être maximisé à travers l'examen de sous-ensembles, ou encore exploré par un algorithme de recherche de motifs. Dans un cadre spectral, la factorisation de la matrice Laplacienne $\mathcal{L} = \mathbf{D} - \mathbf{W}$ révèle des ensembles de nœuds fortement connectés. On peut examiner la composition modale de ces ensembles et sélectionner ceux qui arborent une mixité marquée. D'autres approches se fondent sur le concept de "multi-view clustering", où chaque modalité est considérée comme une vue, et l'on cherche des groupes

où la cohérence se retrouve sur plusieurs vues, ce qui correspond à la logique DSL quand les liaisons inter-modales se confortent mutuellement.

L'objectif commun à toutes ces techniques est de repérer des **communautés** qui reflètent un **thème** riche, associant différents canaux dans un même bloc d'entités. C'est grâce à la synergie inter-modale calculée dans le DSL que de tels groupes peuvent émerger, car la matrice \mathbf{W} renferme déjà l'information des correspondances images, textes, audio. Les algorithmes standard de détection de communautés se retrouvent simplement confrontés à un graphe où les liens inter-modaux sont intégrés. Selon le choix de valorisation ou de filtrage, l'on peut forcer un regroupement plus varié ou au contraire extraire des motifs purement unimodaux. La souplesse de ce processus permet un large éventail d'analyses et d'usages, depuis la recherche de motifs spécifiques dans un gros réseau multimédia, jusqu'à la partition globale en clusters hétérogènes selon la modularité pondérée.

D. Stratégies Stochastiques ou DSL-based pour la Détection

Le **Deep Synergy Learning** (voir chap. 2.2.2) peut lui-même être **modifié** pour **encourager** la connexion inter-modale. Par exemple, dans le **score** de synergie $S(i,j)$, on ajoute un **bonus** δ lorsque $\text{mod}(i) \neq \text{mod}(j)$. Alors la **mise à jour** des poids

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) + \delta \cdot \mathbf{1}_{\text{mod}(i) \neq \text{mod}(j)} - \tau \omega_{i,j}(t)]$$

tend à **renforcer** systématiquement les **liens** multimodaux, ce qui fait **émerger** plus facilement des **clusters** hétérogènes. D'autres variantes consistent à appliquer des **inhibitions** à l'intérieur d'une même modalité pour forcer les entités à se tourner vers d'autres types de nœuds.

En combinant le DSL avec des **méthodes stochastiques** (recuit, random walk, bruit sur $\omega_{i,j}$), on peut **explorer** des partitions plus audacieuses. Le recuit simulé, par exemple, perturbe ponctuellement les pondérations pour *sortir* de minima locaux dominés par des regroupements monomodaux. Cela peut **ouvrir** la voie à des **structures** plus complexes où plusieurs modalités cohabitent au sein d'un même **cluster**.

Conclusion

L'**identification** de **groupes hétérogènes** dans un **SCN multimodal** exige de **combiner** :

- Des **indicateurs** de la **diversité** modale (entropie, indices de Simpson ou de Gini),
- Une **cartographie** fine des **liaisons** entre modalités (matrices $\mathbf{W}^{(m1,m2)}$, degré inter-modal),
- Des **algorithmes** de **communauté** revisités pour valoriser l'hétérogénéité (modularité "cross-modale", score \mathcal{H}),
- Éventuellement, des **ajustements** DSL (bonus de synergie inter-modale, recuit simulé) favorisant l'**émergence** de clusters mixtes.

Cette **boîte à outils** fournit un **cadre** mathématique et algorithmique pour **repérer**, au sein d'un réseau, ces **groupements** où se rencontrent des entités de modalités distinctes. Dans un **DSL** ou tout autre environnement auto-organisé, ces **groupes** hétérogènes revêtent souvent un **intérêt** opérationnel (découverte de thèmes complexes, interfaces multimodales) ou suscitent une **analyse** plus approfondie (anomalies ou phénomènes transversaux).

8.7.2.3. Potentiel d'Utilisation : Indexation Automatique, Annotation, Groupement Conceptuel

Dans un environnement **multimodal**, où différents types d'objets (textes, images, audio, métadonnées, etc.) se trouvent représentés dans le **Synergistic Connection Network** (SCN) d'un **Deep Synergy Learning** (DSL), la possibilité de réaliser une **indexation** automatique, une **annotation** de contenu et un **groupement** conceptuel dynamique revêt une importance cruciale. Le **DSL**, en créant et en maintenant des **pondérations** $\omega_{i,j}$ dictées par la **synergie** $S(i,j)$, offre une **approche adaptative** qui s'étend bien au-delà des simples **classifications** ou **recherches** par mots-clés traditionnelles. L'objectif devient de **faire émerger** des liens et des regroupements entre entités, de **gérer** l'assignation d'étiquettes ou de mots-clés en continu, et de **former** des clusters conceptuels plus larges reflétant l'évolution des données.

A. Indexation Automatique

L'**indexation** vise à lier chaque entité (document texte, image, extrait audio, etc.) à un **ensemble** de mots-clés ou de **concepts** censés en **résumer** la teneur. Dans un **SCN**, l'entité “mot” (ou “concept”) peut coexister au même titre qu'un **document**, le DSL **renforce** leurs liaisons ω dès que la **synergie** s'avère élevée, révélant ainsi une **affinité** ou une co-occurrence fréquente. L'idée s'incarne dans une **dynamique** locale où plus un *mot* m est pertinent pour un *document* d , plus le score $S(m, d)$ est grand, et plus $\omega_{m,d}$ s'accroît à chaque itération. Ce **mécanisme** confère un cadre itératif et “vivant” à l'indexation, permettant de réviser les **associations** mots-documents dès lors que des **informations** nouvelles font évoluer S .

On considère un **ensemble** de mots $\{\mathcal{E}_m\}$ et un ensemble de documents $\{\mathcal{E}_d\}$. L'évaluation de $S(\mathcal{E}_m, \mathcal{E}_d)$ peut reposer sur :

$$S(m, d) = \text{sim}(\mathbf{v}_m, \mathbf{v}_d),$$

où \mathbf{v}_m et \mathbf{v}_d sont des **représentations** (embeddings) vectorielles ; ou encore sur un **score** TF-IDF, ou une **co-occurrence** dans un grand corpus.

À chaque pas t , la **pondération** $\omega_{m,d}(t)$ suit :

$$\omega_{m,d}(t+1) = \omega_{m,d}(t) + \eta [S(m, d) - \tau \omega_{m,d}(t)].$$

Si la **similarité** $S(m, d)$ se maintient élevée, $\omega_{m,d}$ tend vers un **point** d'équilibre $\omega_{m,d}^* \approx S(m, d)/\tau$. Une fois la **pondération** supérieure à un **seuil** θ , on peut considérer que “le mot \mathcal{E}_m indexe le document \mathcal{E}_d ”. Cette **auto-organisation** se substitue ou complète les approches statiques de “bag of words” en offrant un **système adaptatif** où l'arrivée de **nouveaux** mots ou documents réinjecte de la plasticité, permettant de **réajuster** $\omega_{m,d}$ de façon incrémentale.

L'un des principaux avantages d'un **SCN** appliqué à l'indexation réside dans sa capacité d'**adaptation** continue. Contrairement aux systèmes d'indexation traditionnels, où les relations entre mots et documents sont préétablies et figées, un **SCN** ajuste dynamiquement ses **liens** à mesure que de nouveaux documents et termes apparaissent. Cette plasticité permet au réseau de rester pertinent sans nécessiter de mise à jour manuelle constante.

L'**indexation** ne se limite pas à une simple correspondance **mot-document**, mais s'enrichit d'une **structuration dynamique** où un même mot peut référencer plusieurs documents et inversement. La force des liens entre un mot et un document est quantifiée par un **score** $\omega_{m,d}$, qui reflète leur pertinence mutuelle. Plutôt que d'imposer une relation binaire, le **SCN** capture des **niveaux de pertinence**, améliorant ainsi la précision des recherches et des associations.

Enfin, pour éviter une prolifération excessive des liaisons et assurer une **cohérence** structurelle, un **mécanisme de parsimonie** peut être intégré. Ce mécanisme filtre les connexions faibles en supprimant les liens où $\omega_{m,d} < \omega_{\min}$, ne conservant ainsi que les **associations les plus robustes**. Cette approche garantit une indexation plus stable et efficace, en ne retenant que les relations les plus significatives entre mots et documents.

B. Annotation et Étiquetage Dynamique

Au-delà de la simple association “document-mot-clé”, on peut introduire dans le **SCN** des **nœuds** représentant des “catégories” ou “annotations” plus abstraites (thèmes, domaines, labels). Les documents, images ou autres entités qui entretiennent une **synergie** élevée avec un label \mathcal{L} verront la pondération $\omega_{\mathcal{L},d}$ grimper, signant l'**appartenance** (ou la pertinence) de l'entité pour cette **catégorie**.

Dans un cadre strictement **non supervisé**, le DSL peut faire émerger des clusters hétérogènes où l'on peut décider ex post de “**nommer**” un cluster stable en lui attribuant un **label** explicite. Dans un cadre **semi-supervisé**, on intègre dès le départ des **nœuds** “label” dans le SCN, et on laisse la **dynamique** DSL (synergie + plasticité) **relier** ces labels aux entités appropriées.

Mathématiquement, il suffit d'introduire un **ensemble** d'entités $\{\mathcal{E}_{\mathcal{L}}\}$ où \mathcal{L} désigne une catégorie ou un label potentiel, et de définir :

$$S(\mathcal{E}_L, \mathcal{E}_d) = \text{sim}(\mathbf{v}_L, \mathbf{v}_d),$$

où \mathbf{v}_L capture la **représentation** du label (ex. un vecteur sémantique). On applique ensuite la mise à jour usuelle :

$$\omega_{L,d}(t+1) = \omega_{L,d}(t) + \eta [S(L, d) - \tau \omega_{L,d}(t)].$$

À la **convergence**, si $\omega_{L,d}$ dépasse θ , on en déduit que l'entité \mathcal{E}_d est efficacement **annotée** par la catégorie L . L'éventuel **changement** des vecteurs \mathbf{v}_L ou l'introduction d'une **nouvelle** catégorie dans le SCN fera rejoindre ces dynamiques d'adaptation, sans qu'il soit nécessaire de relancer une classification globale.

Contrairement à des systèmes d'**étiquetage** rigides reposant sur une hiérarchie fermée de labels ou un regroupement one-shot, un **SCN** sous **DSL** possède plusieurs avantages. Il peut **intégrer** de nouvelles catégories ou sous-catégories de manière dynamique, sans nécessiter de refonte complète. Il conserve les associations passées tout en les **réactualisant** en cas de conflit ou d'évolution des données. Enfin, il bénéficie d'une **unification** avec l'indexation, où un label ou un mot-clé est simplement une entité \mathcal{E} supplémentaire, régie par les mêmes principes de synergie.

C. Groupement Conceptuel

Un **SCN** ne se limite pas à la relation “entité–mot” ou “entité–catégorie” puisque l'ensemble des nœuds peut inclure des *documents*, des *mots*, des *étiquettes*, des *métadonnées*, voire des “profils d'utilisateur” ou des “auteurs”. À travers la **dynamique DSL**, de **macro-clusters** peuvent se constituer, rassemblant simultanément :

- Des mots-clés.
- Des articles/documents.
- Des catégories ou labels.
- Des entités annexes (des auteurs, des lieux, etc.).

Un tel **macro-cluster** se présente comme un “**groupe conceptuel**” où divers types de nœuds cohabitent autour d'une **cohérence** sémantique ou thématique. Au plan mathématique, on peut détecter ces clusters via la **densité** des liens ω , avec l'aide de techniques mentionnées en (8.7.2.2) pour **repérer** les sous-graphes multimodaux.

Exemple Illustratif

Imaginons un **réseau** où :

- Des *mots* comme “semi-conducteur”, “processeur”, “GPU” possèdent de fortes similarités textuelles ou co-occurrences,
- Des *documents* techniques mentionnent les mêmes notions,
- Un *label* “Informatique Matérielle” est lui-même corrélé à ces documents,
- Un *auteur* A revient régulièrement dans les références de ces articles.

Si la **synergie** se maintient élevée entre “semi-conducteur”, “processeur”, “informatique matérielle”, “auteur A”, “article X, Y, Z”, on aboutit à un **cluster** stable, représentant un **groupe conceptuel** autour du hardware computing. Les pondérations ω marquent chacun des liens (mot ↔ document, document ↔ auteur, label ↔ mot, etc.), donnant une **vision** globale de la **cohérence**. Cette vision s'extrait mathématiquement en repérant un sous-graphe très connecté, dont l'**entropie** modale (ou la **richesse** en entités distinctes) peut être élevée.

Le **DSL** autorise une **adaptation en continu**. Si des *nouveaux mots* (ex. “RISC-V”) apparaissent et entretiennent une forte similarité sémantique avec “processeur”, “semi-conducteur”, les **liens** $\omega_{(\text{RISC-V}, \text{others})}$ se renforcent progressivement et peuvent faire **bifurquer** le cluster existant vers une définition plus large ou entraîner la **naissance** d'un sous-cluster (voir chap. 6.5 sur les bifurcations et agrégations).

D. Conclusion

Les mécanismes du **Deep Synergy Learning**, en développant un **SCN** où sont représentés *documents, mots, catégories, auteurs* ou toute entité pertinente, constituent un **socle** particulièrement flexible pour :

- **Indexation automatique.** Les liaisons $\omega_{m,d}$ traduisent l'appariement mot–document au fil de la synergie, fournissant ainsi une **liste d'index adaptative**.
- **Annotation et étiquetage dynamique.** Les catégories deviennent des nœuds du réseau, dont les liens avec des entités se renforcent si la **similarité** demeure forte ; cela facilite une **classification** continue, même en présence de nouvelles catégories.
- **Groupement conceptuel.** Les **macro-clusters** émergents dans un SCN comportant plusieurs **types** d'entités (mots, documents, labels, auteurs, etc.) se lisent comme des **concepts** ou **thèmes** auto-organisés, reflétant une cohérence à plusieurs dimensions.

En pratique, la **mise à jour**

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

accomplit tout le “travail” de **plasticité** nécessaire pour faire émerger ces **structures** conceptuelles, et ce de façon **itérative**. Du point de vue **opérationnel**, on peut décider d'un **seuil** ω_{\min} pour ne retenir que les **liens** pertinents, ou, à l'inverse, autoriser des **clicques** plus denses qui matérialisent l'existence de **thèmes** complexes. En somme, l'**intégration** dans un **SCN** confère à l'indexation, à l'annotation et au groupement conceptuel un **caractère** organique et évolutif, particulièrement adapté à des corpus multimodaux ou en constante mutation.

8.7.3. Oscillations ou Confusions possibles

Dans l'architecture **multimodale** (texte, image, audio...) du DSL, il est courant qu'une même entité (par exemple, un **mot** en texte) se trouve liée à plusieurs **objets** (images, segments audio, etc.). Cette surabondance de connexions peut engendrer des **oscillations** ou des **confusions**, où la pondération $\omega_{i,j}$ varie de manière instable entre différents partenaires. Dans ce paragraphe (8.7.3), nous analysons comment de tels phénomènes de flou ou d'ambiguïté peuvent survenir et comment on peut y **remédier**.

8.7.3.1. Si un mot est surreprésenté (ex. “cat”) relié à plusieurs images (chats, lions, etc.), peut générer confusion

Lorsqu'un **mot** textuel (ex. “cat”) entretient une **synergie** élevée avec plusieurs **images** (chats domestiques, lions, tigres, etc.) au sein d'un **DSL multimodal**, la **dynamique** de mise à jour des ω peut exhiber des **oscillations** entre ces différentes options, traduisant une **confusion** (ou un “clignotement”) dans l'attribution des liens. Cette section (8.7.3.1) illustre comment un mot **trop générique** ou **polysémique** — relié à plusieurs entités d'une autre modalité — peut déclencher une **instabilité** locale dans le **SCN**.

Considérons le **mot** “cat” apparaissant dans des documents textuels, et un ensemble d'**images** décrivant divers félins (chat domestique, lion, tigre, etc.). Si l'on dispose d'une **fonction** $S(\text{“cat”, image}^k)$ qui évalue la *pertinence* sémantique, on peut très bien observer des valeurs S toutes relativement élevées — par exemple :

$$S(\text{“cat”, chat_domestique.jpg}) \approx 0.85, \quad S(\text{“cat”, lion.jpg}) \approx 0.82, \quad S(\text{“cat”, tigre.jpg}) \approx 0.80.$$

Puisque “cat” recouvre un champ lexical large (chats domestiques, félins sauvages), la **mise à jour** des pondérations $\omega_{\text{“cat”, img}^k}$ tentera de **renforcer** parallèlement les liens vers “chat_domestique.jpg”, “lion.jpg”, “tigre.jpg”, etc. À défaut d'un mécanisme discriminant plus **fort** (inhibition, recuit simulé, ou un ratio de similarité), on peut assister à un **phénomène** d'indécision où, au fil des itérations, la liaison $\omega_{\text{“cat”, img}^k}$ “balance” entre l'une et l'autre image.

La règle DSL (2.2) indique :

$$\omega_{\text{cat}, \text{img}^k}(t+1) = \omega_{\text{cat}, \text{img}^k}(t) + \eta [S(\text{cat}, \text{img}^k) - \tau \omega_{\text{cat}, \text{img}^k}(t)].$$

Si $S(\text{cat}, \text{img}^k) \approx \text{même ordre de grandeur}$ pour plusieurs k , la **dynamique** peut faire croître la pondération vers l'une ($\omega_{\text{cat}, \text{lion.jpg}} \uparrow$), puis, à l'itération suivante, détecter que l'autre ($\text{cat.chat_domestique.jpg}$) est également légitime, ajustant alors $\omega_{\text{cat.chat_domestique.jpg}}$ à la hausse, etc. L'absence d'un **mécanisme** supplémentaire pour *sélectionner* la meilleure correspondance entraîne un cycle d'**indécision**.

Un **cas** simplifié :

- $S(\text{cat.chat_domestique.jpg}) = 0.80,$
- $S(\text{cat.lion.jpg}) = 0.79,$

Ces valeurs très **proches** fournissent un **flux** de renforcement quasi équivalent vers deux liens $\omega_{(\text{cat.chat_domestique})}$ et $\omega_{(\text{cat.lion})}$. Si la dynamique DSL est *sensible* (i.e. η pas trop petit, τ modéré), on peut voir la **pondération** pencher un temps pour “lion.jpg”, puis le “chat_domestique.jpg” reprendre l'avantage, provoquant des **oscillations**.

Cette **oscillation** correspond à une *confusion* du point de vue sémantique. Le mot “cat” rebondit entre *chat domestique* et *lion*, sans se “fixer” clairement. Dans un **SCN** plus large, ce phénomène peut **perturber** l'interprétation, rendant difficile de déterminer si “cat” étiquette davantage “lion.jpg” ou “chat_domestique.jpg”. Il peut également **entraîner** des oscillations dans d'autres liens connexes, notamment lorsque des mots associés à “cat” se retrouvent alternativement liés à l'image du lion ou à celle du chat domestique, sans stabilisation claire.

Le **score** $S(\text{cat}, \text{img}^k)$ est relativement **similaire** pour plusieurs images, ce qui empêche un écart net permettant de trancher définitivement. L'**absence d'inhibition** amplifie ce phénomène, car la pondération $\omega_{(\text{cat}, \cdot)}$ n'est pas contrainte, permettant une répartition sur plusieurs img^k qui peuvent alterner en dominance d'une itération à l'autre. La **polysémie** du mot “cat” en anglais accentue encore cette indécision, puisqu'il peut désigner un **chat domestique** mais aussi des **félins sauvages** comme les lions et les tigres. Sans une hiérarchie explicite (ex. distinction entre “domestic cat” et “wild cat”), la dynamique reste floue et oscille entre plusieurs interprétations possibles.

Dans un *cas-limite*, la dynamique DSL :

$$\omega_{(\text{cat}, \text{img}^k)}(t+1) = (1 - \eta \tau) \omega_{(\text{cat}, \text{img}^k)}(t) + \eta S(\text{cat}, \text{img}^k)$$

ne favorise pas nettement img^α sur img^β si les valeurs S respectives sont de même ordre. La **confusion** résulte du fait que ω **peut** converger vers plusieurs attracteurs locaux de quasi même énergie (ou bien *osciller* autour d'eux).

Le **score** $S(\cdot, \cdot)$ issu de la similarité *strictement* textuelle (ou un embedding sémantique global) ne discrimine pas toujours des nuances cruciales. D'un point de vue **opérationnel**, la confusion “cat” = “lion” ou “chat domestique” peut nuire à l'**annotation** ou la **recherche** d'images — on ne sait plus si le mot “cat” renvoie à un chat de maison ou à un félin sauvage.

On peut imaginer une hiérarchisation plus fine où “**domestic cat**” se distingue de “**wild cat** (lion, tigre)”. Le DSL pourrait alors dédoubler le nœud “cat” en deux entités $\mathcal{E}_{\text{cat_domestic}}$ et $\mathcal{E}_{\text{cat_wild}}$. Sinon, sans ce niveau de détail, la dynamique se heurte à une ambiguïté persistante.

Conclusion

Lorsqu'un **mot** (ex. “cat”) se retrouve **fortement** connecté à plusieurs images (chats, lions, etc.), la **dynamique** du DSL peut **osciller** entre différentes pondérations $\omega_{(\text{cat}, \text{img})}$. Cette **“surreprésentation”** s'accompagne souvent de *valeurs de synergie* trop proches, et l'on voit la pondération ω basculer d'un “félidé” à l'autre. Sur le plan **mathématique**, le DSL manque alors de “critères de discrimination” pour fixer un lien stable, générant un “clignotement” (oscillation) ou une **confusion**. C'est le **symptôme** d'un **mot** générique ou polysémique, d'une

synergie insuffisamment discriminante et de l'absence de **mécanismes** tels que l'inhibition, le recuit ou le splitting, qui permettraient de départager ou de spécialiser les liens.

La suite (8.7.3.2, 8.7.3.3) discutera des **correctifs** envisageables (inhibition, multiplicité de noeuds, etc.) pour **stabiliser** la dynamique, mieux **discerner** le sens contextuel de "cat" et éviter ce va-et-vient improductif.

8.7.3.2. Inhibition multimodale : on restreint la somme des liaisons “texte–images”

Dans un DSL (Deep Synergy Learning) à composante **multimodale**, il arrive parfois que les entités de deux modalités (par exemple, “texte” et “images”) aient tendance à **surconnecter** entre elles, formant trop de liaisons $\omega_{i,j}$ et générant ainsi une **densité** excessive ou des **confusions** dans le réseau. Pour **rééquilibrer** ces interactions, l'**inhibition multimodale** (ou *cross-modality inhibition*) instaure un **frein** à la prolifération de liens, en imposant qu'il existe une **limite** ou un **coût** si la somme de toutes les pondérations “texte–images” devient trop élevée. Cette approche peut s'avérer cruciale afin de **forcer** la sélectivité des liens et de **stabiliser** la dynamique globale du Synergistic Connection Network.

A. Motivation : Équilibrer les Modèles Multimodaux

Dans un SCN multimodal, les entités de type **texte** (paragraphes, mots-clés, documents) peuvent avoir de la **synergie** avec de nombreuses **images** (visuelles), aboutissant potentiellement à un **grand** nombre de liaisons $\omega_{t,v}$. Toutefois, un trop-plein de **liaisons** texte–images conduit à un **réseau** trop dense, rendant difficile la distinction des **associations** réellement pertinentes. De plus, une certaine **modalité**, comme le texte, peut **dominer** en établissant massivement des connexions, ce qui fausse l’auto-organisation en masquant les vraies correspondances fines.

L'**inhibition multimodale** propose alors d'**introduire** un mécanisme qui **pénalise** la somme globale $\sum_{t \in T, v \in V} \omega_{t,v}$ (ou une variante), de sorte à **limiter** l'expansion de ces liens.

On ajoute, dans la **fonction** d'énergie ou dans la **règle** de mise à jour des ω , un **terme** qui “coûte” davantage dès lors que la **somme** $\sum_{t,v} \omega_{t,v}$ (texte–images) devient importante. Sur le plan **mathématique**, cela oriente la **dynamique** du DSL à **choisir** plus sélectivement quelques liens texte–image bien synergiques, au lieu d’élargir tous les couplages de façon indifférenciée. Le résultat **attendu** est un **SCN** plus économique en liaisons inter-modales, et donc plus lisible et plus stable.

B. Formulation Mathématique de l’Inhibition Cross-Modality

Considérons la forme générale d'une **énergie** J traitée par un DSL (voir chap. 2.2.2, 7.2) :

$$\mathcal{J}_0(\Omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} \omega_{i,j}^2 ,$$

terme "synergie" régularisation linéaire ou quadratique

où Ω désigne l'ensemble des $\omega_{i,j}$. Pour **inhiber** la modalité “texte–images”, on ajoute :

$$\mathcal{J}_{\text{multi}}(\boldsymbol{\Omega}) = \mathcal{J}_0(\boldsymbol{\Omega}) + \gamma_{\text{cross}} F(\{\omega_{t,v}\}),$$

avec $t \in \mathcal{T}$ (texte), $v \in \mathcal{V}$ (images), et $\gamma_{\text{cross}} \geq 0$ comme **coefficients de pénalisation**. La **fonction F** peut être :

- $F = (\sum_{t \in \mathcal{T}, v \in \mathcal{V}} \omega_{t,v})^2$.
 - $F = \sum_{t,v} \omega_{t,v}^2$.
 - Un **mix** ou d'autres variantes (somme des valeurs absolues, etc.).

Plus la somme $\sum_{t,v} \omega_{t,v}$ s'élève, plus le **terme** $\gamma_{\text{cross}} F$ grandit, renforçant la “dissipation” de ces liens.

Avec la forme la plus simple $F = (\sum_{t,n} \omega_{t,n})^2$, on a :

$$\frac{\partial F}{\partial \omega_{t_0, v_0}} = 2 \sum_{t, v} \omega_{t, v},$$

de sorte que la descente de gradient injecte un **terme négatif** proportionnel à $\sum_{t, v} \omega_{t, v}$ dans la mise à jour de ω_{t_0, v_0} . Autrement dit, **plus** la somme “texte–image” est déjà grande, **plus** on va freiner l’augmentation de chaque ω_{t_0, v_0} . Ce mécanisme **réduit** la propension du réseau à créer trop de liens entre \mathcal{T} et \mathcal{V} .

On introduit une **compétition** sur les liens “texte–images” où, si certains liens sont déjà forts, **augmenter** encore d’autres liens inter-modaux devient plus coûteux (à cause du terme $\gamma_{\text{cross}} F$). En pratique, cela **encourage** la spécialisation — seules les liaisons jugées vraiment **pertinentes** (score de synergie élevé) parviendront à maintenir un niveau $\omega_{t, v}$ important, tandis que les autres seront “**rabolées**”.

C. Application dans un DSL Multimodal

Supposons qu’on gère un corpus **texte** comprenant des captions et des descriptions ainsi qu’un ensemble d’**images**. Sans **inhibition**, le DSL peut relier un texte \mathcal{E}_t à un grand nombre d’images $\{\mathcal{E}_v\}$ avec des pondérations $\omega_{t, v}$ moyennement élevées, ce qui engendre une confusion pour l’**indexation**. En revanche, avec une **inhibition multimodale**, le réseau est contraint de **retenir** principalement les liens texte–image dont la **similarité** $S(t, v)$ est la plus marquée, décourageant ainsi les pondérations intermédiaires et limitant la sur-connexion.

Cette inhibition “cross-modality” n’affecte pas les **liaisons** texte–texte, les **liaisons** image–image ni les liens d’une autre modalité comme l’audio.

Seules les **connexions** entre **texte** et **images** se voient freinées lorsque leur somme collective dépasse un seuil implicite. Ainsi, on obtient un **SCN** mieux organisé où les entités texte forment un bloc structuré, les entités image un autre, et seules **certaines** paires (texte, image) “franchissent” la barrière entre modalités grâce à une synergie **réellement** forte.

D. Pistes Mathématiques et Extensions

Une variante consiste à imposer, pour **chaque** entité texte t , une **contrainte** $\sum_{v \in \mathcal{V}} \omega_{t, v} \leq \Lambda_t$. Ainsi, on limite le “nombre” total de liens (ou la somme des pondérations) partant de \mathcal{E}_t . Cela revient à un schéma “compétitif” local, où le texte t doit “choisir” l’image la plus alignée, plutôt que de se connecter à dix images moyennement liées. Au niveau **mathématique**, on peut réaliser cela via un terme de type $\sum_t (\sum_v \omega_{t, v} - \Lambda_t)^2$.

Il est possible de **cumuler** une **inhibition intra-modale**, qui limite $\sum_{i, j \in \text{texte}} \omega_{i, j}$ ou $\sum_{v_1, v_2 \in \text{images}} \omega_{v_1, v_2}$ afin d’éviter la formation de clusters excessivement denses dans une seule modalité, et une **inhibition cross-modality**, qui restreint $\sum_{t, v} \omega_{t, v}$ pour empêcher un déséquilibre dans les connexions entre le texte et l’image.

Cette approche mixte garantit un **équilibre** global des liaisons, limitant simultanément la surdensité texte–texte, image–image ou texte–image.

L’**ajout** d’un terme de pénalisation $\gamma_{\text{cross}} (\sum_{t, v} \omega_{t, v})^2$ accentue la **non-convexité** de la fonction d’énergie, ce qui peut rendre la **convergence** plus délicate. On peut alors employer des heuristiques stochastiques (recuit simulé, chap. 7.3) ou un **façonnage** progressif (on augmente γ_{cross} au fil du temps) pour aider la dynamique à trouver un **arrangement** convenable.

Conclusion

L’**inhibition multimodale** (ou “cross-modality”) dans un **DSL** constitue un levier essentiel pour **limiter** la somme totale des liaisons entre deux modalités (ex. texte–images). **Mathématiquement**, on intègre dans la fonction d’énergie (ou directement dans la règle de mise à jour) un **terme** qui pénalise la croissance simultanée de trop nombreuses $\omega_{t, v}$. Les gains obtenus sont :

- **Sélectivité accrue** : on évite que chaque phrase texte se connecte faiblement à toutes les images, ou vice versa.
- **Équilibre** inter-modal : une modalité ne “monopolise” pas l’attention en saturant la matrice ω .

- **Clarté** de structure : en rendant plus **rares** les liaisons texte–image, on rehausse la pertinence des liens qui persistent, et on obtient des **clusters** plus lisibles.

Il est possible de combiner ce principe avec des **budgets** locaux (un nombre maximum de liaisons par entité), de l'**inhibition intra-modale** et des **méthodes** stochastiques (recuit) pour stabiliser la dynamique tout en préservant la flexibilité du **Synergistic Connection Network**.

8.7.3.3. Recuit simulé (Chap. 7.3) comme levier pour réorganiser les clusters si conflit

Dans un **SCN** (Synergistic Connection Network) piloté par un **DSL** (Deep Synergy Learning), il peut arriver que certains **conflits** de répartition se produisent lorsqu'une entité est “disputée” par plusieurs clusters, ou des sous-groupes entiers se coincent dans une configuration **sous-optimale** (mauvaise séparation, liens incorrects). Comme expliqué au **chapitre 7.3**, le **recuit simulé** apporte une **dimension stochastique** permettant de “secouer” le réseau et d’**échapper** à ces minima locaux. La présente section (8.7.3.3) décrit comment ce **mécanisme** s’applique en pratique pour **réorganiser** les pondérations $\omega_{i,j}$ lorsque des **conflits** surviennent ou que la configuration n’est pas satisfaisante.

A. Origine du Conflit et Besoin de Réorganisation

Un **conflit** naît souvent lorsqu’un **nœud** (ou un petit groupe de nœuds) bénéficie d’une **synergie** élevée avec plusieurs **clusters** rivaux. Au fil du temps, les liaisons $\omega_{i,j}$ entre l’entité i et le cluster C_1 augmentent, mais la synergie avec C_2 reste non négligeable. Cela entraîne l’apparition d’**oscillations**, où l’entité i bascule d’un sous-ensemble à l’autre.

Sur le plan **mathématique**, aucune configuration n'est assez “forte” pour emporter définitivement i , et le **SCN** se retrouve dans un état de tension. Dans certains cas, il converge vers un **compromis** insatisfaisant (pondérations moyennes) ou continue à **osciller** sans se fixer.

Même en l’absence de conflit direct, un **SCN** peut se stabiliser localement dans un **minimum** d’énergie $J(\Omega^*)$ plus élevé que d’autres configurations possibles (voir 7.2.2.2 sur les minima locaux). Il n’existe plus d’**incrément** local suffisant pour franchir la barrière d’énergie et rejoindre une répartition **globalement** plus favorable. On parle de **verrouillage** local lorsque la mise à jour déterministe par descente de gradient ne parvient pas à “sauter” au-dessus d’un “col” d’énergie.

B. Recuit Simulé : Injection de Bruit “Tempéré”

Le **recuit simulé** (Chap. 7.3) consiste à ajouter un **terme** aléatoire $\sigma(t) \xi_{i,j}(t)$ à la mise à jour de chaque pondération $\omega_{i,j}$. Ainsi, l’équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \sigma(t) \xi_{i,j}(t),$$

où :

$\sigma(t)$ est la “**température**” (amplitude du bruit) à l’itération t ,

$\xi_{i,j}(t)$ est une **variable** de bruit (souvent gaussienne ou uniforme centrée),

autorise des *fluctuations* aléatoires autour de la logique stricte $\eta[S - \tau \omega]$. Quand $\sigma(t)$ est relativement **grand**, ce “bruit” peut “**briser**” les configurations trop rigides.

Le **recuit** se déroule en deux phases principales :

- **Chauffage** (ou début à température élevée) : $\sigma(t) \approx \sigma_0$ assez grand. On effectue un certain nombre d’**itérations** où les ω subissent des perturbations importantes, permettant des “sauts” entre configurations éloignées.
- **Refroidissement** : on diminue $\sigma(t)$ au fil du temps ($\sigma(t+1) < \sigma(t)$), rendant les perturbations plus modestes. On se **stabilise** alors dans un nouveau **minimum** d’énergie potentiellement plus global.

Ce procédé renvoie à l'**inspiration** de la *métallurgie*, où un métal chauffé et lentement refroidi (recuit) trouve une structure cristalline plus stable.

C. Action du Recuit en Situation de Conflit

En **phase chaude** (température élevée), les liaisons $\omega_{i,j}$ en conflit peuvent être **secouées** hors de l'équilibre local. Par exemple, si un nœud i oscille entre C_1 et C_2 , le bruit peut “**casser**” momentanément la liaison ω_{i,C_1} pour encourager ω_{i,C_2} à croître, ou vice versa. L'avantage est que le réseau peut **explorer** divers rattachements, sans rester bloqué dans un compromis. Après plusieurs itérations, on amorce la **phase de refroidissement** où la dynamique redevient plus déterministe, et les **liens** les plus cohérents avec S finissent par se consolider.

Si la configuration Ω^* atteint localement un “mauvais” minimum, le recuit donne la possibilité à certaines $\omega_{i,j}$ d’aller à l’encontre du gradient local. Ainsi, on franchit parfois la barrière d’énergie, aboutissant à une nouvelle configuration Ω' de **meilleure** énergie $J(\Omega') < J(\Omega^*)$. En pratique, on obtient un **SCN** qui réorganise plus finement ses clusters ou résout la **confusion** de pondérations conflictuelles.

D. Mise en Œuvre Concète

Le chapitre 7.3 présente plusieurs **schémas** pour $\sigma(t)$:

- **Exponentiel** : $\sigma(t) = \sigma_0 \cdot \alpha^t$.
- **Logarithmique** : $\sigma(t) = \frac{\sigma_0}{\ln(t+t_0)}$.
- **Linéaire** : $\sigma(t) = \max(0, \sigma_0 - \lambda t)$.

Le choix dépend des préférences en termes de “vitesse” de refroidissement et de robustesse. En général, on souhaite un **démarrage** suffisamment **chaud** pour autoriser de grandes fluctuations, puis un **refroidissement** lent pour **stabiliser** solidement le réseau.

Trop de bruit (température excessive ou prolongée) peut faire “exploser” les ω aléatoirement ou empêcher toute convergence. Trop peu de bruit ne résout pas le conflit. On cherche donc un **calibrage** adéquat pour $\sigma(t)$. Des heuristiques comme le “recuit simulé adaptatif” adaptent σ selon les progrès de la convergence.

On peut cesser le recuit après un certain **nombre** d’itérations fixes (plan de recuit déterministe) ou lorsqu’un **critère** d’énergie / de variation $\|\Omega(t+1) - \Omega(t)\|$ tombe en dessous d’un seuil. L’**objectif** est de préserver suffisamment de temps pour **réorganiser** les clusters, tout en évitant de maintenir un bruit trop fort quand on est proche d’une bonne solution.

Conclusion

Le **recuit simulé** (Chap. 7.3) se révèle particulièrement **efficace** pour **réorganiser** un **SCN** lorsqu’apparaissent des **conflits** entre clusters ou qu’on s’est enfermé dans une configuration **sous-optimale**. L’introduction d’un **bruit** maîtrisé $\sigma(t) \xi_{i,j}(t)$:

- **Autorise** des “sauts” hors des minima locaux,
- **Brise** temporairement des liens conflictuels ou trop figés,
- **Permet** à la dynamique DSL de redessiner les groupes et de consolider une répartition plus favorable à la **synergie** globale.

Après un certain laps de temps (phase de refroidissement), le réseau se **stabilise** dans un nouvel état plus **cohérent**. Le **recuit** répond ainsi à la nécessité de **Résolution** de conflits et de **reconfiguration** pour un **SCN** multimodal cherchant à se **clustérer** de manière auto-organisée.

8.8. Apports en Applications Concrètes

Après avoir présenté, dans les sections précédentes, les principes du DSL (Deep Synergy Learning) appliqués à la **fusion multimodale** (chap. 8.1 à 8.7), il importe de souligner **comment** ces approches se déclinent dans des **applications concrètes**. Le chapitre 8.8 met l'accent sur la **mise en pratique**, on y verra comment un **SCN** (Synergistic Connection Network) gère simultanément des **données visuelles** (images, vidéos) et des **données textuelles / auditives** afin de produire des fonctions de **recherche**, **d'annotation**, ou de **reconnaissance** multimodale.

- Dans la première section (8.8.1), nous abordons l'**annotation d'images** par le **langage** (mots, phrases),
- Puis, en (8.8.2), la **reconnaissance** audio-visuelle,
- Et en (8.8.3), un exemple où l'on combine IA **symbolique** et IA **sub-symbolique** sur des flux multimodaux.

8.8.1. Annotation d'Images par le Langage

L'un des exemples d'application typiques du **DSL** multimodal concerne l'**association** entre des entités "images" et des entités "mots" ou "phrases". L'objectif est de **retrouver** quels mots ou phrases "légendent" telle image, et inversement, en s'appuyant sur la **synergie** entre ces deux modalités.

8.8.1.1. Entités "images" et entités "mots/phrases"

Dans une approche **multimodale**, il est essentiel de représenter simultanément des **entités** visuelles et des **entités** textuelles au sein d'un **SCN** (Synergistic Connection Network). Les travaux précédents (voir la référence au chapitre 2.2 pour la définition générale de la *synergie*) indiquent que chaque **entité** est caractérisée par un **embedding** propre, et qu'une fonction de **synergie** S permet de comparer deux entités, même si elles ne partagent pas la même modalité. La présente section (8.8.1.1) illustre spécifiquement comment introduire des **images** et des **mots/phrases** dans un **DSL** (Deep Synergy Learning), afin de construire un couplage automatique image–texte, s'apparentant à un processus de **légendage** ou d'**annotation** auto-organisée.

A. Représentation des Entités Images et Textes

La première étape consiste à définir un **ensemble d'images**, noté $\{\mathcal{I}_1, \dots, \mathcal{I}_N\}$, et un **ensemble de textes**, noté $\{\mathcal{T}_1, \dots, \mathcal{T}_M\}$. Chaque **image** \mathcal{I}_i se voit associée à un **vecteur** $\mathbf{v}_i^{(\text{img})}$ (embedding visuel), par exemple généré par un **réseau CNN** ou par un **autoencodeur**. En parallèle, chaque **texte** \mathcal{T}_m (qu'il s'agisse d'un **mot** unique ou d'une **phrase** entière) est transformé en un **embedding** $\mathbf{v}_m^{(\text{txt})}$, potentiellement dérivé de **GloVe**, **Word2Vec**, ou d'un **modèle Transformer** (BERT, etc.). Cette mise en correspondance permet de considérer \mathcal{I}_i et \mathcal{T}_m comme deux **entités** d'un même réseau, à savoir $\mathcal{E}_i^{(\text{img})}$ et $\mathcal{E}_m^{(\text{txt})}$.

B. Définition de la Synergie entre Image et Texte

La **fonction de synergie** $S(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_m^{(\text{txt})})$ capture la **compatibilité** sémantique entre un **embedding** visuel et un **embedding** textuel. Une manière typique de la formuler est d'utiliser la **similarité cosinus** :

$$S(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_m^{(\text{txt})}) = \frac{\mathbf{v}_i^{(\text{img})} \cdot \mathbf{v}_m^{(\text{txt})}}{\|\mathbf{v}_i^{(\text{img})}\| \|\mathbf{v}_m^{(\text{txt})}\|} .$$

Dans de nombreux cas, on peut recourir à des **réseaux** pré-entraînés (voir la référence à la section 2.2.1.2 sur la définition générale de la fonction S), qui projettent images et textes dans un **espace** latent commun. Les valeurs de S ainsi obtenues reflètent la proximité sémantique effective entre l'image et le fragment textuel.

C. Mise à Jour des Pondérations dans le DSL

Le **SCN** (voir la référence à la section 2.2.2.1 sur la mise à jour des **pondérations**) introduit une matrice ω reliant **entités images** et **entités textes**. Notons $\omega_{i,m}$ la **pondération** reliant l'image $\mathcal{E}_i^{(\text{img})}$ et le texte $\mathcal{E}_m^{(\text{txt})}$. Conformément au **DSL**, on dispose d'une **règle** d'évolution :

$$\omega_{i,m}(t+1) = \omega_{i,m}(t) + \eta \left[S\left(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_m^{(\text{txt})}\right) - \tau \omega_{i,m}(t) \right] .$$

Le **terme** η (taux d'apprentissage) régule la vitesse d'adaptation, tandis que τ évite que $\omega_{i,m}$ ne croisse indéfiniment. Au fil des itérations, les **paires** image–texte affichant une **synergie** S plus forte consolident $\omega_{i,m}$, tandis que les autres régressent. Il est possible, par ailleurs, de recourir à des **mécanismes d'inhibition** ou de **recuit simulé** (références en 8.7.3.2 et 8.7.3.3) pour affiner la structure ou échapper à des configurations sous-optimales.

D. Interprétation en Annotation Automatique

Lorsque $\omega_{i,m}$ atteint une valeur notable, on considère que l'image \mathcal{I}_i se trouve **associée** au mot (ou à la phrase) \mathcal{T}_m . D'un point de vue pratique, cela se traduit par un **légendage** ou une **annotation** de l'image \mathcal{I}_i via le texte \mathcal{T}_m . À l'issue de la convergence, le **SCN** fait émerger de façon **auto-organisée** un ensemble de liaisons fortes $\omega_{i,m}$ signalant quels mots décrivent quelles images. Cette logique permet aussi d'identifier des **clusters** multimodaux où un groupe d'images semblables reliera simultanément un sous-ensemble de mots communs, révélant un **thème** transversal.

Formellement, si l'on opère une coupe $\omega_{i,m} > \theta$ pour un certain seuil θ , on isole toutes les **paires** (i, m) jugées pertinentes. Cela se conforme aux principes de **parsimonie** (voir la section 2.2.3 sur la limitation du nombre de liaisons) et aboutit à un graphe bipartite (images–textes) plus clairsemé, où chaque image se connecte uniquement aux segments textuels les plus appropriés.

E. Avantages Mathématiques et Pratiques

Le **DSL** ne requiert pas d'apprentissage supervisé strict ni de labels fixés, dans la mesure où la **synergie** se fonde sur une mesure préétablie (distance, similarité, projection multimodale). D'un point de vue théorique, cette approche garantit une grande **flexibilité** pour gérer des entités **hétérogènes**, à condition de posséder une fonction S capable de comparer un **embedding** visuel avec un **embedding** textuel. Les références à la section 8.7.3.2 (inhibition multimodale) ou 8.7.3.3 (recuit simulé) montrent comment stabiliser la dynamique ou résoudre les conflits si de multiples mots tentent de décrire la même image de façon indistincte, ou si plusieurs images s'attachent à un même mot trop générique.

Conclusion

Dans le cadre du **Synergistic Connection Network**, il est possible de définir des **entités** "images" et des **entités** "mots/phrases" en leur attribuant des **embeddings** respectifs et en postulent une **synergie** $S(\text{img}, \text{txt})$. La **mise à jour** $\omega \leftarrow \omega + \eta [S - \tau \omega]$ (voir la section 2.2.2.1) conduit à renforcer les liens images–textes pertinents et à atténuer les autres, ce qui se traduit en un **processus de légendage** automatique. Les **clusters** qui surgissent à partir de ces liens élevés fournissent une **structure** auto-organisée associant, d'une part, un sous-groupe d'images et, d'autre part, un sous-groupe de mots. Cette méthode s'inscrit dans une perspective **non supervisée**, conférant au réseau un **pouvoir** adaptatif pour gérer simultanément différentes modalités et actualiser en continu les correspondances image–texte.

8.8.1.2. DSL auto-organise $\omega_{\text{image},\text{text}}$ pour identifier quels mots légendent quelles images

Dans un **scénario multimodal** mettant en jeu des **entités** de type **image** et **texte**, il est possible d'exploiter le **Deep Synergy Learning (DSL)** afin de faire émerger automatiquement la correspondance entre un contenu visuel et des tokens textuels pertinents. L'objectif est de laisser la dynamique interne du **SCN** (Synergistic Connection Network) construire une **matrice de pondérations** $\omega_{\text{image},\text{text}}$, laquelle indique en fin de convergence les **couples** (image, mot) jugés les plus cohérents. Les considérations mathématiques et les propriétés d'**auto-organisation** présentées plus haut (voir la référence à la section 2.2.2) s'appliquent alors pour renforcer les paires dont la **synergie** est élevée et pour affaiblir les associations moins pertinentes.

A. Représentation des Images et Mots

Les entités visuelles sont représentées par des **vecteurs** $\mathbf{x}_i^{(\text{img})} \in \mathbb{R}^{d_{\text{img}}}$, dont l'obtention repose souvent sur un **réseau convolutif** (CNN) ou sur un **encodeur** (voir la discussion sur les embeddings en section 8.8.1.1). De même, les **entités** textuelles (mots ou tokens) sont décrites par des **embeddings** $\mathbf{x}_j^{(\text{txt})} \in \mathbb{R}^{d_{\text{txt}}}$, obtenus par Word2Vec, GloVe ou un **modèle** Transformer. Afin de quantifier la **synergie** entre une image \mathcal{I}_i et un mot \mathcal{T}_j , on définit souvent

$$S(i,j) = \text{cos similarity}(\mathbf{x}_i^{(\text{img})}, \mathbf{x}_j^{(\text{txt})}) = \frac{\mathbf{x}_i \cdot \mathbf{x}_j}{\|\mathbf{x}_i\| \|\mathbf{x}_j\|} .$$

Cette formule mesure la **proximité** dans le **domaine** latent, et plus $S(i,j)$ est grand, plus l'image i et le mot j sont censés véhiculer un même contenu conceptuel.

B. Pondérations $\omega_{\text{image},\text{text}}$ et Dynamique du DSL

La **matrice** ω reliant l'entité image \mathcal{I}_i et l'entité texte \mathcal{T}_j évolue selon la règle d'**auto-organisation** (voir la référence à la section 2.2.2.1), de la forme

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] ,$$

où η est le **taux** d'apprentissage et τ un **coefficient** limitant. Si $S(i,j)$ est élevé, la pondération $\omega_{i,j}(t)$ croît progressivement, révélant l'**affinité** image–texte. Inversement, si la synergie est faible ou nulle, le terme $[S(i,j) - \tau \omega_{i,j}(t)]$ demeure négatif, entraînant la réduction de $\omega_{i,j}(t)$. À mesure que les itérations avancent, les paires (image, mot) considérées comme importantes s'établissent avec une **pondération** forte, tandis que les autres chutent proche de zéro.

Dans certains cas, on peut inclure un **terme d'inhibition** (voir la section 8.7.3.2) pour empêcher qu'une image ne se connecte simultanément à trop de mots, ou qu'un mot ne s'étale sur trop de visuels. On peut aussi introduire un **bruit** simulant le **recuit** (voir la section 8.7.3.3) pour échapper à des minima locaux.

C. Équations d'Énergie et Couplages Image–Texte

On peut formaliser cette mécanique en postulant une **fonction** d'énergie

$$\mathcal{J}(\omega) = - \sum_{i,j} \omega_{i,j} S(i,j) + \frac{\tau}{2} \sum_{i,j} [\omega_{i,j}]^2 ,$$

dont la **descente** de gradient implicite recouvre la mise à jour ci-dessus. Les paires (image, mot) pour lesquelles $S(i,j)$ est élevé offrent un gain énergétique, conduisant à l'**émergence** de liens $\omega_{i,j}$ importants. Si l'on rajoute un **terme** de pénalisation pour liaisons excessives, la fonction \mathcal{J} devient **non convexe**, renvoyant alors à la nécessité éventuelle d'algorithmes stochastiques (comme le recuit).

D. Utilité Pratique : Légendes Automatiques et Filtrage

Après **convergence**, il se produit une **auto-organisation** où chaque image \mathcal{I}_i se trouve associée à un petit ensemble de mots $\{\mathcal{T}_j\}$ ayant $\omega_{i,j}$ élevé. Interpréter cette relation en termes de **légendes** ou d'**annotations** s'avère naturel. Lorsque $\omega_{i,j}$ atteint une valeur élevée, cela signifie que "l'image i est décrite par le mot j ". De même, on peut traiter la requête inverse : un mot \mathcal{T}_j est relié fortement à l'image \mathcal{I}_i , ce qui facilite un mécanisme de **recherche** cross-modale.

En pratique, on peut fixer un **seuil** θ et ne considérer que les paires $\omega_{i,j} > \theta$. Cela restitue les **mots** pertinents pour chaque image. Les **clusters** qui se forment (voir la référence à la section 8.7.2) montrent des sous-groupes d'images partageant un vocabulaire proche et inversement, des mots rattachés à des visuels similaires. Cette structuration constitue un **processus** de classification ou d'**indexation** non supervisé, piloté par la **synergie**.

Conclusion

Le **DSL**, en ajustant automatiquement $\omega_{\text{image}, \text{text}}$, parvient à repérer quels mots légètent quelles images de manière **auto-organisée**. La règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

renforce les couples (image, mot) à **forte** synergie et affaiblit les autres, ce qui se traduit concrètement par la **découverte des meilleures légendes**. Au-delà, cette structure bipartite formée par ω offre une vision **multimodale** du corpus, mettant en évidence les liaisons transverses entre entités hétérogènes et fournissant ainsi une base pour la **recherche** et la **classification** sémantique dans le réseau.

8.8.1.3. Exemple : un dataset “Flickr8k” ou autre, observation des clusters (chats, chiens, personnes...)

Dans une configuration **multimodale** combinant des **images** et du **texte** (légendes, mots, etc.), il est souvent illustratif d'étudier des bases telles que **Flickr8k** (ou **Flickr30k**, **MS-COCO**, etc.) où chaque image comporte plusieurs **légendes** décrivant son contenu. L'objectif est de montrer comment un **SCN** (Synergistic Connection Network) auto-organise les **entités** (visuelles, textuelles) en **clusters** cohérents, par exemple “chats”, “chiens”, “personnes”, etc. La dynamique du **DSL** (Deep Synergy Learning) assure cette classification spontanée via les **synergies** entre images, entre textes, et entre image et texte, évitant ainsi la supervision explicite.

A. Présentation du Dataset “Flickr8k”

Les ensembles d’images de **Flickr8k** comportent environ huit mille **photos** extraites de la plateforme Flickr, chacune associée à **cinq** légendes (phrases courtes décrivant la scène, le ou les objets présents). Sur le plan **multimodal**, cela crée une structure où il existe :

- Des **entités** $\{\mathcal{E}_i^{(\text{img})}\}$, représentant chacune un embedding visuel (voir la section **8.8.1.1** pour la forme de l’embedding).
- Des **entités** $\{\mathcal{E}_j^{(\text{txt})}\}$, correspondant aux tokens ou phrases provenant des légendes, chaque token disposant d’un embedding textuel (Word2Vec, GloVe, BERT, etc.).

B. Mise en Œuvre d’un SCN Multimodal

Pour traduire les images et les légendes en un **réseau** unique, il convient d’instituer :

- Un **embedding** visuel $\mathbf{v}_i^{(\text{img})}$ pour chaque image I_i . Cet embedding peut résulter d’un **CNN** comme VGG ou ResNet.
- Un **embedding** textuel $\mathbf{w}_j^{(\text{txt})}$ pour chaque token (ou pour une phrase entière) issu des légendes associées.
- Une **synergie** $S(\mathcal{E}_i^{(\text{img})}, \mathcal{E}_j^{(\text{txt})})$ capable de quantifier la **compatibilité** entre $\mathbf{v}_i^{(\text{img})}$ et $\mathbf{w}_j^{(\text{txt})}$. On peut se contenter d’une **distance** exponentielle ou d’une **similarité** cosinus entre les vecteurs d’embedding :

$$S(i,j) = \text{cos similarity}(\mathbf{v}_i^{(\text{img})}, \mathbf{w}_j^{(\text{txt})}).$$

Parallèlement, on peut également établir :

- Des synergies **image–image** ($S(\text{img}, \text{img})$) si l’on souhaite grouper les images similaires.
- Des synergies **texte–texte** ($S(\text{txt}, \text{txt})$) si l’on veut refléter la ressemblance sémantique entre différents mots ou phrases.

C. Observation des Clusters (Chats, Chiens, Personnes, etc.)

La **dynamique** DSL (voir la section 2.2.2) amène la matrice $\{\omega_{i,j}(t)\}$ de pondérations à évoluer au fil des itérations. L'équation de mise à jour,

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)],$$

assure le **renforcement** des paires (i, j) présentant une synergie importante (image et tokens proches), tandis que les paires moins pertinentes se voient affaiblies. À mesure que le système converge, chaque ensemble de **liens** ω se **stabilise**, les **images** affichant un **contenu** semblable, comme des chiens, se relient à des mots-clés identiques tels que “dog” ou “canine” et tendent à constituer un **cluster** commun, renforcé par des **tokens** similaires. De même, les **images** représentant des **chats**, des **personnes** ou des **scènes** sportives regroupent chacune un **noyau** de tokens récurrents, formant ainsi des **macro-clusters** homogènes.

D'un point de vue **mathématique**, on peut filtrer les liens $\omega_{i,j}$ dépassant un seuil θ . Les **sous-graphes** émergents sont alors très denses en leur sein (images très connectées aux mêmes légendes ou mots-clés), et peu connectés aux autres blocs. Cette structure traduit un **partage** sémantique. Un cluster “chiens” agrège toutes les images canines et les tokens évoquant un chien, un cluster “chats” rassemble les images félines et les mots “cat”, “kitty”, etc.

D. Validation sur Flickr8k

Dans une étude pratique, on charge toutes les images **Flickr8k** (environ 8000) dans le SCN, on en extrait des **embeddings** (dimension 512 ou 2048 selon le réseau), et pour chaque mot récurrent dans les légendes, on génère un embedding textuel (Word2Vec ou BERT). La fonction de synergie S combine la **similarité** cosinus (voir la section 8.8.1.2). Après un certain nombre d'**itérations**, on note le rassemblement d'**images** appartenant au même thème, et le **DSL** fait apparaître des **macro-clusters** cohérents. Tous les liens $\omega_{i,j}$ conduisent les photos de chiens et les tokens “dog” ou “puppy” à former un bloc, les photos de chats et le token “cat” se concentrent dans un autre bloc, etc. De même, les images montrant des **personnes** en extérieur se lient avec les mots “people”, “person”, “man”, “woman”, etc. Cette segmentation **auto-organisée** reflète la structure sémantique du corpus image-légende sans exiger de supervision externe.

Conclusion (8.8.1.3)

Lorsque l'on applique un **SCN** multimodal aux images et légendes (ou mots) d'un dataset comme **Flickr8k**, la **synergie** entre embeddings visuels et textuels amène l'**auto-organisation** des pondérations $\omega_{i,j}$. Il apparaît alors un **clustering** naturel où chaque bloc regroupe des images et des tokens correspondant à un même concept (chats, chiens, personnes, actions, etc.). Ce phénomène illustre la **capacité** du DSL à repérer spontanément les *catégories* implicites, sans supervision, en s'appuyant simplement sur la **cohérence** locale des similarités image-texte (et éventuellement image-image, texte-texte). Les expériences montrent ainsi, de manière empirique et mathématique, la validité de l'hypothèse selon laquelle la **dynamique** d'un **réseau** (SCN) fondé sur la **synergie** peut **révéler** ou **découvrir** des regroupements hétérogènes pertinents dans un **dataset** multimodal.

8.8.2. Reconnaissance Audio–Visuelle

Au sein d'un **SCN** (Synergistic Connection Network) appliqué à la **reconnaissance audio–visuelle**, on traite simultanément des **flux vidéo** (image, mouvement) et des **flux audio** (sons, paroles, bruitages), en formant des **entités** distinctes pour chacun. L'objectif est de **repérer** les liens $\omega_{i,j}$ forts entre des segments vidéo et des segments audio qui correspondent, par exemple, à la même action ou au même événement (voix d'une personne qui parle, chien qui aboie, clap synchronisé, etc.).

8.8.2.1. Entités “segments vidéo” + entités “segments audio”

Un **SCN** (Synergistic Connection Network) appliqué à l'analyse **audio–visuelle** implique de représenter, d'une part, des **segments vidéo** et, d'autre part, des **segments audio**, chacun tenu pour une **entité** du réseau. La structure **multimodale** ainsi formée fournit un cadre permettant de lier chaque portion visuelle à la portion sonore qui lui correspond, selon une **synergie** S_{AV} . La présente section (8.8.2.1) détaille la manière de **segmenter** et de **caractériser**

les flux vidéo et audio, ainsi que la définition d'une **synergie inter-modale** destinée à guider l'**auto-organisation** des pondérations $\omega_{\mathcal{V}, \mathcal{A}}$.

A. Segments vidéo

La **vidéo** se découpe couramment en **sous-séquences** $\{\mathcal{V}_k\}$, par exemple des intervalles de une à deux secondes, ou des plans détectés par des méthodes de détection de coupure. Chaque **segment** \mathcal{V}_k peut alors être traité comme une **entité** à part entière dans le **SCN**. Sur le plan **mathématique**, on construit un ensemble $\{\mathcal{V}_1, \dots, \mathcal{V}_{n_v}\}$. Pour décrire \mathcal{V}_k , on extrait un **vecteur de features** \mathbf{v}_k , qui peut provenir d'un **CNN** (mesurant la structure spatiale de l'image clé), de **motion vectors** (synthétisant la dynamique), d'un **histogramme** de couleurs ou encore d'un **embedding** sémantique (détection d'objets et de leur position).

D'un point de vue purement **interne** à la modalité vidéo, on peut définir une **synergie** $S_{\text{vid}}(\mathcal{V}_p, \mathcal{V}_q)$ reflétant la similarité entre deux segments (temps similaire, objets similaires, etc.). Cela aide à relier les segments **visuellement** proches. Toutefois, dans un contexte **audio-vidéo**, on se focalise sur la **relation** entre un segment vidéo \mathcal{V}_p et un segment audio \mathcal{A}_r .

B. Segments audio

En **parallèle**, le **flux audio** se décompose en **sous-séquences** $\{\mathcal{A}_l\}$, qui peuvent correspondre à des intervalles de une seconde, ou à des tranches homogènes détectées selon un critère acoustique (variation de spectre, changement de locuteur). Chaque entité audio \mathcal{A}_l est décrite par un **vecteur** \mathbf{a}_l issu de **features** classiques (MFCC, LPC) ou de **représentations** plus récentes (embeddings neuronaux basés sur des spectrogrammes).

Comme pour la vidéo, on pourrait définir $S_{\text{aud}}(\mathcal{A}_l, \mathcal{A}_m)$ afin de mesurer la similarité interne à la modalité audio (deux segments contenant un son similaire). Néanmoins, dans la logique de **fusion** audio-vidéo, on s'intéresse davantage aux **pondérations** $\omega_{\mathcal{V}_p, \mathcal{A}_r}$ entre un segment **vidéo** \mathcal{V}_p et un segment **audio** \mathcal{A}_r , symbolisant leur **synchronicité** ou leur **cohérence**.

C. Synergie Inter-Modale entre Segment Vidéo et Segment Audio

Pour associer un **segment vidéo** \mathcal{V}_p et un **segment audio** \mathcal{A}_r , on définit une **synergie** $S_{\text{AV}}(\mathcal{V}_p, \mathcal{A}_r)$. Cette fonction reflète typiquement la concordance temporelle (la partie audio coïncide-t-elle avec ce moment visuel ?) et la similarité sémantique (ex. un **abolement** détecté dans le flux audio alors qu'on **voit** un chien à l'écran). On peut schématiser :

$$S_{\text{AV}}(\mathcal{V}_p, \mathcal{A}_r) = \alpha \delta_{\text{time}}(\mathcal{V}_p, \mathcal{A}_r) + \beta \text{Sim}_{\text{embedding}}(\mathbf{v}_p, \mathbf{a}_r),$$

où $\delta_{\text{time}}(\mathcal{V}_p, \mathcal{A}_r)$ mesure la superposition temporelle (combien de recouvrement entre l'intervalle vidéo et l'intervalle audio ?), tandis que $\text{Sim}_{\text{embedding}}(\mathbf{v}_p, \mathbf{a}_r)$ quantifie la compatibilité sémantique (ex. un extrait de spectrogramme caractéristique d'un certain son, mis en regard d'un objet reconnu dans la vidéo). Les **coefficients** α et β ajustent l'importance du facteur temporel par rapport à la similarité sémantique.

D. Création d'un SCN Multimodal

Du point de vue du **Synergistic Connection Network**, on dispose de deux **ensembles** d'entités : $\{\mathcal{V}_1, \dots, \mathcal{V}_{n_v}\}$ pour les segments vidéo, et $\{\mathcal{A}_1, \dots, \mathcal{A}_{n_a}\}$ pour les segments audio. Pour chaque paire $(\mathcal{V}_p, \mathcal{A}_r)$, la pondération $\omega_{\mathcal{V}_p, \mathcal{A}_r}(t)$ évolue suivant la **règle** DSL (voir la section 2.2.2 pour la mise à jour des poids) :

$$\omega_{\mathcal{V}_p, \mathcal{A}_r}(t+1) = \omega_{\mathcal{V}_p, \mathcal{A}_r}(t) + \eta [S_{\text{AV}}(\mathcal{V}_p, \mathcal{A}_r) - \tau \omega_{\mathcal{V}_p, \mathcal{A}_r}(t)].$$

Si la **fonction** S_{AV} indique une forte correspondance (par exemple, un son de klaxon pendant que la vidéo montre une voiture), on aura $\omega_{\mathcal{V}_p, \mathcal{A}_r}$ qui s'accroît. À la fin des itérations, les **liaisons** $\omega_{\mathcal{V}_p, \mathcal{A}_r}$ établissent un couplage audio-vidéo plus ou moins dense, révélant les "sous-scènes" où l'on observe un parfait accord entre l'image et le son.

E. Interaction Interne aux Modalités

Bien qu'on s'intéresse principalement aux **corrélations** audio-visuelles, il est souvent souhaitable de laisser le SCN gérer également des pondérations **intra-modales**, c'est-à-dire ω_{V_p, V_q} entre deux segments vidéo, ou ω_{A_l, A_m} entre deux segments audios. Le fait de maintenir ces liens intra-modaux peut renforcer la **cohérence** globale. Si V_p et V_q montrent le même objet en plan rapproché, et si A_l et A_m partagent des similarités de sons, la dynamique du DSL peut croiser ces informations pour stabiliser des **clusters** ou sous-groupes audiovisuels plus riches.

Conclusion

Dans un **SCN** consacré à la reconnaissance **audio-visuelle**, on définit deux **ensembles** d'entités, les **segments vidéo** $\{V_k\}$ et les **segments audio** $\{A_l\}$. Chacun est pourvu d'un **embedding** v_k ou a_l . La **synergie inter-modale** $S_{AV}(V_p, A_r)$ se veut une combinaison de la **proximité temporelle** et de la **compatibilité** sémantique (par ex. un son d'aboielement concordant avec la présence d'un chien à l'écran). La **dynamique** DSL relie alors ces deux "blocs" (vidéo et audio) au moyen de **pondérations** $\omega_{V, A}$ adaptatives. Cette configuration permet au réseau d'**auto-organiser** le couplage audio-visuel, révélant les **sous-scènes** cohérentes et favorisant la **fusion** multimodale de manière distribuée et émergente, plutôt que via un pipeline figé.

8.8.2.2. Le SCN tisse des liens si la synchronie est forte (ex. on voit un chien aboyer, on entend un aboielement)

Dans un système **multimodal** combinant plusieurs **flux** (p. ex. un flux **visuel** et un flux **audio**), le **Synergistic Connection Network (SCN)** met en relation des **entités** issues de canaux différents, et la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ évalue la mesure selon laquelle ces entités s'**accordent** ou se **synchronisent**. Lorsqu'il existe une correspondance temporelle ou sémantique forte (tel un chien dans la vidéo et l'aboielement correspondant dans l'audio), le **DSL** (Deep Synergy Learning) consolide les pondérations ω reliant ces entités, ce qui produit in fine un **couplage** stable reflétant un **événement** audio-visuel cohérent.

A. Principes de la synchronie multimodale

Une **synchronie** audio-visuelle repose sur deux facteurs, la **coïncidence temporelle** et la **correspondance sémantique** ou causale. Si l'on considère \mathcal{E}_i une entité d'un flux **visuel** représentant un segment d'images montrant un chien ouvrant la gueule et \mathcal{E}_j une entité du flux **audio** correspondant à un segment sonore présentant un aboielement, on peut définir la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$ comme un mélange d'indicateurs. Cette synergie repose sur leur **chevauchement temporel** ainsi que sur leur **rapprochement sémantique**, un chien qui aboie correspondant parfaitement à un son d'aboielement. On peut penser à une **corrélation** ou un alignement temps-fréquence, ou à de l'**information mutuelle** capturant l'occurrence simultanée de l'événement visuel et du motif sonore.

B. Renforcement si la synchronie est forte

La **logique** du SCN (voir la section 2.2.2 sur la mise à jour) stipule que si $S(\mathcal{E}_i, \mathcal{E}_j)$ est **élevée**, la **pondération** $\omega_{i,j}$ reliant l'entité \mathcal{E}_i à l'entité \mathcal{E}_j augmente. De manière concrète, la règle s'écrit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)].$$

Si, par exemple, on observe un chien remuant la gueule dans le flux **vidéo** et qu'on entend simultanément un **aboielement** dans le flux **audio**, la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ apparaît **maximale**, menant à une hausse notable de $\omega_{i,j}$. Le SCN se "rappelle" alors que l'entité visuelle "chien aboyant" coïncide avec l'entité audio "aboielement".

Cette **sélection** adaptative des liaisons aboutit, sur le long terme, à un réseau où seuls les couplages réellement significatifs survivent ou s'amplifient — autrement dit, on ne retient que les **paires** dont la synchronie S est jugée forte.

C. Exemples concrets de tissage de liens

Lorsqu'un **chien** aboie, on se trouve en présence d'un motif **visuel** (le chien, la posture du museau ouvert) et d'un motif **sonore** (fréquences de l'abolement). La synergie $S(\text{chienVis}, \text{aboieAud})$ atteint un niveau élevé. Dans la matrice de pondérations $\{\omega_{i,j}\}$, on renforce donc le lien $\omega_{\text{chienVis}, \text{aboieAud}}$. De la même façon, si on voit un **chat** miauler et que l'audio capture un "miaou", le SCN consolide $\omega_{\text{chatVis}, \text{miaouAud}}$. Les paires (chienVis, miaulementAud) n'ont pas de synchronie, d'où un score bas et une pondération réduite.

Cette **stratégie** ne se cantonne pas aux animaux. Toute corrélation temporelle entre un **objet** ou une **action** visible et un **son** associé (par exemple, une batte frappant une balle + un bruit de frappe) aboutit à un renforcement dans le SCN. Plus la cohérence temporelle est strictement localisée, plus le score de **similarité** ou de **corrélation** calculé dans la fonction S est important.

D. Interprétation mathématique de la fusion visuo-auditive

Dans un cadre **audio-visuel**, la synergie S peut être formalisée de multiples façons. On peut envisager :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \rho(\mathbf{x}_i, \mathbf{x}_j),$$

où \mathbf{x}_i est un **embedding** pour l'entité visuelle \mathcal{E}_i et \mathbf{x}_j un **embedding** pour l'entité audio \mathcal{E}_j . Si la corrélation $\rho \approx 1$, la mise à jour de ω avantage fortement ce couplage (voir la référence 2.2.2.1), stabilisant un **lien** entre flux visuel et flux sonore. À un niveau plus **global**, un ensemble d'entités image-son se regroupe en **cluster** multimodal quand les pondérations ω s'avèrent élevées, formant un sous-réseau cohérent (voir la référence 8.8.2.3 à venir).

E. Avantages pour l'identification d'événements multimodaux

Un tel SCN offre la capacité de repérer des **événements** où l'on identifie à la fois le composant **visuel** et son pendant **sonore**. L'**intégration** des signaux devient alors plus robuste. Si le visuel est trop ambigu, l'audio clarifie la scène, et inversement. Les entités "chienVis" et "aboieAud" se retrouvent couplées, illustrant la **causalité** (le chien est la source de l'abolement). Le SCN, dans sa version DSL, n'exige pas de supervision. La simple récurrence temporelle et la similarité calculée suffisent à rendre $\omega_{\text{chienVis}, \text{aboieAud}}$ prépondérant par rapport à d'autres paires moins synchronisées.

Conclusion

Le SCN tisse donc des **liens** entre différents **flux** multimodaux lorsque la **synchronie** détectée (temporelle, sémantique) est forte. L'exemple le plus illustratif est celui d'un chien aboyant. Si la partie visuelle (chien ouvrant la gueule) et la partie sonore (fréquences d'abolement) se **superposent** dans le temps et la scène, la **synergie** calculée par la fonction S atteint un niveau élevé, ce qui se traduit, au sein de la mise à jour DSL, par l'augmentation de la pondération ω . Ce mécanisme aboutit, après convergence, à un **réseau** où les entités réellement couplées (vues-sons) s'**auto-organisent** en structures cohérentes, rendant ainsi l'identification d'événements audio-visuels plus solide et plus explicite.

8.8.2.3. Exemples : vidéo de scènes diverses, regroupement par type de son

Dans un **SCN** (Synergistic Connection Network) conçu pour la **fusion multimodale**, les flux **audio** et **vidéo** se décomposent en **segments** que l'on associe à des **entités** distinctes. Le **DSL** (Deep Synergy Learning) applique son principe d'**auto-organisation** aux pondérations ω reliant ces entités, de sorte à **faire émerger** des **macro-clusters** cohérents alliant la dimension visuelle (types de scènes) et la dimension sonore (types de sons). La présente section (8.8.2.3) illustre ce processus par l'exemple d'une **vidéo** complexe contenant diverses **scènes** et d'un **flux sonore** parallèlement enregistré, permettant de regrouper automatiquement "scènes de forêt + sons d'oiseaux", "scènes urbaines + bruits de circulation", etc.

A. Segmentation et Représentation des Scènes Vidéo

Un flux **vidéo** est généralement découpé en **séquences** ou **scènes** $\{\mathcal{V}_1, \dots, \mathcal{V}_{n_v}\}$. Chaque segment \mathcal{V}_i est converti en un **embedding** $\mathbf{v}_i \in \mathbb{R}^d$, que l'on peut obtenir par un **réseau** convolutif ou un **autoencodeur** appliqué à des frames ou à un agrégat de frames. On peut ainsi caractériser des contenus variés, une scène de plage, une scène urbaine, une scène

de forêt, etc. D'un point de vue purement **intra-modal**, la fonction de **synergie** $S(\mathcal{V}_i, \mathcal{V}_k)$ reflète la proximité des embeddings \mathbf{v}_i et \mathbf{v}_k (voir la section 8.8.2.1), ce qui permet de détecter et de regrouper les scènes visuellement proches, par exemple "paysages de montagne" ou "plans en intérieur".

B. Segmentation et Représentation des Segments Audio

Le flux **audio** se découpe pareillement en **segments** $\{\mathcal{A}_1, \dots, \mathcal{A}_{n_a}\}$. Chaque segment \mathcal{A}_j est décrit par un vecteur \mathbf{a}_j , dérivé par exemple de **MFCC** (coefficients cepstraux), de **spectrogrammes** ou de **réseaux** neuronaux spécifiques (embeddings audio). De la même manière, la **synergie** $S(\mathcal{A}_j, \mathcal{A}_m)$ indique la similarité entre deux segments sonores, révélant des types de sons comparables comme les bruits de trafic, les chants d'oiseaux, les voix humaines ou la musique instrumentale. Cette évaluation intra-modale conduit à la formation de **clusters** de segments audio, chacun représentant un registre acoustique homogène.

C. Couplage Audio–Vidéo et Mise en Œuvre SCN

Au-delà de l'**analyse** séparée (vidéo vs. audio), l'intérêt d'un **SCN** multimodal réside dans la capacité à **relier** un segment vidéo \mathcal{V}_i à un segment audio \mathcal{A}_j par une pondération $\omega_{i,j}$. Cette pondération évolue sous l'effet de la **synergie** $S(\mathcal{V}_i, \mathcal{A}_j)$. La fonction S capture à la fois la **concordance** temporelle (les deux segments se chevauchent dans le temps) et la **correspondance** sémantique (le contenu sonore correspond-il à la scène ?). Une formulation possible :

$$S_{AV}(\mathcal{V}_i, \mathcal{A}_j) = \alpha \delta_{\text{time}}(\mathcal{V}_i, \mathcal{A}_j) + \beta \text{Sim}(\mathbf{v}_i, \mathbf{a}_j),$$

où δ_{time} identifie si les segments se superposent (ou se succèdent) d'un point de vue chronologique, et $\text{Sim}(\mathbf{v}_i, \mathbf{a}_j)$ mesure la cohérence sémantique (par ex. spectrogramme d'abolement associé à la présence d'un chien dans la vidéo). Dans le **SCN**, la pondération $\omega_{i,j}(t)$ se met à jour suivant la **règle DSL** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S_{AV}(\mathcal{V}_i, \mathcal{A}_j) - \tau \omega_{i,j}(t)].$$

Lorsque la synergie est forte (ex. on voit des **oiseaux** dans la vidéo et on entend des **chants d'oiseaux** au même moment), $\omega_{i,j}$ grimpe, entraînant la création d'un **lien** solide entre la scène visuelle correspondante et le son idoine.

D. Émergence de Macro-Clusters Multimodaux

Les pondérations ω ainsi établies ne relient pas seulement "scène vidéo–scène audio", mais exploitent également les liens **intra-modal** (vidéo–vidéo, audio–audio). On obtient in fine un **réseau** structuré où les segments vidéo **similaires** forment des **clusters** internes, tels que des groupes représentant des "scènes de forêt" ou des "scènes urbaines". De la même manière, les segments audio **similaires** se regroupent en catégories distinctes, comme un cluster dédié aux "voix humaines" ou aux "moteurs de voiture". Enfin, les liaisons **cross-modales**, reliant des segments visuels et sonores synchrones ou cohérents, engendrent des **macro-clusters** multimodaux intégrant, par exemple, des "scènes de forêt associées aux sons d'oiseaux" ou des "séquences urbaines combinées aux bruits de circulation".

Au sens **mathématique**, ce phénomène signifie que le **DSL** repère les configurations de pondérations $\{\omega_{i,j}\}$ minimisant l'énergie globale, de telle sorte que de grandes ensembles d'entités (audio et vidéo) partagent de fortes liaisons internes s'ils relèvent d'un **contexte** homogène.

E. Exemple Concret de Fusions de Flux

Imaginons qu'un système traite une **longue vidéo** filmée dans plusieurs lieux, avec une scène en **forêt** où apparaissent des arbres et des animaux, une scène en **ville** montrant des immeubles et du trafic, ainsi que diverses **transitions** entre ces environnements. Parallèlement, l'audio comporte des **segments** de chants d'oiseaux, des **bruits** de voiture, et parfois de la **musique**. Lors de l'**auto-organisation**, les segments vidéo représentant une forêt se renforcent mutuellement en raison de similarités en termes de couleurs et de textures, tandis que les segments audio contenant des chants d'oiseaux se regroupent entre eux. Les liens **cross-modaux** deviennent plus marqués sur la période correspondante, établissant une association entre la forêt et le chant d'oiseaux, ce qui aboutit à un **macro-cluster** intégrant ces deux types d'entités. De la même manière, les segments vidéo représentant une ville entrent en synergie avec les segments audio caractérisés par des klaxons ou des bruits de foule.

On voit alors apparaître plusieurs **macro-clusters**, tels qu'un bloc "nature" et un bloc "urbain". Une **musique** de fond, si elle se propage dans différentes scènes, peut se lier un peu à plusieurs segments vidéo, sans former un cluster trop spécifique — ce qui dépendra de la force de la synergie calculée.

Conclusion

Les **exemples** de multiples **scènes vidéo** et de **regroupement** par type de **sons** illustrent la philosophie multimodale du SCN. Les entités $\{\mathcal{V}_i\}$ (segments vidéo) et $\{\mathcal{A}_j\}$ (segments audio) se **cohérent** et se **diférencient** simultanément, sous l'effet de la **synergie** S évaluée tant en intra-modal qu'en cross-modal. Les **macro-clusters** émergent alors comme de grandes communautés "son + scène" (forêt/oiseaux, route/voitures, intérieur/voix), témoignant du rôle de l'**auto-organisation** dans la découverte de **situations** ou de **contextes** communs. Sur le plan **mathématique**, cette fusion s'exprime par la **réduction** de l'énergie globale et la **stabilisation** des pondérations $\{\omega_{i,j}\}$ qui relient effectivement, de manière persistante, les flux visuels et sonores participant au même **événement**.

8.8.3. IA Symbolique–Sub-Symbolique en Multimodal

Dans une approche **multimodale**, un **SCN** (Synergistic Connection Network) est déjà appelé à gérer diverses sources (image, audio, texte, etc.). Lorsqu'on ajoute en plus des **règles logiques** (dimension **symbolique**), la complexité du réseau augmente d'un cran. Au-delà des simples embeddings sub-symboliques (vision/audio), on introduit également un niveau **logique** (ex. "if meowing then cat"). Le **DSL** (Deep Synergy Learning) doit alors organiser un **triple couplage** entre l'**image**, représentée par des vecteurs CNN ou des features extraites d'une scène, l'**audio**, analysé sous forme de spectrogrammes ou d'embeddings acoustiques, et les **règles logiques**, qui permettent d'établir des connexions sémantiques, par exemple en associant le concept "cat" à la détection d'un "meow" ou en validant une règle comme "milk if dairy-product?".

8.8.3.1. Lorsqu'on ajoute des règles logiques (ex. "if meowing then cat" + embeddings), le SCN gère un triple couplage image–audio–règle

Le **DSL** (Deep Synergy Learning), appliqué à un **SCN** (Synergistic Connection Network), peut recevoir comme **entrées** non seulement des **entités** d'origine sub-symbolique (telles que des embeddings d'images ou d'audio) mais également des **règles** de nature **logique** (entités symboliques). L'exemple d'une règle "if meowing then cat" illustre comment, dans une configuration **multimodale**, le **SCN** peut exploiter simultanément l'**information** visuelle (un chat détecté dans l'image), l'**information** sonore (un miaulement capturé par l'audio) et la **règle** logique dictant un lien causal ("si j'entends meow, alors il y a un chat"). La présente section (8.8.3.1) décrit :

- La **représentation** symbolique d'une règle,
- Le **triple couplage** entre image, audio et logique,
- L'**émergence** d'un concept multimodal plus solide ("cat") grâce à cette triangulation.

A. Représentation Symbolique et Synergie

Bien qu'un **SCN** traite essentiellement des **embeddings** sub-symboliques (par ex. des vecteurs issus de CNN ou des spectrogrammes neuronaux), on peut y **inclure** des **entités** symboliques. Une **règle** telle que "if meowing then cat" se voit alors modélisée par un **nœud** \mathcal{E}_{rule} qui possède lui aussi un **vecteur** ou un **descripteur** dans un espace symbolique ou sémantique. De façon formelle, on attribue à cette entité \mathcal{E}_{rule} un "embedding logique" $\mathbf{r} \in \mathbb{R}^d$, ou tout au moins un identifiant que l'on peut mettre en correspondance avec des motifs d'**audio** ou des **objets** visuels.

La **synergie** $S(\mathcal{E}_{logic}, \mathcal{E}_{audio})$ ou $S(\mathcal{E}_{logic}, \mathcal{E}_{image})$ reflète le degré de correspondance entre la **règle** symbolique et le **signal** perçu. Par exemple, si la règle "if meowing then cat" s'applique, et qu'on détecte un **miaulement** sur le flux audio, la **compatibilité** est forte. De même, si l'image suggère clairement la forme d'un chat, la **règle** peut valider cette hypothèse via un score positif. Dans le **SCN**, on met à jour $\omega_{rule, audio}$ ou $\omega_{rule, image}$ conformément à la règle de mise à jour (voir la section 2.2.2).

B. Couplage Triple : Image–Audio–Règle

Lorsqu'il existe trois entités \mathcal{E}_{img} , \mathcal{E}_{aud} , $\mathcal{E}_{\text{rule}}$, chacune peut entretenir une **synergie** avec les deux autres. La mesure $S(\text{img}, \text{aud})$ évalue l'**accord** entre l'image, par exemple un chat visible à l'écran, et le son, comme un **miaulement** correspondant. La synergie $S(\text{img}, \text{rule})$ détermine dans quelle mesure le **visuel** est en phase avec le concept "chat", sous-jacent à une règle du type "if meowing then cat". Enfin, la relation $S(\text{aud}, \text{rule})$ valide si le segment sonore détecté correspond à la condition évoquée dans la règle, confirmant ainsi une **cohérence tri-modale**.

En un sens, on aboutit à un **triangle** de pondérations dans le SCN avec $\omega_{\text{img}, \text{aud}}$, $\omega_{\text{img}, \text{rule}}$, $\omega_{\text{aud}, \text{rule}}$. Si, par exemple, on constate simultanément un miaulement sur l'audio, la présence d'un chat sur la vidéo et la règle "if meowing then cat" dans le **réseau**, chaque lien se trouve **renforcé**, ce qui accroît la **convergence** vers un **macro-nœud** ou **cluster** incarnant la "situation" où un chat produit un miaulement conformément à la règle logique.

C. Stabilité des Concepts Multimodaux

En **intégrant** un niveau symbolique (les règles) dans le **SCN**, on permet à la **logique** de s'ajouter aux indicateurs sub-symboliques (image, audio). On peut formuler mathématiquement la synergie globale comme :

$$S_{\text{global}}(\text{img}, \text{aud}, \text{rule}) = \alpha S(\text{img}, \text{aud}) + \beta S(\text{img}, \text{rule}) + \gamma S(\text{aud}, \text{rule}).$$

Lorsque la somme $\alpha S(\text{img}, \text{aud}) + \beta S(\text{img}, \text{rule}) + \gamma S(\text{aud}, \text{rule})$ est élevée, le **SCN** renforce les liens $\{\omega_{\text{img}, \text{aud}}, \omega_{\text{img}, \text{rule}}, \omega_{\text{aud}, \text{rule}}\}$, ce qui aboutit à un **concept** stable (ex. le "cluster" correspondant à un chat). On dit alors que la **cohérence** du concept "chat" est alimentée par la fois par l'image (l'apparence du chat), l'audio (le miaulement), et la **règle** "if meowing then cat".

Cette **triangulation** favorise une convergence plus rapide et plus sûre qu'un couplage seulement audio–image, car la **règle** logique agit comme un **tuteur** symbolique, clarifiant la relation "un miaulement implique un chat". Lorsque ce triple couplage est validé, le DSL **stabilise** fortement le noeud "cat" et les pondérations afférentes, constituant ainsi un **cluster** multimodal conceptuellement robuste.

D. Avantage : l'Auto-Organisation produit des Concepts plus Stables

La présence, dans le **SCN**, d'un **niveau** symbolique n'est pas nécessairement complexe sur le plan numérique. On peut se contenter d'allouer un **embedding** \mathbf{r}_k pour chaque **règle** $\mathcal{E}_{\text{rule}^k}$, et d'un mécanisme de **match** pour lister les conditions (miaulement, aboiement, etc.) et leurs conséquences (chat, chien). L'**auto-organisation** du DSL fait le reste. On retiendra les points saillants :

- **Convergence plus facile** : la connaissance symbolique "if meowing then cat" complète les indices sub-symboliques (on voit un chat, on entend un "meow"). Les liaisons ω se renforcent plus globalement, permettant au SCN de former un **concept** "chat" de manière plus évidente.
- **Expliquabilité** : une fois convergé, on peut repérer le **rôle** de la règle dans la formation du cluster. L'analyse du SCN indique que les liens image–règle, audio–règle, image–audio se sont mutuellement validés.
- **Extension** : on peut ajouter toute une série de règles analogues ("if barking then dog", "if crowing then rooster"), ou des énoncés plus complexes décrivant la logique de la scène (par ex. "if flapping wings then bird").

Conclusion

Lorsqu'on **ajoute** des **règles** logiques — du type "if meowing then cat" — à un **SCN** qui traite déjà **image** (apparence d'un chat) et **audio** (miaulement), on obtient un **triple couplage** :

- **Sub-symbolique** : embeddings visuels et sonores,
- **Symbolique** : règles logiques énonçant des relations causales,
- **SCN** : mise à jour des liens ω reliant image–audio, image–règle, audio–règle.

L'**auto-organisation** du DSL tire parti de ces trois **flux** pour faire émerger un **cluster** ou un **macro-nœud** plus stable, incarnant le concept “chat” de manière explicite (règle “if meowing then cat”) et implicite (image–audio). Outre l’amélioration de la **robustesse** et de la **rapidité** de convergence, cette intégration symbolique–subsymbolique facilite une **explicabilité** accrue, permettant de justifier qu’un certain cluster est étiqueté “chat” parce qu’il réunit simultanément un visuel félin, un son de miaulement, et une règle logique validant la correspondance.

8.8.3.2. Avantage : l’auto-organisation fait émerger des concepts multimodaux plus stables

La **fusion multimodale** dans un **SCN** (Synergistic Connection Network) piloté par un **DSL** (Deep Synergy Learning) présente un avantage déterminant en permettant de **solidifier** l’émergence de **concept**s ou *macro-nœuds* regroupant plusieurs entités grâce au concours simultané de différents **canaux** visuels, audio, textuels, etc. Lorsque plusieurs modalités corroborent un même groupement, la **synergie** s’en trouve renforcée et la **stabilité** d’un cluster multimodal augmente. Le présent exposé (8.8.3.2) souligne les mécanismes mathématiques et pratiques qui rendent les concepts plus robustes lorsqu’ils reposent sur des indices diversifiés.

A. Rappel du Principe d’Auto-organisation dans un Cadre Multimodal

Un **SCN** multimodal comprend des entités \mathcal{E}_i issues de différentes **sources** (images, sons, textes, éventuellement règles symboliques). La **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ mesure leur **compatibilité** ou **complémentarité**. Dans un cas simplifié, on pourrait définir :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \alpha S_{\text{vis}}(i, j) + \beta S_{\text{text}}(i, j) + \gamma S_{\text{aud}}(i, j).$$

La **pondération** $\omega_{i,j}$ entre \mathcal{E}_i et \mathcal{E}_j suit la mise à jour DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)].$$

Ce mécanisme **auto-organisé** renforce les **paires** (ou les ensembles) d’entités qui affichent une **synergie** élevée, ce qui aboutit à des **clusters** de nœuds ayant un score important. Dans un cadre **multimodal**, si plusieurs canaux confirment la même **association** (ex. un chat visible et un “meow” audible), la **somme** des composantes de $S(i, j)$ favorise une pondération $\omega_{i,j}$ d’autant plus grande.

B. Stabilité Accrue des Concepts Multimodaux

Lorsqu’un **concept** émerge de la conjonction de plusieurs canaux, ce concept se montre **plus stable** qu’un concept ne s’appuyant que sur un seul flux. Supposons un **cluster** combinant $\mathcal{E}_{\text{image}}$ (un embedding visuel), $\mathcal{E}_{\text{audio}}$ (un extrait sonore) et $\mathcal{E}_{\text{text}}$ (un fragment linguistique). Même si l’un des canaux est **perturbé** (bruit sur l’audio), les deux autres canaux (image et texte) maintiennent la cohésion. D’un point de vue **mathématique**, la pondération ω reste soutenue par la synergie cumulée $\alpha S_{\text{vis}} + \beta S_{\text{text}} + \dots$. Cela évite qu’un **bruit** localisé (par ex. un enregistrement audio pollué) ne dissolve entièrement le cluster.

La **mise à jour** $\omega(t+1) = \omega(t) + \eta [S - \tau \omega]$ bénéficie particulièrement de la **somme** des synergies issues des canaux. Si l’entité \mathcal{E}_i correspond simultanément à des indices visuels et textuels cohérents, la **valeur** de $S(i, j)$ (et donc la force du gradient) est plus grande que dans un cadre *mono-modal*. Ainsi, la pondération $\omega_{i,j}$ atteint plus rapidement et plus fermement un **point** d’équilibre élevé. En conséquence, la **configuration** finale (graphe de pondérations ω) présente des “puits” d’énergie plus profonds, rendant les clusters multimodaux plus **difficiles** à déstabiliser.

Sur un plan **combinatoire**, la présence de multiples canaux pourrait suggérer une complexification, mais en réalité, les **ambiguïtés** se réduisent si un concept doit **convaincre** plusieurs modalités en même temps. Un “faux alignement” qui semblerait plausible en vision peut être **démenti** par l’audio ou le texte. De ce fait, seul un alignement **multi-canal** récoltant l’approbation de tous les canaux parvient à se stabiliser. Cette intégration agit donc comme un **filtre** naturel pour éliminer les configurations ambiguës, entraînant un **concept** plus net (ex. le “chien qui aboie” est soutenu par l’aspect canin et la tonalité d’aboïement, écartant un “chat” ou autre animal).

C. Impact Mathématique sur l'Émergence des Concepts

Dans la logique (chap. 7.2) où la descente d'énergie $\nabla J(\omega)$ oriente la mise à jour ω , la **partie** “gain” de J inclut le produit $\omega_{i,j} \times S(i,j)$ (qui s'additionne pour chaque couple). Lorsque la synergie $S(i,j)$ comporte plusieurs **composantes** issues de canaux distincts, la **somme** est d'autant plus élevée si tous les canaux **convergent**. Le **cluster** résultant dispose alors d'un **gain** plus important, le rendant énergétiquement plus **stable** et moins enclin à être perturbé par de petites fluctuations.

Les analyses de **dynamique** dans le SCN montrent que des **clusters** forts (pondérations élevées et mutuellement soutenues) correspondent à des **attracteurs** de la mise à jour $\omega(t+1) = \omega(t) + \dots$. Dans un cadre multimodal, un “concept” drainant la cohérence de plusieurs flux, s’impose comme un attracteur **plus** stable (ou de gradient plus prononcé). Cela se formalise par un **gradient** où la “force” s’appliquant en faveur d’un concept prend en compte la **somme** $\alpha S_{\text{vis}} + \beta S_{\text{text}} + \dots$, creusant un **minimum local** plus profond sur l'espace des pondérations.

D. Conclusion

L'**auto-organisation** d'un SCN dans un contexte **multimodal** renforce la solidité des **concepts** émergents. Les pondérations ω ne se fondent plus sur une seule source d'information, mais intègrent plusieurs canaux (visuel, audio, textuel, règles symboliques). Cette configuration présente des avantages clairs. Les **concepts** formés par la **convergence** de plusieurs modalités bénéficient d'une stabilité accrue, car ils sont moins sensibles au bruit dans une modalité particulière et se renforcent rapidement si toutes les modalités s'accordent. Les **ambiguïtés** se réduisent, car il est peu probable qu'un “faux” concept puisse simultanément tromper plusieurs flux. Sur le plan **mathématique**, les **attracteurs** dans l'espace des pondérations ω gagnent en profondeur, rendant la configuration finale plus **robuste** et moins susceptible de s'effondrer ou de dériver.

C'est ainsi que le **DSL** multimodal exploite des **contributions** diverses (image, son, texte, etc.) pour **faire émerger** des **macro-clusters** plus fiables, dont l'identification ou l'étiquetage se révèle **plus stable** que dans un schéma cloisonné et mono-modal.

8.9. Aspects Évolutifs et Temps Réel

Au sein d'un **DSL** (Deep Synergy Learning) multimodal, il est fréquent que les **données** (images, sons, texte) arrivent de façon **continue** et **évolutive** : on parle alors de **flux** (streams) plutôt que de lots statiques. Dans un tel contexte, le **SCN** (Synergistic Connection Network) doit s'adapter en **temps réel**, en absorbant progressivement de nouveaux segments vidéo ou audio, tout en conservant la structure synergique préexistante. Les sections 8.9.1 à 8.9.3 décrivent comment gérer et visualiser cette dynamique évolutive dans un cadre réellement **multimodal**.

8.9.1. Flux Multimodal en Continu

La **notion** de flux multimodal implique qu'au fil du temps t , on reçoit en **continu** des **extraits** (frames) vidéo, des **segments** audios et éventuellement des **sous-titres** ou des **annotations** textuelles. Le **SCN** se trouve alors confronté à la tâche de **fusionner** ou de **clusteriser** ces nouvelles entités dans la matrice $\{\omega_{i,j}\}$, sans redémarrer l'apprentissage depuis zéro. Les mathématiques du DSL (mise à jour itérative de ω) doivent donc être étendues pour accueillir l'**arrivée incrementale** de données, comme développé en Chap. 7 (sections 7.6 sur l'adaptation continue).

8.9.1.1. Scénarios : streaming vidéo + audio + sous-titres

Dans de nombreux systèmes **multimodaux**, il n'existe pas un lot unique de données à traiter hors ligne, mais plutôt un **flux continu** de contenus (par exemple, un **streaming** issu d'une conférence, d'un cours en ligne ou d'une vidéo en direct) où les canaux **visuels**, **sonores** et **textuels** parviennent simultanément. La présente section (8.9.1.1) envisage un scénario concret où un **flux** vidéo fourni en temps réel, accompagné d'un **flux** audio synchrone et de **sous-titres** ou annotations textuelles intermittentes. L'objectif du **SCN** (Synergistic Connection Network) est d'**intégrer** ces nouvelles entités au fur et à mesure de leur arrivée et de **maintenir** une auto-organisation cohérente permettant de faire émerger des **clusters** (macro-nœuds) multimodaux en temps réel.

A. Exemple de flux continu

Un **service** de streaming peut recevoir :

- Des **frames vidéo** successives $\{F_t\}$,
- Un **flux audio** $\{A_t\}$ enregistré en parallèle,
- Des **sous-titres** (ou tout autre texte) $\{T_t\}$ arrivant par blocs ou au fil de la transcription,
- Des **métadonnées** supplémentaires (type chapitrage, indicateurs d'interactivité, etc.).

Dans un **SCN** (voir la section 2.2.1 sur la définition d'entités), chaque **entité** \mathcal{E} correspond à un segment visuel, une portion audio ou un bloc de texte. L'ensemble de ces entités $\{\mathcal{E}_n\}_{n \in \mathbb{N}}$ grandit au fil du temps. À chaque **pas**, on insère dans le SCN la nouvelle **entité** provenant du flux vidéo $\mathcal{E}_{(\text{vid},t)}$, celle issue du flux audio $\mathcal{E}_{(\text{aud},t)}$ et éventuellement un **chunk** textuel $\mathcal{E}_{(\text{txt},u)}$.

La question centrale est de savoir comment calculer les **pondérations** ω reliant ces nouvelles entités aux entités déjà présentes, et d'éviter que le **coût** de mise à jour ne devienne prohibitif à mesure que le flux s'allonge.

B. Arrivée Incrémentale et Mise à Jour

À chaque itération, au temps τ (pour ne pas confondre avec le τ de décroissance), on reçoit une entité \mathcal{E}_{new} . Dans un **SCN**, on doit alors **connecter** \mathcal{E}_{new} à un sous-ensemble d'entités déjà existantes. On calcule la **synergie** :

$$S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) = f(\mathbf{x}_{\text{new}}, \mathbf{x}_j),$$

où \mathbf{x}_{new} et \mathbf{x}_j sont les embeddings ou les signatures associées à \mathcal{E}_{new} et \mathcal{E}_j . Concrètement, si la nouvelle entité est un frame vidéo, on compare son **embedding visuel** (réseau CNN, par ex.) avec ceux des frames vidéo passés et avec les signatures audio/texte temporellement proches. Cela entraîne la mise à jour :

$$\omega_{\text{new},j}(t+1) = \omega_{\text{new},j}(t) + \eta [S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) - \tau \omega_{\text{new},j}(t)].$$

Dans un streaming de longue durée, le nombre d'entités $\{\mathcal{E}_j\}$ peut croître de façon considérable, rendant le coût $O(n)$ (ou $O(n^2)$ avec la mise à jour croisée) trop grand. Plusieurs **stratégies** limitent l'explosion :

- **Fenêtre glissante** : on ne compare la nouvelle entité qu'aux entités proches dans le temps $\{j \mid |t(\text{new}) - t(j)| \leq \Delta\}$.
- **k-NN** : on recherche les k entités les plus proches en embedding, plutôt que de tout comparer.
- **Parcellisation** : on gère des sous-réseaux ou on agrège les entités plus anciennes en clusters, évitant un survol exhaustif de tous les noeuds (voir chap. 7.6).

C. Problème de Synchronisation

Dans le flux **audio–vidéo**, un **frame** $\mathcal{E}_{(\text{vid},t)}$ correspond au temps t , tandis que le segment audio $\mathcal{E}_{(\text{aud},t')}$ peut se situer un demi-frame plus tard si la synchronisation n'est pas parfaite. Les **sous-titres** $\mathcal{E}_{(\text{txt},u)}$ arrivent souvent par blocs couvrant plusieurs secondes, avec un léger décalage. Le **SCN** doit donc **adapter** la définition de la synergie pour accepter un battement temporel :

$$S(\mathcal{E}_{(\text{vid},t)}, \mathcal{E}_{(\text{aud},t')}) = g(\mathbf{x}_{\text{vid},t}, \mathbf{x}_{\text{aud},t'}) \delta_{\text{time}}(t, t'),$$

où $\delta_{\text{time}}(t, t')$ rend la synergie presque nulle si $|t - t'|$ est trop grand, ou la pénalise proportionnellement à l'écart. On peut également restreindre $\omega_{\text{vid,aud}}$ à des paires (t, t') jugées proches dans la chronologie.

Lorsque des **chunks** textuels couvrent plusieurs secondes (ex. 2 à 5 s), on associe chaque chunk $\mathcal{E}_{(\text{txt},u)}$ à une plage $[t_0, t_1]$. On évalue la synergie $S(\text{vid,txt})$ ou $S(\text{aud,txt})$ en considérant si le segment vidéo (ou audio) intersecte temporellement la fenêtre $[t_0, t_1]$. À nouveau, cela permet de ne pas comparer tout à tout, mais seulement ce qui se chevauche.

D. Implications sur la Synergie Multimodale

Grâce à la **gestion** continue des entités, le **SCN** perçoit au **fil du temps** la corrélation entre frames vidéo et segments audio ou textuels coïncidant dans la chronologie. Les **pondérations** ω se renforcent lorsqu'une concordance survient (p. ex. une personne s'exprimant à l'écran correspond au segment audio de la même voix). L'auto-organisation fonctionne ainsi **en temps réel**, au lieu d'être un traitement batch a posteriori.

La **fréquence** d'arrivée des frames (p. ex. 30 par seconde) peut être très élevée, rendant la mise à jour ω coûteuse s'il faut sonder la synergie avec toutes les entités passées. Des méthodes d'approximation (voisinage, fenêtre glissante, chap. 7.6) atténuent le problème. Sur le plan mathématique, on suppose souvent un **incrément** Δt fixant la durée de validité d'une comparaison, ou on projette l'entité $\mathbf{x}_{\text{vid},t}$ dans un cluster intermédiaire pour limiter la granularité.

Conclusion

En **streaming** vidéo + audio + sous-titres, le **SCN** s'adapte à l'**arrivée incrémentale** d'entités. Chaque **frame** vidéo, chaque **segment** audio, chaque **bloc** de texte est inséré à mesure qu'il apparaît, avec un calcul local de la **synergie** $S(\text{new},j)$ restreint à un **voisinage** temporel ou sémantique. Le SCN met alors à jour $\omega_{\text{new},j}$ selon la règle DSL (voir la référence 2.2.2.1). Cette **dynamique en ligne** résout le **problème de la synchronisation** (frames, audio, sous-titres ne sont pas strictement alignés) en autorisant un battement temporel $\delta_{\text{time}}(t, t')$.

Sur le plan **mathématique**, la croissance progressive du réseau requiert des **heuristiques** pour prévenir un surcoût en $O(n^2)$. Les notions de **voisinage** (fenêtre temporelle) et de **k-NN** aident à maintenir la **scalabilité**. En fin de compte, cette approche **en flux** et **multi-canal** permet l'**auto-organisation** des segments multimédias au rythme de leur arrivée, créant en temps réel des **clusters** de contenu unifiant l'image, le son, et le texte (chap. 8.9.2 approfondira la stabilisation et le suivi de ces clusters).

8.9.1.2. Mise à jour incrémentale (Chap. 7.6), insertion progressive de frames, segments audios, etc.

Dans un contexte **multimodal** (vision, audio, texte) où les données **arrivent en continu**, un **SCN** (Synergistic Connection Network) basé sur un **DSL** (Deep Synergy Learning) se doit de gérer l'**insertion** de nouvelles **entités** (frames vidéo, segments audio, sous-titres, etc.) sans tout reprendre à zéro. La section 7.6 souligne précisément les mécanismes d'**insertion progressive** et de **mise à jour incrémentale** des pondérations ω . Appliqués à un scénario de **streaming**, ces méthodes permettent de préserver la structure **auto-organisée** du réseau, tout en intégrant les **nouveaux** flux à la volée.

A. Principes de la mise à jour incrémentale

Lorsque l'on reçoit une **nouvelle** entité \mathcal{E}_{n+1} , par exemple un frame vidéo $\mathcal{E}_{(\text{vid}, \tau)}$ ou un chunk audio $\mathcal{E}_{(\text{aud}, \tau')}$, le **SCN** s'agrandit. La matrice ω passe de taille $n \times n$ à $(n + 1) \times (n + 1)$. Pour tout noeud \mathcal{E}_j déjà existant, on ajoute deux nouvelles lignes/colonnes $\omega_{(n+1),j}, \omega_{j,(n+1)}$, que l'on peut initialiser à zéro ou à une valeur aléatoire de faible amplitude.

D'un point de vue **mathématique**, l'opération ne nécessite pas de **recalcul** exhaustif sur l'ensemble des paires passées. Les pondérations $\omega_{i,j}$ reliant entre elles des entités déjà présentes (indices 1 ... n) demeurent **inchangées**. On se borne à évaluer la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ pour un **voisinage** restreint (ou complet, si la charge de calcul le permet). On met alors à jour :

$$\omega_{(n+1),j}(t + 1) = \omega_{(n+1),j}(t) + \eta [S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau \omega_{(n+1),j}(t)].$$

Cette formule reflète la **descente locale** (ou l'auto-organisation) évoquée au chapitre 7.2, mais appliquée uniquement au **nouveau** noeud \mathcal{E}_{n+1} et aux noeuds de son voisinage $\{j\}$.

Pour améliorer l'**efficacité**, la comparaison $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ est restreinte à un **ensemble** $N(n + 1) \subseteq \{1, \dots, n\}$ de taille contrôlée, comprenant par exemple les k plus proches voisins ou les entités partageant un intervalle temporel similaire. Par ailleurs, les **pondérations** $\omega_{i,j}$ entre les noeuds déjà existants ($i, j \leq n$) restent **inchangées**, ou bien elles sont mises à jour via la routine standard, sans nécessiter un recalcul complet du réseau.

Le **cluster** global déjà formé se **stabilise** localement, tandis que le nouvel arrivant "trouve sa place" en fonction de la **synergie** calculée avec son voisinage.

Cette approche incrémentale assure que la **topologie** et la **répartition** des macro-clusters déjà formés dans le **SCN** ne soient pas totalement **réinitialisées** à l'arrivée d'un nouvel élément. Seule la **région** du graphe environnant le noeud \mathcal{E}_{n+1} se réadapte, permettant un **apprentissage continu** où la structure globale se **déploie** graduellement sur l'axe du temps (voir chap. 7.6).

B. Application à la Fusion Audio-Visuelle

Dans un **stream** combinant image et son, chaque **frame** vidéo (30 images/s) et chaque **bloc** audio (échantillonné différemment) arrive comme une **entité** \mathcal{E}_{n+1} . À son arrivée, on calcule la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ avec, par exemple, les entités \mathcal{E}_j de la même tranche temporelle. Si un segment audio $\mathcal{E}_{n+1}^{(\text{aud})}$ ressemble fortement à un autre (profil spectral identique) ou coïncide temporellement avec des frames vidéo pertinents, $\omega_{(n+1),j}$ se voit **renforcé**. Sinon, la pondération demeure faible.

Outre l'intra-modal (audio-audio, vidéo-vidéo), on peut aussi croiser $\mathcal{E}_{n+1}^{(\text{aud})}$ avec des frames vidéo $\mathcal{E}_k^{(\text{vid})}$ si \mathcal{E}_k tombe dans la même fenêtre temporelle. Le **SCN** (qui gère la **synergie** cross-modale, chap. 8.8) accueille ainsi une **connexion** éventuelle $\omega_{(n+1),k}$ plus élevée si un aboiement sonore correspond à l'apparition d'un chien à l'écran.

C. Mémoire et Approche Incrémentale

Quand on ajoute la nouvelle entité \mathcal{E}_{n+1} , on inscrit $\omega_{(n+1),j}$ dans la matrice, avec initialisation $\omega_{(n+1),j} \approx 0$. Ensuite, au moment $t \rightarrow t + 1$,

$$\omega_{(n+1),j}(t + 1) = \omega_{(n+1),j}(t) + \eta [S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau \omega_{(n+1),j}(t)].$$

Cette mise à jour localisée assure que la **structure** déjà présente ne subisse pas de re-calculation exhaustif.

Avantages

Le **coût** de calcul est réduit, car seul le **voisinage** immédiat d'une entité (quelques douzaines ou centaines d'éléments) est sondé, plutôt que l'ensemble du **SCN**. La **continuité** est assurée puisque les liens existants $\omega_{i,j}$ pour $i,j \leq n$ ne sont pas supprimés, évitant ainsi de **déstabiliser** les clusters déjà établis. L'**apprentissage** reste **continu**, le SCN s'agrandissant **progressivement** et chaque nouvelle insertion venant s'imbriquer naturellement dans la structure.

Risques

Si les données entrantes (frames, audio, texte) affluent à un rythme très élevé, le nombre d'entités \mathcal{E}_k peut rapidement devenir **ingérable**. Il est alors nécessaire de recourir à des techniques de **fusion** ou de **compression**, comme le concept de **micro-cluster** présenté en **chapitre 7.6.3**, afin de condenser des noeuds anciens ou de désactiver certaines entités trop vieilles. Sans ces mécanismes, la matrice ω risque de **croître indéfiniment**, rendant le SCN **difficilement exploitable**.

D. Observations Mathématiques

La **mise à jour** incrémentale prônée par le chap. 7.6 évite un $O(n^2)$ complet. On se contente de $O(k)$ ou $O(\log n)$ comparaisons si l'on maintient une structure de voisinage (k plus proches, ou recherche de window temporelle). Sur un flux **multimodal**, c'est crucial pour la **scalabilité**.

Le **SCN** ne converge plus au sens classique (puisque de nouvelles entités arrivent sans cesse), mais tend vers un **régime** stationnaire ou quasi-stationnaire où les **clusters** se stabilisent localement. Chaque **nouveau** noeud se voit insérer dans le cluster le plus pertinent, tandis que les clusters anciens conservent leur cohésion sauf si la nouvelle entité introduit une perturbation suffisamment grande pour déclencher un réajustement local (ch. 7.4 ou 7.5 sur l'inhibition compétitive).

Conclusion

La **mise à jour incrémentale** décrite au chapitre 7.6 est fondamentale pour intégrer, **en flux**, de nouveaux frames vidéo, segments audio, sous-titres, etc. dans un **SCN multimodal**. Les **pondérations** ω s'ajustent localement, évitant tout recalcul global. Cela soutient une **auto-organisation** continue, où chaque entité nouvelle consolide ou perturbe des **clusters** existants sans détruire la structure acquise. Par ce biais, le **DSL** propose un schéma d'**apprentissage en ligne**, essentiel pour des systèmes de streaming ou de traitement progressif (p. ex. vidéos longues, flux temps réel) dans lesquels images, sons et textes s'accumulent au fil du temps.

8.9.2. Convergence et Stabilisation

Dans le **contexte multimodal**, une fois que le **SCN** (Synergistic Connection Network) a commencé à agréger les flux (vidéo, audio, texte) et à mettre à jour les pondérations $\omega_{i,j}$, il est nécessaire de définir **comment** et **quand** l'on considère qu'un **cluster** (réunissant plusieurs segments ou entités) se **stabilise**. Cette stabilisation reflète l'idée d'une **convergence** de la dynamique $\{\omega_{i,j}(t)\}$ vers une configuration cohérente, du point de vue **synergique**. Les flux vidéo, audio, et textuel convergent vers un "ensemble" qui se **renforce mutuellement**.

Dans la section 8.9.2, nous étudions plus précisément :

- Les **critères** indiquant qu'un segment multimodal (vidéo+audio+texte) forme un **cluster stable** (§8.9.2.1),
- Le **problème** d'alignement temporel, souvent épiqueux (§8.9.2.2), puisque l'information visuelle et audio n'est pas toujours parfaitement **synchrone**, et que le texte peut survenir en décalé.

8.9.2.1. Critères pour estimer qu'un segment vidéo+audio+texte forme un "cluster" stable

Dans un **SCN** (Synergistic Connection Network) appliqué à un **DSL** (Deep Synergy Learning) multimodal, plusieurs segments d'un flux vidéo (images ou frames), d'un flux audio (extraits sonores) et, le cas échéant, d'un flux textuel (sous-titres, annotations) peuvent s'assembler pour constituer un **cluster** commun. Le problème se pose alors de déterminer **quand** on considère que ce cluster est "abouti" ou "stabilisé", au sens où les pondérations ω reliant les entités (vidéo, audio, texte) atteignent un état relativement constant. Cette section (8.9.2.1) examine divers **critères** mathématiques et heuristiques pour estimer qu'un ensemble \mathcal{C} de segments multimodaux forme un **cluster** stable.

A. Vue mathématique : convergence des pondérations ω

Chaque **entité** \mathcal{E}_i du SCN représente un **fragment** issu d'un flux ou d'une modalité. Il peut s'agir d'un segment vidéo \mathcal{V}_a , d'un segment audio \mathcal{A}_b ou d'un sous-titre \mathcal{T}_c . La **pondération** $\omega_{i,j}$ connecte l'entité \mathcal{E}_i à l'entité \mathcal{E}_j . L'**auto-organisation** (chap. 2.2.2) met à jour ces pondérations ω en fonction de la **synergie** $S(i,j)$. Un **cluster** $\mathcal{C} \subseteq \{\mathcal{E}_1, \dots\}$ est un sous-ensemble d'entités (p. ex. $\{\mathcal{V}_a, \mathcal{A}_b, \mathcal{T}_c\}$) que le SCN finit par **regrouper** (fortes pondérations) et **isoler** du reste.

On considère qu'un cluster \mathcal{C} se **stabilise** lorsqu'au fil du temps t , les variations $|\omega_{i,j}(t+1) - \omega_{i,j}(t)|$ deviennent négligeables pour toutes les paires $(i,j) \in \mathcal{C}$. Une écriture formelle consiste à exiger :

$$|\omega_{i,j}(t+1) - \omega_{i,j}(t)| \leq \varepsilon \quad \text{pour } (i,j) \in \mathcal{C} \times \mathcal{C},$$

avec $\varepsilon > 0$ un petit seuil. Cet **arrêt** de l'évolution signale que la dynamique du DSL (règle $\omega_{i,j} \leftarrow \omega_{i,j} + \eta[S(i,j) - \tau \omega_{i,j}]$) ne modifie plus sensiblement liaisons internes au cluster \mathcal{C} .

B. Critères de cohésion interne

Au-delà de la **convergence** en dérivée (c.-à-d. les pondérations ne bougent plus), il est possible d'imposer un certain **niveau** de cohérence. Par exemple, on définit :

$$\Omega(\mathcal{C}) = \sum_{(i,j) \in \mathcal{C} \times \mathcal{C}} \omega_{i,j}.$$

Si $\Omega(\mathcal{C})$ dépasse un **seuil** θ , on considère que le sous-ensemble \mathcal{C} présente une synergie suffisante pour être qualifié de "cluster" stable (la somme des pondérations internes étant conséquente). Dans les cas multimodaux, \mathcal{C} comprend des entités de plusieurs flux (vidéo, audio, texte) dont les pondérations croisées ω sont élevées.

On peut également contrôler la **distribution** des liaisons internes au cluster. Une variance ou un étalement trop grand (certains $\omega_{i,j}$ élevés, d'autres très faibles) pourrait indiquer un cluster encore "en train" de se préciser. Si au contraire la répartition des $\omega_{i,j}$ internes est plutôt **homogène** (toutes fortes ou moyennes), on y voit un indice de **stabilité** et d'unité du groupe.

C. Dynamique multimodale

Lorsqu'un **segment** vidéo \mathcal{V}_a et un segment audio \mathcal{A}_b correspondent au même intervalle temporel (p. ex. la même personne qui parle), la **pondération** $\omega_{\mathcal{V}_a, \mathcal{A}_b}$ tend à augmenter (chap. 8.8.2). Si, en plus, un **texte** \mathcal{T}_c (sous-titre) se révèle lié (le locuteur prononce ce sous-titre au moment t), la **pondération** $\omega_{\mathcal{V}_a, \mathcal{T}_c}$ et $\omega_{\mathcal{A}_b, \mathcal{T}_c}$ augmentent également.

On aboutit à un triplet $(\mathcal{V}_a, \mathcal{A}_b, \mathcal{T}_c)$ formant un **cluster** potentiel. La stabilisation se détecte quand :

- Les pondérations ω entre ces trois entités évoluent peu d'une itération à l'autre,
- Le **score** de cohésion interne $\Omega(\{\mathcal{V}_a, \mathcal{A}_b, \mathcal{T}_c\})$ est supérieur à un certain θ .

Sur un flux plus long (par ex. conférence), on peut suivre l'évolution des sous-groupes. Tant que "intervenant 1" parle et que la vidéo montre la même personne, le **cluster** {video, audio, texte} maintient des liaisons élevées. Dès qu'un changement se produit (nouvelle personne, nouvelle voix, nouveau sous-titre), la **stabilité** de l'ancien cluster chute, et un nouveau cluster commence à se former.

D. Algorithmes de suivi pour déclarer un cluster stable

Une approche pratique (chap. 7.3.1) consiste à calculer :

$$\Delta_{\text{mean}}(\omega, \mathcal{C}, t) = \frac{1}{|\mathcal{C}|^2} \sum_{(i,j) \in \mathcal{C} \times \mathcal{C}} |\omega_{i,j}(t+1) - \omega_{i,j}(t)|.$$

Si Δ_{mean} tombe en dessous d'un seuil δ et s'y maintient (disons, sur plusieurs itérations consécutives), on estime que le sous-groupe \mathcal{C} est **convergent**. Numériquement, cela donne un **critère** d'arrêt local.

Une autre idée est de regarder la densité ou la moyenne des **connexions** internes versus les connexions externes (type "modularité"). On peut ainsi vérifier si :

- La somme interne $\Omega_{\text{in}} = \sum_{(i,j) \in \mathcal{C}} \omega_{i,j}$ est grande,
- La somme externe $\Omega_{\text{out}} = \sum_{i \in \mathcal{C}, j \notin \mathcal{C}} \omega_{i,j}$ est petite.

Un ratio $Q = \Omega_{\text{in}} / (\Omega_{\text{in}} + \Omega_{\text{out}})$ élevé (> 0.8 , par ex.) peut signifier que \mathcal{C} est un cluster nettement séparé.

Conclusion

Pour **estimer** que l'union (segment vidéo \mathcal{V}_a , segment audio \mathcal{A}_b , texte \mathcal{T}_c) forme un **cluster stable**, on peut :

- **Surveiller la convergence** des pondérations ω en vérifiant que la variation entre deux itérations successives respecte la condition $|\omega_{i,j}(t+1) - \omega_{i,j}(t)| \leq \varepsilon$.
- **Évaluer la cohésion** interne $\Omega(\mathcal{C})$ et s'assurer qu'elle dépasse un seuil θ .
- **Contrôler la variance** interne des liens ou la **différence** entre la somme interne et la somme externe pour constater une partition stable.
- **Appliquer** des algorithmes de suivi (dérivée moyenne Δ_{mean} , ratio de modularité) pour déclarer le cluster **finalisé** au sens où la dynamique du SCN ne le modifie plus sensiblement.

En combinant ces critères, on obtient un **diagnostic** fiable de la **stabilisation** d'un cluster multimodal, essentiel pour exploiter l'**auto-organisation** dans des scénarios (chap. 8.9) où flux vidéo, audio, et textes se lient spontanément en macro-nœuds sémantiques.

8.9.2.2. Problème d'alignement temporel (vision, audio ne sont pas toujours synchrones)

Un SCN (Synergistic Connection Network) conçu pour un scénario **multimodal** (chap. 8.9) doit généralement gérer le **décalage** ou la **désynchronisation** entre différents **flux** tels que la **vision** (frames vidéo) et l'**audio** (segments sonores). Les données de ces canaux proviennent parfois de dispositifs acquis à des **fréquences** différentes ou subissent des **délais** (latences) inégaux, de sorte que l'image vue à un instant t ne coïncide pas nécessairement avec l'extrait sonore enregistré à cet instant. Cette section (8.9.2.2) développe les enjeux mathématiques et les approches adoptées dans le **DSL** pour gérer l'**alignement temporel** entre vision et audio.

A. Sources d'Asynchronie

La **désynchronisation** entre les flux **vision–audio** peut se manifester pour plusieurs raisons.

Les **différences de fréquences** d'acquisition constituent un premier facteur. Une caméra filmant à 30 images par seconde ne coïncide pas nécessairement avec un signal audio échantillonné à 48 kHz ou découpé en blocs de 20 ms, ce qui peut générer un **décalage** dans la correspondance temporelle entre les deux flux.

Les **délais de capture ou de transmission** introduisent un autre type de désalignement. Il arrive que le son soit enregistré avec un *offset* par rapport à la vidéo, ce qui entraîne un déphasage global entre ce que l'on voit et ce que l'on entend.

Enfin, des **événements asynchrones** peuvent survenir, même lorsque les flux sont **globalement alignés**. Un objet apparaissant soudainement dans une scène visuelle peut ne pas correspondre exactement à l'échantillonnage audio le plus proche, générant ainsi une **distorsion temporelle** entre l'image et le son.

D'un point de vue **mathématique**, si l'on désigne par t l'index vidéo (frame F_t) et par t' l'index audio (bloc audio $A_{t'}$), la condition de synchronie $t \approx t'$ n'est pas forcément exacte. On peut trouver un offset ℓ tel que $t' \approx t + \ell$. Une dérive dans le temps complique encore la chose car ℓ peut varier.

B. Comment modéliser la Synergie temporelle dans le DSL

Dans le **DSL**, la **synergie** $S(\mathcal{E}_{\text{vid},t}, \mathcal{E}_{\text{aud},t'})$ ne doit pas être calculée pour tous les couples (t, t') . On introduit souvent une **fenêtre** Δ afin de considérer qu'un segment vidéo à l'instant t et un segment audio à l'instant t' peuvent être corrélés au même événement lorsque $|t - t'| \leq \Delta$. Dans ce cas, $S(\text{vid}, \text{aud})$ est calculée via une métrique (co-occurrence, corrélation, etc.). À l'inverse, si $|t - t'| > \Delta$, on pose $S(\text{vid}, \text{aud}) \approx 0$. Cette **restriction** temporelle évite l'“association” d'un frame vidéo de milieu de séquence avec un segment audio survenant bien plus tard.

Le **décalage** ℓ entre l'audio et la vidéo peut prendre plusieurs formes.

Un **décalage global et constant** se produit lorsque la caméra et le micro introduisent un **offset fixe** ℓ^* . Dans ce cas, une correction simple consiste à imposer que la synergie entre les flux vidéo et audio soit nulle lorsque $|(t + \ell^*) - t'| > \Delta$. Cela garantit que seules les correspondances temporelles respectant une tolérance Δ sont prises en compte.

Un **décalage local et variable** est plus complexe, car il évolue au fil du temps. Un algorithme d'**alignement** tel que la **cross-corrélation** ou le **Dynamic Time Warping** peut estimer, pour chaque portion temporelle, une valeur optimale de $\ell(t)$. Dans ce cas, la fonction de synergie $S(\text{vid}, \text{aud})$ doit intégrer cette dynamique temporelle en tenant compte de $\ell(t)$. Cela complexifie la définition de la synergie, mais permet d'obtenir un **alignement plus précis** lorsque la désynchronisation varie progressivement au cours du temps.

Le **SCN** s'appuie sur la mise à jour $\omega_{(t),(t')} \leftarrow \omega_{(t),(t')} + \eta [S(t, t') - \tau \omega_{(t),(t')}]$. Pour que ω s'élève, il faut un **score** de synergie non nul, ce qui requiert une **fenêtre** temporelle ou un offset local approprié. Dans le cas contraire, la liaison $\omega_{(t),(t')}$ reste proche de zéro, signifiant que le SCN ne perçoit aucune co-occurrence (vision–audio) entre frame t et bloc t' .

C. Dilemme mathématique : Correction globale vs. correction locale

Si l'on a un offset ℓ^* unique (ex. +100 ms sur l'audio), on peut rectifier tous les index audio en posant $\tilde{t}' = t' + \ell^*$. Cette manipulation “aligne” l'audio sur la vidéo de manière fixe. Le SCN s'en retrouve simplifié car $S(\text{vid}, t, \text{aud}, t')$ se calcule pour $|t - (\tilde{t}')| \leq \Delta$. Cependant, toute dérive progressive ou changeante n'est pas prise en compte.

Lorsque la désynchronisation n'est pas constante dans le temps, on recourt à des méthodes plus complexes (cross-corrélation glissante, DTW). On obtient alors une fonction $\ell(t)$ où l'audio présente un décalage local de $\ell(t)$. Le SCN doit alors faire $S(\text{vid}, t, \text{aud}, t')$ non nul si $|t' - (t + \ell(t))| \leq \Delta$. D'un point de vue **computational**, cela augmente la difficulté, mais garantit un **alignement plus fin**, crucial pour des scènes variables.

D. Cas de la synergie multimodale à sampling irrégulier

La vidéo peut être échantillonnée en 30 fps, l'audio peut l'être en 16 kHz (ou plus), et des sous-titres arriver par segments d'une seconde. On ne dispose pas d'un index temporel “universel”. On manipule donc plusieurs **listes** d'horodatage. On cherche un critère permettant de déterminer si le segment vidéo $\mathcal{E}_{(\text{vid},t)}$ correspond ou non à la portion audio $\mathcal{E}_{(\text{aud},t')}$.

Deux façons de faire :

- **Re-sampling** : on projette tout dans une timeline commune $\{t_k\}$. On assigne à un bloc audio \mathcal{A}_m la référence temporelle $\tilde{t}_m \approx$ moyenne de son intervalle. Idem pour le frame vidéo \mathcal{V}_n . On compare ensuite $|\tilde{t}_m - \tilde{t}_n|$.
- **Kernel** : on déclare $S(\text{vid}, t, \text{aud}, t') = \kappa(|t - t'|, \delta)$ (ex. kernel Gaussien, ou nul si $|t - t'| > \delta$). Cela revient à imposer une pondération sur la différence temporelle, la synergie chutant rapidement au-delà d'une fenêtre δ .

E. Impact sur la Qualité d'Apprentissage

Un **mauvais** alignement temporel pénalise fortement la **fusion** multimodale. le SCN associe des frames vidéo à des segments audio inappropriés, ce qui dilue la **synergie** globale et dégrade la reconnaissance d'événements (chap. 8.8.2). À l'inverse, un alignement maîtrisé rend $S(\text{vid}, t, \text{aud}, t')$ élevé pour les paires effectivement synchrones (ex. la parole d'un locuteur et sa bouche en mouvement). Le **DSL** renforce alors les pondérations adéquates, favorisant un **cluster** stable autour de l'événement commun.

Conclusion

Le **problème d'alignement temporel** dans un **SCN** multimodal (par ex. flux visuel + flux audio) exige de prendre en compte la **désynchronisation** possible entre les canaux. Les méthodes consistent à :

- **Définir** une **fenêtre** Δ ou un **offset** ℓ^* (ou $\ell(t)$ local) pour associer un frame vidéo t à un segment audio t' .
- **Adapter** la fonction S de sorte qu'elle soit non nulle seulement si $|(t + \ell^*) - t'| \leq \Delta$ (ou autre critère d'intersection).
- **Préserver** la logique de la mise à jour DSL. La pondération $, \omega_{(\text{vid}, t), (\text{aud}, t')}$ se renforce si la synergie est haute, indiquant une co-occurrence temporelle valide.

Sans ce traitement d'alignement, la synergie calculée risque d'associer des segments “décalés” dans le temps, brouillant la **fusion** multimodale et affaiblissant la **cohérence** des clusters. L'alignement temporel apparaît donc comme un **préambule** (ou un mécanisme intégré) indispensable pour que le **DSL** identifie correctement les événements audio-visuels et en tire des **concepts** clairs.

8.9.3. Suivi et Visualisation

Dans tout **système multimodal** (chap. 8), la gestion des liaisons $\omega_{i,j}$ entre entités (images, segments audio, tokens textuels, etc.) peut rapidement devenir **complexe** si l'on ne dispose pas d'outils adéquats pour **suivre** l'évolution du réseau et **visualiser** la structure émergente. Le **DSL** (Deep Synergy Learning), en permettant des mises à jour $\omega_{i,j}(t+1)$ fondées sur la synergie $S(i, j)$, suscite la formation de **clusters** multimodaux, la fusion ou la scission de groupes, et des réajustements constants des pondérations.

Cette section (8.9.3) met l'accent sur les méthodes et **outils** permettant de **montrer** ou **d'analyser** :

- Les **changements** dans la matrice ω (ou la structure du SCN) au fil des itérations,
- L'**interaction** entre différentes modalités (images–sons, images–textes, etc.),
- L'**intégration** de ces outils dans un pipeline de classification ou de prise de décision.

8.9.3.1. Outils pour montrer l'évolution de $\omega_{i,j}$ entre images, sons, mots-clés

Dans le **DSL** (Deep Synergy Learning), un **SCN** (Synergistic Connection Network) relie des **entités** issus de divers flux multimodaux (images, segments audio, mots-clés, etc.). Au fil des itérations, les **pondérations** $\omega_{i,j}$ reliant ces entités se modifient suivant la règle de mise à jour, reflétant la **synergie** $S(i, j)$. Il est souvent **crucial** de **montrer** ou

de suivre l'évolution de ces pondérations $\omega_{i,j}$, afin de comprendre comment le réseau se structure, comment émergent les **clusters** multimodaux, et si l'**auto-organisation** se déroule correctement. La présente section (8.9.3.1) aborde plusieurs **outils** et méthodes de visualisation utiles à cet effet.

A. Matrice Dynamique : Heatmaps et Graphiques Temporels

Une stratégie simple et informative consiste à représenter la **matrice** $\Omega(t) = [\omega_{i,j}(t)]$ sous forme de **heatmap**. Chaque cellule (i,j) est colorée selon la valeur de $\omega_{i,j}(t)$. Pour un **réseau** de taille n , on obtient ainsi une **matrice** $n \times n$ visualisée. En faisant varier le temps t (ou l'itération) comme un **slider**, on peut “**dérouler**” (playback) la séquence de heatmaps, observant :

- Les liaisons qui **montent** (ex. en passant d'une valeur $\omega_{i,j} = 0.05$ à 0.7)
- Les liaisons qui s'**atténuent**
- Les **clusters** (groupes de noeuds internes) qui se dessinent via un bloc de pondérations fortes dans la heatmap.

Une **analyse** plus quantitative est possible en calculant la norme (ex. ℓ_1, ℓ_2), la **somme** $\sum_{i,j} \omega_{i,j}(t)$ ou la **distribution** $\{\omega_{i,j}(t)\}$ itération par itération. Ces statistiques renseignent sur le taux global de renforcement vs. effacement.

Une variante plus focalisée consiste à tracer, pour **quelques** paires (i,j) d'intérêt (par ex. un segment audio et une image soupçonnée de correspondre), la **courbe** $\omega_{i,j}(t)$ au cours des itérations. Cela met en lumière le **rythme** d'augmentation ou de décroissance ainsi que l'**instant** où le couple (i,j) atteint une valeur élevée et se stabilise.

De manière plus globale, on peut considérer l'évolution de la **cohésion** $\Omega(\mathcal{C}, t)$ (la somme ou la moyenne des $\omega_{i,j}$ internes à un cluster \mathcal{C}), révélant la formation d'un **groupe** stable.

B. Visualisation Graphe 2D/3D

Au lieu d'afficher une **matrice**, on peut construire un **graphe** où chaque **noeud** représente une entité (image, mot-clé, segment audio, etc.), et chaque **arête** possède un **poids** $\omega_{i,j}$. Un algorithme de layout (type ForceAtlas, Force-Directed) place les noeuds en 2D ou 3D, éloignant ceux dont $\omega_{i,j} \approx 0$ et rapprochant ceux reliés par un $\omega_{i,j}$ plus fort.

En le faisant évoluer au fil des itérations (chaque itération du DSL correspond à une “image” de ce graphe), on **voit** la structure se **réorganiser** graduellement. Des noeuds “image” et des noeuds “audio” fortement liés se rapprochent, formant des **clusters** multimodaux. Les arêtes trop faibles disparaissent ou restent filiformes (fines, peu visibles), tandis que les arêtes fortes (pondérations élevées) s'épaissent.

Dans un environnement multimodal complet (images, sons, mots, règles, etc.), la **visualisation** globale peut devenir saturée. On peut se **focaliser** sur deux modalités (par ex. *images* vs. *mots*) en ne montrant que les liaisons $\omega_{i,j}$ pour $i \in \text{Images}, j \in \text{Mots}$. Cela revient, en pratique, à extraire un **sous-bloc** de la matrice $\Omega(t)$ ou un **sous-graphe** du SCN, limité aux entités image–texte. On voit alors plus facilement quelles **images** se connectent à quels **mots**.

C. Outils de Focalisation sur Groupes Particuliers

Outre la **sélection** d'une modalité (image, audio, texte), on peut filtrer les $\omega_{i,j}$ trop faibles (en dessous d'un seuil θ). Cela permet de clarifier la **vision** du graphe. Seules les **connexions** qui ont atteint un certain degré de solidité (par ex. $\omega_{i,j} > 0.3$) sont affichées, mettant en évidence un **clustering** mieux délimité.

Si l'on sait qu'un **cluster** \mathcal{C} (ex. un sous-ensemble de frames et de segments audio) s'est formé, on peut **surveiller** la somme interne $\sum_{(i,j) \in \mathcal{C} \times \mathcal{C}} \omega_{i,j}(t)$ et la somme externe $\sum_{i \in \mathcal{C}, j \notin \mathcal{C}} \omega_{i,j}(t)$. En temps réel, on observe si la cohésion interne du cluster \mathcal{C} continue de croître ou stagne, et si les liaisons vers l'extérieur décroissent ou non, reflétant une **stabilisation** (chap. 8.9.2 sur la stabilité).

D. Exemples concrets

On peut disposer d'un ensemble de **mots-clés** (tokens ou embeddings textuels) et d'un ensemble de **images** (embeddings visuels). La **matrice** $\omega(t)$ reflète la force de correspondance image-mot. En affichant une heatmap, on verra, itération après itération, **qui** se lie à **qui**. Un groupe d'images "chats" se connectera aux mots "cat, kitten, feline", tandis qu'un autre groupe "voiture" pointera vers "car, vehicle, road". On pourra surveiller la formation ou la désagrégation de blocs dans la heatmap.

Dans un **graph** 2D/3D, chaque nœud audio (ex. segment musical) se retrouvera à proximité des nœuds vidéo où la scène "correspond" (même intervalle de temps, cohérence de contenu). Au fur et à mesure, on voit se constituer des **clusters**. Par exemple, un "cluster urbain" (vidéo de rues + bruits de moteur + sons de klaxon) et un "cluster nature" (vidéo de forêt + chant d'oiseaux). Les arêtes $\omega_{i,j}$ s'épaississent pour ces entités de forte synergie.

E. Suivi de l'Évolution Temporelle

Pour un **monitoring** en ligne, on **enregistre** $\Omega(t)$ ou des mesures résumées à intervalle régulier. On **visualise** ensuite l'évolution à l'aide d'un "**player**" temporel de type timeline ou sous forme de courbes représentant $\omega_{i,j}$ en fonction du temps t . On peut croiser ces observations avec les **critères** de stabilisation afin d'identifier le moment où un **cluster** se consolide.

Conclusion

Pour **montrer** l'évolution des **pondérations** $\omega_{i,j}$ dans un **SCN** (images, sons, mots-clés, etc.), on dispose principalement de :

- **Matrice** (heatmap) dynamique, où l'on voit la formation de blocs $\omega_{i,j}$ forts,
- **Visualisation** graphe 2D/3D animée (layouts). Les nœuds (entités) se rapprochent ou s'éloignent selon la force ω , exhibant la naissance de **clusters** et la dissipation des liens faibles,
- **Focalisation** sur des **sous-ensembles** (filtrage par modalité, par intensité de ω , etc.), offrant des analyses ciblées,
- **Suivi** temporel (dashboard, archivage) permettant de tracer l'évolution de la **cohésion** d'un cluster ou la progression d'une liaison $\omega_{i,j}$.

Ces outils assurent une **lecture** claire de la **dynamique** auto-organisée, révélant comment le DSL parvient à **unir** des entités multimodales au sein de **concepts** ou **macro-nœuds** cohérents.

8.9.3.2. Tableaux de bord (dashboard) multimodal, exploitation possible dans un pipeline de classification

Dans un **DSL** (Deep Synergy Learning) appliqué à un **SCN** (Synergistic Connection Network) multimodal, il est essentiel de *monitoreder* l'évolution et la structure de la fusion (ch. 8.8 et 8.9) de manière **interne** et en temps réel. Les **tableaux de bord** (ou *dashboards*) répondent à ce besoin. Ils fournissent une **vue** synthétique de l'état du réseau, des **pondérations** ω , des **clusters** formés, et des **tendances** de l'auto-organisation. Cette section (8.9.3.2) décrit comment de tels tableaux de bord peuvent être intégrés dans un **pipeline** plus large, par exemple un **workflow** de **classification** multimodale.

A. Notion de Tableaux de Bord (Dashboards) Multimodaux

Les **tableaux de bord** permettent de **visualiser** et **superviser** l'évolution du SCN. Dans un **contexte** multimodal, ils offrent une vue sur la **structure** des clusters et macro-nœuds reliant des entités d'origines différentes, telles que les images, l'audio et le texte. Ils affichent également les **pondérations** $\omega_{i,j}(t)$ les plus élevées ainsi que leur distribution et permettent de suivre les **dynamiques** de la synergie S ou de la pondération ω au fil du temps.

Dans la pratique, ces dashboards comprennent plusieurs **modules** ou **vues**. Une **vue graphe** représente en 2D ou en 3D les nœuds et leurs arêtes pondérées ω , souvent avec un layout de type “force-directed”. Une **heatmap** ou **matrice** $\Omega(t)$ permet d’analyser un sous-bloc spécifique, comme les relations image–texte ou image–audio, afin d’observer l’évolution de certaines liaisons. Des **courbes** temporelles illustrent la progression de l’**auto-organisation**, en affichant par exemple la somme de ω ou la fonction d’énergie $J(\Omega)$ au cours des itérations (chap. 2.2.2 ou 7.2). Enfin, des **indicateurs** tels que la taille des clusters, la densité des liens ou encore l’entropie permettent de suivre la **cohésion** des sous-groupes multimodaux.

B. Exploitation dans un Pipeline de Classification

Dans un **pipeline** de classification (en vision, NLP, etc.), on opère souvent une **fusion** à un niveau “feature” (on concatène des vecteurs) ou à un niveau “decision” (on combine des scores). Avec un **SCN** (voir chap. 8.8), la fusion se produit par **auto-organisation** des entités multimodales, générant **clusters** ou macro-nœuds. On peut alors **extraire** ces **macro-nœuds** comme des **features** pour un classifieur en aval. La structure permet de **déterminer** de manière auto-organisée quel segment audio s’attache à quelle scène vidéo et d’**exploiter** la matrice ω pour prendre des décisions hiérarchiques.

Le **dashboard** multimodal joue un **double rôle**. Il permet de **moniturer** en temps réel pour vérifier si le SCN regroupe correctement les flux, par exemple en associant les segments vidéo “chien” aux segments audio “abolement”. Il sert également à **orienter** ou **paramétrier** le pipeline, permettant à un opérateur ou à un module automatique de modifier η , τ ou des coefficients d’inhibition selon l’état de la structure observée. De plus, on peut réinjecter un “recuit” (voir 7.3.2) en cas de blocage détecté.

Lorsque le **SCN** stabilise un **cluster** \mathcal{C} (par ex. ensemble d’images, audio, mot-clé sémantique), un **classifieur** (apprentissage supervisé, ou un module symbolique) peut en déduire une étiquette (ex. “chat” ou “voiture”). Le tableau de bord montre aussi la **progression** de cette classification, indiquant si le cluster \mathcal{C} correspond à “chat” (forte présence de miaulements et de la forme féline détectée).

C. Intégration de Paramètres et Feedback

Un **tableau de bord** peut comprendre des **curseurs** permettant d’ajuster les paramètres η , τ et le “taux” d’inhibition. Il peut également inclure un **bouton** de recuit “light” ou “heavy” pour intervenir en cas de **blocage** (minima local) dans la formation des clusters.

Le pipeline de classification en aval reçoit la **nouvelle** structure ω mise à jour. Cette interaction signifie qu’on injecte un **contrôle** humain (ou algorithmique) dans les hyperparamètres du DSL, se basant sur la **visualisation** pour décider quand intervenir.

Dans certains cas, on peut disposer d’un **feedback** partiel (par ex. un label correct pour un segment vidéo–audio), permettant de **valider** ou **invalider** un cluster formé par le SCN. On peut afficher dans le dashboard la **précision** ou la **réussite** de la classification multimodale associée, renforçant la **compréhension** du niveau de confiance.

D. Cas Pratique

Supposons un pipeline où le **SCN** reçoit des frames vidéo pour reconnaître la personne qui parle, des segments audio pour identifier la voix et du texte pour analyser les sous-titres. Un **dashboard** permet alors de visualiser plusieurs éléments clés :

- Un **graphe 2D** représentant les **connexions** et la pondération ω entre segments vidéo affichant “personne A”, segments audio correspondant à “voix A” et sous-titres mentionnant “M. A.”.
- Des **courbes** illustrant la somme des liens $\sum_{(vid,aud)} \omega_{i,j}$, mettant en évidence la force des correspondances audio–visuelles.
- Une **liste des clusters actuels**, par exemple, “Intervenant 1 (vidéo #2, audio #5, texte #1)”.

L’utilisateur ou un module supervisé peut alors **intervenir** si un cluster “Intervenant 1” est incorrect, en ajustant un paramètre ou en injectant une correction.

Le pipeline final (classification “Qui parle en ce moment ?”) s’appuie sur ce cluster auto-organisé pour étiqueter “c’est la voix de M. A, vue à l’écran”.

Sans tableau de bord, on a un SCN qui s’auto-organise en **boîte noire**. On ignore si certains liens ω se sont **excessivement** solidifiés à cause d’un bruit ou d’une confusion (ex. un segment audio d’oiseau relié à un plan vidéo de chat). Le dashboard **révèle** visuellement ces erreurs potentielles. On peut alors corriger ou ajuster le pipeline (ch. 7.4 sur l’inhibition ciblée).

Conclusion

Les **tableaux de bord** (dashboard) multimodaux sont des **outils** essentiels pour **moniturer l’auto-organisation** du SCN. Ils **montrent** la structure (pondérations ω , clusters formés), **suivent** l’évolution (heatmaps, graphes animés) et **facilitent** la décision. Intégrés dans un **pipeline** de classification :

- Ils **documentent** la répartition et la cohésion des entités (image, audio, texte),
- Ils fournissent un **contrôle** interactif ou algorithmique sur η, τ , l’inhibition, etc.,
- Ils **alimentent** la **sortie** classification en indiquant des clusters multimodaux consolidés, réutilisables pour l’inférence.

Cette **approche** promeut une **transparence** et une **adaptabilité** accrues dans un workflow complet, combinant l’**auto-organisation** multimodale et la **supervision/ajustement** continu dans l’optique d’une classification fiable et évolutive.

8.10. Limites, Défis et Pistes de Recherche

Même si le **DSL multimodal** (voir chapitres précédents) offre de nouvelles capacités pour intégrer, associer et organiser des flux variés (image, audio, texte, etc.), il se heurte inévitablement à des **contraintes** liées à la **complexité** des données réelles, à la **qualité** parfois imparfaite des embeddings, et à des enjeux de **sécurité** et de **biais**. La présente section (8.10) met en relief ces **limites** et **défis**, tout en esquissant des **pistes de recherche** pour aller plus loin.

8.10.1. Complexité

L'un des principaux obstacles auxquels fait face un **SCN** (Synergistic Connection Network) en mode multimodal repose sur le **nombre** d'entités à traiter. Segmenter un flux vidéo en frames, un flux audio en unités ou fonctions, un texte en phrases ou en tokens peut entraîner une explosion de la quantité totale d'éléments $\{\mathcal{E}_i\}$. Cela se traduit par une **dimension** potentiellement gigantesque pour la matrice $\{\omega_{i,j}\}$, d'où un **coût** de mise à jour et de recherche en $O(n^2)$.

8.10.1.1. Le nombre d'entités explose si on segmente un flux vidéo (frames), audio (fonctions) et texte (phrases)

Dans un **SCN** (Synergistic Connection Network) destiné à la **fusion multimodale** (vision, audio, texte, etc.), la quantité d'**entités** à traiter peut croître de façon considérable dès lors qu'on segmente finement chaque flux. Les sections précédentes (chap. 8.8, 8.9) mettent en évidence l'intérêt de la segmentation pour associer segments vidéo à segments audio. La présente section (8.10.1.1) détaille la **problématique** qui en découle, à savoir l'**explosion** du nombre de nœuds à gérer dans le **DSL**, et donc d'un potentiel $O(n^2)$ liaisons $\{\omega_{i,j}\}$.

A. Segmentation Vidéo

Un **flux vidéo** de durée D (quelques minutes, voire plus) peut être découpé en **centaines** ou **milliers** de frames si l'on conserve la plupart des images (ex. 25 ou 30 fps). Chaque **image** devient alors une **entité** $\mathcal{E}_i^{(vid)}$. De surcroît, si l'on opte pour une segmentation plus granulaire (patchs par frame, détection d'objets locaux), la multiplication des entités s'accroît davantage.

Avec F frames et P patchs par frame, on obtient $n = F \times P$ entités vidéo. Si F s'élève à plusieurs milliers et P n'est pas négligeable, la taille n devient rapidement **massive**. Dans un **SCN**, le nombre de liaisons ω peut atteindre $O(n^2)$. À mesure qu'on traite des vidéos plus longues ou plus riches en segments, ce volume de nœuds et de liens menace de **saturer** la mémoire et le temps de calcul.

B. Segmentation Audio

Le **flux audio** s'analyse généralement via des **blocs** d'une durée de quelques millisecondes (10, 20, 40 ms) ou via des unités (segments phonétiques, etc.). Chaque bloc se convertit en un **vecteur** de features (MFCC, spectrogramme) puis en une entité $\mathcal{E}_j^{(aud)}$. Comme la fréquence d'échantillonnage peut être élevée (ex. 16 kHz, 48 kHz), le **nombre** de segments audio grimpe vite à des **milliers** ou plus, tout au long d'un enregistrement de plusieurs minutes.

Un exemple concret illustre cette problématique. Dix minutes d'audio, segmenté en **fenêtres** de 5 ms (non superposées), dégage 120 000 blocs. Cela engendre autant d'entités dans le **SCN** si aucun pré-regroupement n'est opéré. La mise à jour en $O(n^2)$ devient alors problématique.

C. Segmentation Textuelle

Les **flux** de texte (ex. sous-titres, transcription, documents associés) peuvent aussi générer de très **nombreuses** entités. On segmente souvent le texte en **phrases** ou en **tokens** (mots, sous-mots). Une conférence, un livre ou un corpus volumineux conduit potentiellement à des **dizaines** voire **centaines de milliers** de tokens.

Au sein du **SCN**, chaque **token** ou **phrase** serait un noeud $\mathcal{E}_k^{(txt)}$. Or, ajouter une telle masse de nœuds alourdit énormément la structure, gonflant la matrice $\{\omega_{i,j}\}$. Lorsque le **DSL** traite la synergie **image–texte** (chap. 8.8.1) ou **audio–texte** (subtitling), l'explosion de n'entités nuit à la faisabilité d'une mise à jour naïve en $O(n^2)$.

D. Fusion Multimodale : le cumul des entités

Lorsque le **SCN** veut simultanément gérer :

- Les entités **vidéo** (n_{vid}) extraites des frames ou patchs,
- Les entités **audio** (n_{aud}) issues de la segmentation sonore,
- Les entités **textuelles** (n_{txt}) (phrases, tokens, etc.),
- Éventuellement d'autres sources (capteurs, règles symboliques),

on obtient un volume total $n_{\text{vid}} + n_{\text{aud}} + n_{\text{txt}} + \dots$. Ce total n peut atteindre rapidement le **million**, rendant impossible toute gestion brute de $n(n - 1)/2$ pondérations ω . Les algorithmes DSL, s'ils ne prévoient pas de **stratégies** de limitation, risquent d'**exploser** en mémoire et en temps de calcul.

Conclusion

Lorsque l'on segmente un flux **multimédia** (vidéo, audio, texte) de façon fine, le nombre d'**entités** dans un **SCN** croît potentiellement de manière **exponentielle**. On observe une accumulation rapide de frames, de blocs audio millisecondes et de tokens textuels en grande quantité. Ce phénomène aboutit à un **problème de scalabilité** :

- La **matrice** des pondérations ω atteint une taille $O(n^2)$ difficile à stocker,
- Les boucles de mise à jour en $O(n^2)$ sont inenvisageables dès que $n \sim 10^5$.

Ainsi, la simple segmentation “détailée” d'un flux vidéo, audio, texte se heurte à un mur de complexité. La suite du chapitre (8.10.1.2) s'attachera à discuter les **heuristiques** et **algorithmes** (sélection de voisinage, élagage, fusion hiérarchique) permettant de **contenir** cette explosion, tout en préservant la logique auto-organisée du DSL dans un cadre multimodal à haute granularité.

8.10.1.2. Nécessité d'heuristiques (k plus proches voisins, cluster gating, etc.)

Lorsque la **taille** du **SCN** (Synergistic Connection Network) croît significativement, la charge de calcul en $O(n^2)$ devient inacceptable. Il faut envisager des **stratégies** d'allégement pour gérer les **pondérations** ω entre n entités sans manipuler toutes les paires (i, j) . Les sections précédentes (8.10.1.1) ont montré comment le **nombre** d'entités peut exploser en multimodal (images + audio + texte), ce qui **impose** l'emploi d'**heuristiques**. L'idée principale est de **resterindre** la mise à jour $\omega_{i,j}$ à un **sous-ensemble** limité de paires. Cette section (8.10.1.2) présente deux familles d'approches, le “ k plus proches voisins” (k -NN) et le “cluster gating”.

A. Heuristique du “ k plus proches voisins” (k -NN)

L'approche k -NN **restreint** la recherche de **synergie** $S(i, j)$ à un **voisinage** $N_k(i)$ autour de l'entité \mathcal{E}_i .

On **définit** un critère (souvent basé sur des embeddings, distances ou similarités initiales) pour **classer** les autres entités $\{\mathcal{E}_j\}_{j \neq i}$ par ordre de proximité vis-à-vis de \mathcal{E}_i .

On **choisit** les k plus proches d'entre elles (ou un nombre restreint équivalent).

La **mise à jour** $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \dots$ ne s'applique qu'à ces k voisins (et possiblement, à quelques nouveaux candidats si des événements indiquent que la distance a évolué). Sinon, $\omega_{i,j}$ reste figé à 0 ou une valeur ancienne.

Sur le plan **mathématique**, l'algorithme DSL se limite ainsi à $O(n k)$ ou $O(n \log n)$ si l'on actualise régulièrement les k plus proches voisins via une structure comme un **kd-tree** ou **ball-tree**. Cela ramène la complexité de $O(n^2)$ à $O(n k)$, ce qui devient gérable si $k \ll n$.

Les **avantages** de cette approche résident principalement dans l'**économie** en temps de calcul, puisque seules les k entités les plus pertinentes sont explorées pour chaque nœud du réseau. Cela permet d'éviter l'explosion combinatoire du nombre de paires à évaluer tout en préservant les liens significatifs. De plus, un **filtrage** automatique s'opère sur les connexions les plus faibles ou improbables, ce qui limite la saturation du réseau et garantit une organisation plus claire et plus efficace du **SCN**.

Toutefois, cette approche présente certaines **limites**. Un phénomène de **cloisonnement** peut apparaître lorsque, dans un premier temps, une entité \mathcal{E}_i est classée parmi les k plus proches voisins d'une autre entité \mathcal{E}_j , mais que l'inverse ne se produit pas. Cela introduit un **biais** structurel, où certaines relations potentielles ne peuvent pas émerger naturellement à cause d'un voisinage restreint. De plus, il existe un risque d'**erreurs** de partition lorsque des liaisons pertinentes, bien que caractérisées par une distance initialement grande, sont écartées trop tôt. Une solution consiste à prévoir un **rafraîchissement** périodique du voisinage, permettant ainsi à des liens **long-courriers** mais sémantiquement valides d'apparaître progressivement et d'être intégrés au sein du **SCN**.

B. Cluster Gating : Filtrage sur la Base de Clusters Émergents

Une seconde approche consiste à **grouper** (même de façon approximative) les entités en **clusters** (ou super-nœuds), puis à n'autoriser la mise à jour $\omega_{i,j}$ qu'entre entités dans un même cluster ou entre clusters voisins. On parle de "**cluster gating**". Cela peut être vu comme un **gating** binaire $G(i,j) \in \{0,1\}$:

$$\omega_{i,j}(t+1) = G(i,j) \left[\omega_{i,j}(t) + \eta (S(i,j) - \tau \omega_{i,j}(t)) \right],$$

de sorte que si $G(i,j) = 0$, la liaison $\omega_{i,j}$ n'est pas mise à jour (reste 0 ou inchangée). Ce "gating" s'active ($= 1$) pour les paires (i,j) jugées **compatibles**. i et j appartiennent au même cluster ou à des clusters proches dans l'arbre hiérarchique (chap. 6).

Le **bénéfice** du **cluster gating** est de réduire considérablement le **nombre** de liaisons évaluées. Une fois qu'un **gros** cluster \mathcal{C}_α s'est stabilisé, on peut limiter l'évolution $\omega_{i,j}$ aux entités à l'intérieur de \mathcal{C}_α ou aux entités proches, sans s'occuper de paires très distantes.

Le **risque** est qu'un cluster ou super-nœud **verrouille** excessivement la structure, rendant difficiles les re-liaisons inter-clusters. On peut envisager un **rafraîchissement** occasionnel, réexaminant certaines paires inter-clusters pour autoriser des réassignments.

C. Pourquoi de telles Heuristiques ?

La **motivation** principale est la **scalabilité**. En se passant d'un traitement exhaustif $O(n^2)$, on ramène le calcul à $O(n k)$ ou $O(n c)$ selon la méthode. Les heuristiques veillent à ne pas brider trop la découverte de **nouveaux** liens potentiellement significatifs.

Conclusion

La **croissance** du nombre d'entités dans un **SCN** multimodal rend **inévitable** le recours à des **heuristiques** telles que :

- **k plus proches voisins** (k-NN) : chaque entité n'actualise $\omega_{i,j}$ que pour ses k plus proches,
- **Cluster gating** : on ne met à jour ω qu'à l'intérieur ou à la frontière de clusters déjà identifiés.

Ces stratégies réduisent le coût de $O(n^2)$ à un volume d'opérations plus **gérable**, tout en conservant une **logique** auto-organisée. Le DSL demeure opérationnel sur de grands ensembles de frames vidéo, segments audio et unités textuelles, sans s'effondrer face à la complexité. Ces heuristiques forment donc un **socle** crucial dans les pipelines réels où la segmentation (ch. 8.10.1.1) produit un nombre massif d'entités multimodales, empêchant tout calcul complet sur l'ensemble des paires.

8.10.2. Qualité de la Synergie

Dans un **SCN** (Synergistic Connection Network) multimodal, la pertinence de la **synergie** $S(i, j)$ dépend étroitement de la **qualité des représentations** (embeddings) associées à chaque modalité (image, texte, audio, etc.). Si ces embeddings sont imparfaits — bruités, mal calibrés ou incomplets —, il en résulte un calcul de synergie moins fiable, susceptible de **tromper** le réseau dans la formation de clusters ou de super-nœuds inappropriés. La présente section (8.10.2) vise à examiner comment **évaluer** ces embeddings, puis comment les **calibrer** ou les **normaliser** pour conserver un niveau de synergie cohérent à travers plusieurs canaux.

8.10.2.1. Évaluation des embeddings multimodaux : s'ils sont imparfaits, le SCN peut se tromper

Un **SCN** (Synergistic Connection Network) déployé dans un **DSL** (Deep Synergy Learning) multimodal s'appuie fortement, pour son calcul de **synergie** $S(i, j)$, sur des **embeddings** sub-symboliques (vision, audio, texte) censés traduire la similarité ou la correspondance entre entités. Lorsque ces embeddings sont **imparfaits** (qualité sous-optimale, biais, surapprentissage...), la mesure $S(\mathcal{E}_i, \mathcal{E}_j)$ en est faussée. De telles erreurs d'estimation engendrent une **auto-organisation** trompeuse. Le SCN peut créer ou renforcer des liens $\omega_{i,j}$ là où la **réalité** (sémantique) ne le justifie pas, ou, à l'inverse, négliger des connexions pertinentes. La présente section (8.10.2.1) discute comment l'**embedding** imparfait se répercute sur la fiabilité du SCN, et pourquoi il importe de **surveiller** la qualité des embeddings avant ou pendant l'exécution du DSL.

A. Impact d'un Embedding Imparfait

Dans un **SCN** multimodal, chaque **entité** \mathcal{E}_i (image, texte, segment audio, etc.) se voit associé un **vecteur** \mathbf{v}_i . La **synergie** $S(i, j)$ est souvent calculée comme une **similarité** (cosinus, produit scalaire, distance exponentielle) ou une mesure statistique (information mutuelle) dérivée des embeddings. Si les embeddings sont **imparfaits** — par exemple, mauvaise généralisation, confusion dans les features, absence de capture des nuances sémantiques — la **valeur** $S(i, j)$ risque de ne plus refléter la véritable proximité ou compatibilité sémantique des entités \mathcal{E}_i et \mathcal{E}_j .

Considérons :

- $\mathbf{v}_i^{(\text{image})}$: embedding tiré d'un **CNN** trop peu entraîné, confondant parfois un chien et un chat,
- $\mathbf{v}_j^{(\text{txt})}$: embedding textuel issu d'un modèle à couverture insuffisante (les mots "cat" et "car" se révèlent trop proches).

Le **SCN**, au moment de calculer $S(\mathcal{E}_i^{(\text{image})}, \mathcal{E}_j^{(\text{txt})})$, est induit en erreur si l'**embedding** cosinus $\cos(\mathbf{v}_i, \mathbf{v}_j)$ s'avère artificiellement élevé pour la paire (chien, "car") ou trop faible pour la paire (chien, "dog"). On verra alors le **DSL** renforcer $\omega_{(\text{chien}), (\text{car})}$ et ignorer $\omega_{(\text{chien}), (\text{dog})}$, ce qui fausse le regroupement (cluster incorrect).

B. Conséquences Mathématiques dans le SCN

La dynamique DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$$

s'opère à chaque itération, où $S(i, j) = \text{sim}(\mathbf{v}_i, \mathbf{v}_j)$. Si $\mathbf{v}_i, \mathbf{v}_j$ sont mal positionnés dans l'espace d'embedding, $\text{sim}(\mathbf{v}_i, \mathbf{v}_j)$ peut être **trop** grand ou **trop** petit par rapport à la réalité. Cela engendre un **renforcement** de liens $\omega_{i,j}$ qui ne devraient pas l'être et un **abandon** ou une **sous-estimation** de liaisons qui, dans un embedding correct, seraient révélées comme pertinentes.

On obtient alors un **réseau** ω dont la topologie n'est pas conforme à l'authentique structure sémantique, et qui risque de converger vers un **minimum local** "faux".

Dans un **DSL**, les **clusters** émergents (groupes de noeuds fortement connectés) correspondent en principe à des **concept**s ou des ensembles sémantiques partagés (p. ex. "chien + aboiement + mot dog"). Mais si l'embedding fait

confondre “chien” et “chat”, le SCN peut aboutir à un cluster (chien, chat, meowing) ou (chien, “cat”), un bloc **incohérent**. Les pondérations internes ω se stabilisent néanmoins parce qu’aucun autre signal n’est venu **corriger** l’erreur.

En somme, le SCN se fiant aux embeddings donne un “garbage in, garbage out”. Si la similarité S est trompeuse, l’organisation ω l’est aussi.

C. Nécessité d’une Évaluation et d’une Calibration des Embeddings

Avant d’utiliser un embedding (image, audio, texte) dans le SCN, il est prudent de **contrôler** sa qualité. Par exemple, si on a un jeu de test (images + labels), on vérifie si la distance cosinus reflète la proximité de labels. Ou dans le texte, on regarde si les embeddings distinguent bien “cat” et “car”. L’échec de ce test incite à **réentraîner** ou **raffiner** l’embedding.

Si on ne peut pas réentraîner, on peut recourir à une **calibration** statistique, par ex. un **scale** ou un **offset** tenant compte des distributions courantes, un alignement sémantique (coordonner l’échelle de l’espace image et l’espace texte). Des méthodes *fine-tuning cross-modal* (ex. style CLIP) permettent parfois de réduire le fossé entre embeddings hétérogènes.

Dans certains cas (voir chap. 8.6–8.7), on peut injecter un **feedback** top-down. Si le SCN détecte des incohérences massives dans un cluster, on soupçonne un problème d’embedding ; on envoie un signal ou on “réapprend” localement le mapping pour mieux aligner $\mathbf{v}_i, \mathbf{v}_j$. Cela suppose un DSL unifié ou couplé à l’étape d’entraînement de l’embedding.

Conclusion

Lorsque les **embeddings** multimodaux (image, audio, texte) sont **imparfaits**, la **mesure** $S(i, j)$ calculée par le SCN devient peu fiable. Le **DSL** risque alors de :

- **Renforcer** des liaisons $\omega_{i,j}$ erronées,
- **Omettre** des correspondances légitimes,
- Former des **clusters** incohérents ou des attracteurs non pertinents.

Dès lors, un **examen** de la **qualité** (cohérence, test sur un set de référence) des embeddings s’impose, voire un **calibrage** ou un *fine-tuning*, afin d’assurer que la similarité sub-symbolique $\text{sim}(\mathbf{v}_i, \mathbf{v}_j)$ reflète au mieux la **réalité** sémantique. Faute de quoi, même un SCN paramétré correctement peut “se tromper”, victime des biais ou limitations de l’embedding. Le chapitre suivant (8.10.2.2) portera justement sur les **approches** pour **calibrer** ou **normaliser** la fonction de synergie S en vue de pallier des embeddings imparfaits.

8.10.2.2. Approches pour calibrer $S(i, j)$, ex. normalisation cross-modal

Lorsque l’on applique un **SCN** (Synergistic Connection Network) à un **DSL** (Deep Synergy Learning) dans un **contexte multimodal** (images, audio, textes, etc.), la **fonction** $S(i, j)$ quantifiant la synergie entre deux entités \mathcal{E}_i et \mathcal{E}_j se heurte à des **hétérogénéités** de **modalités** et d’**échelles**. Les différentes **sources** (vision, audio, texte...) ne délivrent pas des vecteurs ou mesures directement comparables. Cette section (8.10.2.2) détaille les méthodes de **calibrage** et de **normalisation** visant à **unifier** ou du moins **harmoniser** les scores de similarité/distance, afin de rendre $S(i, j)$ **cohérent** à l’échelle du SCN tout entier.

A. Problématique de la Normalisation Cross-Modal

Chaque **modalité** (image, audio, texte, etc.) produit des embeddings $\mathbf{v} \in \mathbb{R}^d$ selon des **dimensions** et des **échelles** variables. On peut par exemple avoir :

- Un embedding visuel $\mathbf{x}_i^{(\text{img})} \in \mathbb{R}^{2048}$ provenant d’un CNN,
- Un embedding audio $\mathbf{x}_j^{(\text{aud})} \in \mathbb{R}^{256}$ résument un segment sonore,

- Un embedding textuel $\mathbf{x}_k^{(\text{txt})} \in \mathbb{R}^{768}$ d'un modèle transformer.

Comparer directement $\|\mathbf{x}_i - \mathbf{x}_j\|$ ou $\cos(\mathbf{x}_i, \mathbf{x}_j)$ sans **calibrer** l'échelle (dimension, distribution, amplitude) peut poser problème. Une **modalité** pourrait avoir des valeurs de norme plus élevées par construction, ou la distribution statistique des embeddings diffère drastiquement, conduisant à des **scores** S difficilement comparables.

Même **au sein** d'une modalité unique, on peut générer plusieurs **scores**. Un pour la forme globale, un autre pour la texture, un troisième pour la scène audio globale, un quatrième pour la voix localisée, etc. Dans un environnement **multimodal**, on obtient un éventail de **fonctions** $S^{(m)}$, qu'il faut **fusionner** en un **score** global $S(i, j)$. Pour éviter qu'une **mesure** (avec amplitude plus élevée) ne domine, on doit **normaliser** ou **pondérer** chacun de ces scores.

B. Formulations Mathématiques de Normalisation

Une **méthode** répandue est de **centrer–réduire** les valeurs de similarité d'une modalité. Supposez qu'on considère la distribution $\{S^{(\text{mod})}(i, j)\}$ sur l'ensemble des paires (i, j) de cette modalité, on définit :

$$\tilde{S}^{(\text{mod})}(i, j) = \frac{S^{(\text{mod})}(i, j) - \mu_{\text{mod}}}{\sigma_{\text{mod}}}$$

pour un offset μ_{mod} (moyenne) et une échelle σ_{mod} (écart-type). On peut ensuite le **ramener** à $[0,1]$ par un min–max ou une sigmoïde. Ainsi, on obtient des **scores** comparables entre modalités. Sur le plan **mathématique**, on veille à ce que chaque flux ait une distribution de similarités homogène, évitant qu'un flux “embeddings plus longs” dépasse systématiquement les autres.

Si on a M modalités, on peut écrire :

$$S(i, j) = \sum_{m=1}^M \alpha_m \tilde{S}^{(m)}(i, j),$$

où $\tilde{S}^{(m)}(i, j)$ est la **version** normalisée intramodale de la similarité $S^{(m)}$, et α_m (tel que $\sum_m \alpha_m = 1$) reflète l'importance de chaque modalité. C'est un **mélange** convexe. On s'assure ainsi que la synergie finale $S(i, j)$ est **équilibrée**.

Plus **complexe**, on peut laisser α_m **varier** dans le temps, ou être optimisé par un mini-algorithme. Le **SCN** peut, par exemple, estimer qu'une modalité est moins fiable (captor bruité) et réduire α . Cette **auto-adaptation** augmente le **degré** de plasticité dans le DSL, puisqu'on ne se borne plus à un paramétrage statique.

C. Exemples de Cross-Modal Normalisation

Dans des embeddings textuels (ex. BERT), on emploie souvent la **similarité cosinus**, bornée entre $[-1, 1]$. Pour un embedding image basé sur des distances euclidiennes, on la convertit en une similarité (ex. $S_{\text{img}} = \exp(-\alpha \|\mathbf{x}_i - \mathbf{x}_j\|)$) bornée entre $(0, 1]$. On peut ensuite **rééchelonner** ces deux scores pour qu'ils aient des **moyennes** ou des **intervalles** comparables.

Dans certains flux (ex. intensités spectrales audio), on peut recourir à un **log** ou à une **sigmoïde** pour “compresser” l'éventail de valeurs. Par exemple,

$$\hat{S}^{(m)}(i, j) = \text{sigmoid}\left(\gamma(S^{(m)}(i, j) - \mu^{(m)})\right),$$

de sorte à limiter les **extrêmes** et ramener le score dans $[0, 1]$. Cela se révèle utile quand la **distribution** des mesures dans une modalité présente de larges écarts ou comporte des outliers.

D. Intérêt pour le DSL

Une fois la fonction de synergie $S(i, j)$ normalisée, la règle DSL

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)]$$

part d'une **base** plus fiable, on ne risque pas de voir une modalité "alpha" dominer simplement parce que ses valeurs S se situaient dans un registre d'amplitude plus élevé. Chaque canal **contribue** de façon plus proportionnée, ce qui **renforce** la cohérence et la **stabilité** des clusters multimodaux (chap. 8.9, 8.10).

Si, dans un flux, la similarité se situe **toujours** ~ 0.9 (surdimension), alors que dans un autre, elle varie $\sim 0.2\text{--}0.5$, le SCN penche en faveur du premier, aboutissant à un **biais**. Un calibrage cross-modal évite ces écarts amplifiés et assure une **convergence** plus homogène.

Conclusion

Pour **calibrer** $S(i, j)$ dans un **DSL** multimodal, on recourt à

- **Normalisation** intramodale (centrer–réduire, min–max, log-scale) afin que chaque flux fournisse des **scores** comparables.
- **Pondération** (ex. mélange convexe) pour combiner les similarités normalisées de plusieurs modalités.
- **Auto-adaptation** potentielle où le DSL ajuste dynamiquement les poids modaux si un canal s'avère peu fiable.

D'un point de vue **mathématique**, ce calibrage assure une **comparabilité** et une **équité** entre modalités, protégeant la mise à jour $\omega_{i,j}$ de dérives dues à l'échelle. Sur le plan **pratique**, cela **améliore** la robustesse du SCN et la **cohésion** des clusters multi-canaux. Aucune modalité ne prend le dessus injustement, et l'**auto-organisation** reflète plus fidèlement la réalité sémantique recherchée.

8.10.3. Sécurité et Biais

Au-delà des aspects d'intégration multimodale et d'optimisation, un **DSL** (Deep Synergy Learning) appliqué à des **flux** (texte, images, audio, etc.) doit aussi s'interroger sur la **sécurité** et la **neutralité** des données traitées. Les **biais** présents dans les corpus textuels, visuels ou sonores risquent de **déformer** la formation des clusters (donnant lieu à des regroupements stéréotypés ou injustes), tandis que des flux malveillants (ex. un **fake audio**) peuvent compromettre la **confiance** dans la synergie calculée.

8.10.3.1. Biais dans le texte (ou les images) → clusters déformés

Lorsqu'un SCN (Synergistic Connection Network) travaille sur un **DSL** (Deep Synergy Learning) multimodal, il s'appuie sur des **embeddings** ou des **représentations** sub-symboliques issus de **corpus** divers (textuels, visuels, etc.). Or, si les **données** textuelles ou visuelles sont entachées de **biais** (culturels, genres, ethniques...), la **synergie** calculée $S(i, j)$ se retrouve biaisée. Le **DSL**, au fil de l'auto-organisation, construit alors des **clusters** potentiellement **déformés**, ancrant les stéréotypes du corpus dans la structure même du réseau. La présente section (8.10.3.1) explore la manière dont ces biais se manifestent et leurs conséquences sur la formation de **clusters**.

A. Origine des Biais dans les Données

De nombreux corpus textuels (provenant de l'internet, d'archives, de données historiques) véhiculent des **stéréotypes** ou des **asymétries**. Par exemple, "médecin" pourrait être corrélé à un genre masculin, "infirmière" à un genre féminin, ou d'autres liens discriminants relatifs aux métiers, aux groupes ethniques, etc. D'un point de vue **mathématique**, les **embeddings** (Word2Vec, GloVe, BERT...) enregistrent ce déséquilibre, aboutissant à des **similarités** entre mots reflétant des biais latents.

Dans les bases d'images (pour la **vision**), un **déséquilibre** peut se manifester dans la représentation de certains groupes (ex. plus d'images d'hommes que de femmes dans un métier donné, ou d'un groupe ethnique particulier). Un algorithme de **reconnaissance** ou d'**embedding** entraîné sur ces données assimile alors des notions partielles ou sur-spécifiques, impactant la **synergie** $S(\text{img}_i, \text{img}_j)$ et la façon dont il associe images à concepts.

Dans un SCN multimodal, ces biais “visuels” peuvent **déformer** l’idée qu’a le réseau de la diversité d’apparences, menant à des confusions ou à un alignement trop strict avec l’échantillonnage du dataset.

B. Impact sur la Formation des Clusters

Le **DSL** fonctionne en renforçant $\omega_{i,j}$ lorsque $S(i,j)$ est jugé élevé. Mais si $S(i,j)$ est déjà corrompu par un **biais** de l’embedding (par ex. “femme” corrélé à “infirmière” beaucoup plus qu’à “médecin”), alors le SCN va **créer** ou **stabiliser** un cluster qui associe “femme” et “infirmière”, tout en écartant “femme” de “médecin”. De même, le cluster “médecin-homme” peut se **cristalliser**, reproduisant un stéréotype.

À mesure que la **dynamique** DSL se déroule, ces “petits biais” dans S s’accumulent et **créent** des macro-nœuds reflétant la distribution biaisée du corpus plutôt que la **réalité** qu’on voudrait représenter. Le SCN peut ainsi *sur-associer* “ingénieur” à “homme” ou “voile” à une ethnie spécifique. Ces **déformations** sont d’autant plus graves qu’elles peuvent renforcer d’autres liaisons (effet de rétroaction), menant à des **clusters** encore plus stéréotypés (boucle d’amplification).

C. Exemples

Imaginons un SCN multimodal image–texte où “chien” est très souvent décrit par un mot “dog” associé à un adverbe “cute”. Le SCN finira par sur-pondérer $\omega_{\text{dog}, \text{cute}}$. Mais si le corpus n’a que peu d’exemples de “chien” “agressif”, l’association “dog–aggressive” reste faible, faussant la pluralité possible des chiens. Pire, s’il y a un biais “cat = female, dog = male” dans le corpus, le SCN entérine des **clusters** genrés d’animaux.

Dans un flux de visages, le SCN peut surreprésenter un certain groupe ethnique dans le cluster “client souriant”, en ignorant d’autres facettes faute d’exemples ou d’embeddings adaptés. Les pondérations ω se hiérarchisent alors selon un prisme biaisé, privant le réseau d’une vision équitable.

D. Approches pour Déetecter/Atténuer ces Biais

On peut **analyser** la matrice $\{\omega_{i,j}\}$ ou les macro-clusters issus du SCN en cherchant des indicateurs de regroupement stéréotypé. D’un point de vue **mathématique**, on peut définir un **indice** de concentration d’un certain groupe (genre, ethnie, etc.) dans un cluster particulier et voir s’il diverge de la distribution globale.

Une option consiste à **rebaïsser** les scores $S(i,j)$ susceptibles de véhiculer un stéréotype, en introduisant un terme de “debias” :

$$S'(i,j) = S(i,j) - \Delta_{\text{bias}}(i,j).$$

Si \mathcal{E}_i et \mathcal{E}_j contiennent des attributs genrés, on peut injecter un correctif statistique pour diminuer la **corrélation** artificielle.

À un niveau plus **amont**, on peut **réentraîner** l’embedding (image, texte) sur un corpus plus équilibré ou recourir à des techniques d’**augmentations** de données pour réduire le biais au stade de l’ingestion. Cela diminue le besoin de correction tardive dans le SCN.

E. Enjeux Éthiques et Sociétaux

Le **biais** dans un SCN multimodal n’est pas qu’un souci technique, il peut consolider des représentations discriminantes, perpétuer des stéréotypes, influencer des **décisions** injustes (cf. chap. 8.10.3.2 peut-être). Sur le plan mathématique, le SCN n’est pas en tort s’il s’appuie sur un $S(i,j)$ biaisé par l’embedding, mais d’un point de vue **responsabilité** et **équité**, il revient à l’ingénierie du DSL de **déetecter**, **comprendre** et **corriger** ces défauts.

Conclusion

Des **biais** dans le texte ou l’image affectent directement la **synergie** S , ce qui amène le **SCN** à **former** des **clusters** déformés (stéréotypés, incomplets). Les liens $\omega_{i,j}$ se basent sur des embeddings ou des distributions initiales qui, s’ils ne sont pas neutres, induisent des regroupements centrés sur des stéréotypes, voire discriminants. Le **DSL** peut **exacerber** ces biais. Plus un stéréotype s’infiltre dans S , plus la dynamique renforce des clusters cohérents avec ce stéréotype. D’un point de vue mathématique, l’émergence de tels **macro-nœuds** repose sur une **structure** de données

imparfaite. D'un point de vue sociétal ou éthique, il importe de **déetecter** et **corriger** ces biais dans la synergie, en affaiblissant ou réévaluant les liens affectés, et/ou en **nettoyant** la distribution d'apprentissage pour s'assurer d'une plus juste représentation.

8.10.3.2. Sécurité : si un flux audio est “fake”, confusion potentielle ?

Dans un **SCN** (Synergistic Connection Network) multimodal où cohabitent des **flux** visuels (vidéo), **audio** (enregistrements sonores) et **textes** (sous-titres, annotations), la **qualité** et l'**authenticité** de chaque flux conditionnent la **fiabilité** de la **synergie** $S(i, j)$. Un flux **audio** “fake” (ex. deepfake vocal) risque de **leurrer** le SCN, car il peut être *artificiellement* cohérent avec un flux visuel ou textuel. Cette section (8.10.3.2) explore comment un audio “falsifié” peut semer la **confusion** dans la structure ω du DSL et quelles **approches** sécuritaires on peut envisager pour mitiger ce risque.

A. Contexte : le “fake audio” dans un DSL multimodal

Un **fake audio** consiste en un signal sonore **généré** ou **modifié** (ex. imitation d'une voix, ajout de propos inexistant) dans le but de **duper** un système. Sur le plan **mathématique**, l'embedding \mathbf{v}_{fake} d'un tel flux peut **ressembler** à celui d'un audio légitime $\mathbf{v}_{\text{legit}}$ si l'algorithme de deepfake est efficace. Le **DSL**, dans sa phase de calcul $S(\text{aud}, \text{vid})$ ou $S(\text{aud}, \text{txt})$, peut alors **renforcer** $\omega_{\text{fake}, \text{autres}}$ à un haut niveau, croyant que la correspondance est réelle.

Si le **SCN** adopte la règle de mise à jour

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

et que l'audio “fake” est **superficielement cohérent** avec un flux vidéo (ex. un mouvement labial), la similarité $S(\text{fake}, \text{vid})$ sera jugée élevée et les liaisons ω s'intensifient. Cela conduit le SCN à intégrer cette entité audio “fake” au sein d'un **cluster** multimodal (e.g., un macro-nœud représentant la scène ou le locuteur). La **structure** du DSL s'en voit altérée, ce qui peut détériorer la **reconnaissance** d'événements, car un flux sonore trompeur est pris en compte. Cela peut également induire une **fausse** correspondance entre la vidéo et le contenu sonore, tout en contaminant le flux textuel, par exemple si l'audio falsifié est aligné à des sous-titres fallacieux.

B. Mécanismes : confusion potentielle dans la synergie

Le DSL renforce $\omega_{\text{fake}, \text{legit}}$ dès lors que $S(\text{fake}, \text{legit})$ est élevé, ce qui peut arriver si l'embedding audio “fake” \mathbf{v}_{fake} se trouve **proche** de $\mathbf{v}_{\text{locuteur}}$ dans l'espace des features (ex. même timbre, même intonation). Cela **leurre** la règle DSL en faisant croire à un score de similarité fort. L'**auto-organisation** se base sur un “garbage in” partiel, provoquant un cluster “faux” où la voix simulée est acceptée comme partie légitime de la scène.

Une fois que $\omega_{\text{fake}, \text{locuteur}}$ est fort, l'entité audio “fake” reçoit plus de **validité** dans le SCN, influençant la construction de macro-nœuds (chap. 8.6). Si, par exemple, ce locuteur est lié à un sous-titre **texte**, l'audio “fake” peut s'immiscer dans un cluster “Texte–Image–Audio” consolidé. Il en résulte une “**confusion**”. Le flux “fake” se retrouve **validé** par le réseau, induisant potentiellement des **actions** erronées en aval (ex. classification trompée, décision erronée).

C. Pistes de Sécurisation et Contrôle

Une première **défense** consiste à exiger plus qu'une simple correspondance audio–vidéo. On multiplie les **tests** de cohérence (contrôle labial précis, tonalité vocale), ou on confronte le flux audio à d'autres indicateurs (cf. recuit simulé, chap. 7.3). Si une entité “fake” ne parvient pas à maintenir une cohérence **simultanée** avec d'autres flux ou d'autres moments du réseau, la synergie chute, évitant la consolidation du lien ω .

Sur le plan **mathématique**, on peut intégrer un module “**score** de suspicion” $\text{suspect}(\mathcal{E}_{\text{fake}}) \in [0,1]$ dans la fonction de **synergie** :

$$S'(\text{fake}, j) = (1 - \text{suspect}(\text{fake})) \times S(\text{fake}, j).$$

Si on détecte (via un algorithme externe) un fort risque de deepfake, $\text{suspect}(\text{fake}) \approx 1$, annihilant la pondération ω . Cela **empêche** l'entité falsifiée de piéger le SCN.

Si l'on relève une incohérence à un certain stade (ex. la vidéo indique un locuteur différent de ce que l'audio signale), on peut **amplifier** l'inhibition (chap. 7.4) dirigée contre ce flux audio suspect. Dans ce cas, le DSL diminue rapidement les liaisons $\omega_{fake,..}$.

D. Conclusion

Dans un **SCN** multimodal, un flux audio “fake” peut **confondre** la synergie S et semer la **confusion** dans la **structure** ω . Sur le plan **mathématique**, c'est la conséquence directe d'une **similitude** apparente entre \mathbf{v}_{fake} et des entités légitimes \mathbf{v}_{real} . Les liaisons ω se renforceront, validant involontairement le flux trompeur. Au niveau **sécurité**, il est donc nécessaire de :

- **Déetecter** activement les deepfakes,
- **Réduire** la force de ω si la suspicion est élevée,
- **Vérifier** la cohérence cross-modale pour ancrer un flux audio dans la réalité (contrôle des alignements labiaux, du contexte, etc.).

Dans un scénario plus vaste (chap. 8.10.4 sur robustesse), ces **contrôles** anti-fake garantissent que le DSL ne devienne pas un vecteur de contamination, consolidant un flux sonore mensonger et induisant le SCN en erreur.

8.10.4. Recherche Future

L'exploration du **DSL** (Deep Synergy Learning) multimodal ne se limite pas aux démos ou prototypes actuels ; il ouvre un vaste champ pour des **déploiements** plus ambitieux, tant en volume de données qu'en diversité sensorielle. Dans cette sous-section (8.10.4.1), nous soulignons la **perspective** d'appliquer le DSL à des **datasets massifs**, comportant potentiellement des **millions** de vidéos ou de documents, et d'y gérer des **synergies** à une échelle inédite.

8.10.4.1. Appliquer le DSL multimodal à de très grands datasets (millions de vidéos / documents)

Dans un **SCN** (Synergistic Connection Network) mis en œuvre pour un **DSL** (Deep Synergy Learning) multimodal, le **nombre** d'entités $\{\mathcal{E}_i\}$ peut exploser dès qu'on envisage des **datasets** à l'échelle de millions (ou dizaines de millions) d'éléments, par exemple dans la gestion de **vidéos** massives (YouTube, TikTok) ou de **documents** textuels volumineux. Cette section (8.10.4.1) aborde :

- Les **défis** de l'échelle et de la complexité,
- Les **infrastructures** HPC (High-Performance Computing) ou de calcul distribué qui s'imposent,
- Les **approches** hybrides et approximatives permettant de maintenir la logique DSL sans être submergé par un coût $O(n^2)$.

A. Échelle et Complexité

Lorsque l'on traite des **millions** de vidéos (ou documents), on se retrouve avec un **nombre** d'entités $n \approx 10^6$, voire plus. Le **SCN** stocke $\{\omega_{i,j}\}$, et le nombre de paires (i,j) peut monter à $O(n^2) \approx 10^{12}$. Cette **matrice** ω est impossible à manipuler pleinement en mémoire standard, et la mise à jour DSL en $O(n^2)$ par itération devient irréalisable.

Pour éviter cet écueil, on adopte des stratégies de **sparsification** (k plus proches voisins, gating de clusters) qui **limitent** la mise à jour $\omega_{i,j}$ aux seules paires (ou sous-blocs) jugées pertinentes. Les sections précédentes (8.10.1.2) évoquent ces heuristiques. Sur le plan **mathématique**, cela ramène le coût à $O(n k)$ ou $O(n \log n)$ selon l'indexation, plus gérable pour de larges n .

B. Infrastructure et HPC

Traiter des **millions** de vidéos/documents implique souvent d'avoir accès à des **clusters** HPC (High-Performance Computing) ou du **cloud** distribué. Cela requiert un **partitionnement** de la grande **matrice** ω :

- On divise l'ensemble d'entités $\{1, \dots, n\}$ en $\{\mathcal{V}_1, \dots, \mathcal{V}_m\}$.
- Chaque partition \mathcal{V}_p gère localement $\omega_{i,j}$ pour $(i, j) \in \mathcal{V}_p$.
- Les paires $(i \in \mathcal{V}_p, j \in \mathcal{V}_q)$ supposent une **synchronisation** inter-bloc (messagerie, verrous épisodiques).

D'un point de vue **théorique**, on peut modéliser cela comme des **sous-graphes** partiels, rassemblés périodiquement. Du fait de la nature **distribuée**, la mise à jour DSL devient asynchrone ou semi-synchrone, et la convergence doit être considérée dans un cadre distribué (chap. 7.5 sur la parallélisation).

Une autre manière de décomposer le traitement est de **segmenter** (batcher) la base en sous-lots plus petits (ex. 10^4 entités). On applique la mise à jour DSL localement sur chaque lot, puis on fusionne les résultats ou on effectue une étape de **raccord**. Cela introduit une forme de **mini-batch** auto-organisé, analogue à ce qui se pratique dans l'apprentissage profond classique.

C. Approches Hybrides

Si le dataset provient d'un **flux** continu (ex. nouveau contenu vidéo chaque jour), on mélange :

- La **segmentation** en mini-lots (batches),
- La **recherche** de voisins (k-NN) restreinte,
- L'**insertion incrémentale** (voir chap. 9.1 sur la dynamique en flux).

Ce dispositif garantit qu'on ne fasse **jamais** d'itération $O(n^2)$ complète, mais plutôt un ensemble d'opérations locales $O(n k)$.

Une pratique courante consiste à définir un **seuil** θ dans l'espace d'embeddings de la modalité., tel que

$$S(i, j) = \begin{cases} \text{compute}, & \text{si } \|\mathbf{x}_i - \mathbf{x}_j\| \leq \theta, \\ 0, & \text{sinon.} \end{cases}$$

Cela évite de calculer des distances $\|\mathbf{x}_i - \mathbf{x}_j\|$ pour toutes paires, et impose qu'on stocke $\omega_{i,j}$ seulement si $\|\mathbf{x}_i - \mathbf{x}_j\| \leq \theta$. Dans la pratique, on peut user de techniques d'**approximate nearest neighbor** (ANN, hashing, etc.) pour localiser les candidats (j) dans une boule $\|\mathbf{x}_i - \mathbf{x}_j\| \leq \theta$.

D. Potentiel en Recherche

Du point de vue **théorique**, on peut étudier le comportement d'un SCN à **millions** d'entités en s'inspirant d'**objets** comme les **graphons** en théorie des graphes à grande échelle, ou en adoptant des arguments de **limite** $n \rightarrow \infty$. On examinerait alors la **formation** de clusters comme une transition de phase, avec un niveau de similarité moyen.

On se demande aussi comment, mathématiquement, la **fusion** des canaux (image–audio–texte) demeure stable ou cohérente quand n grossit. S'il existe un flux surreprésenté, peut-il dominer ? Des formalismes de grande dimension montrent que certains canaux peuvent saturer la matrice ω si on ne prévient pas la surexposition.

E. Exemple de Scénario

Supposons qu'on veuille **analyser** 10^6 vidéos courtes, par exemple des extraits de **TikTok** ou **YouTube Shorts**. On **extrait** pour chaque vidéo un embedding visuel $\mathbf{x}_i^{(\text{vid})}$, un embedding audio $\mathbf{x}_i^{(\text{aud})}$ et éventuellement un embedding textuel $\mathbf{x}_i^{(\text{txt})}$. On se retrouve ainsi avec $n = 10^6$ entités vidéo, audio ou texte, voire plus. Afin d'éviter une complexité de $O(n^2) = 10^{12}$ calculs par itération, une **heuristique** de type k-NN est appliquée, restreignant les connexions de

chaque entité à ses $k = 100$ plus proches voisins. La mise à jour est alors distribuée sur un cluster HPC, chaque nœud traitant un sous-bloc, tandis que la partie inter-bloc des pondérations ω est synchronisée régulièrement. Après un certain nombre d'itérations, la structure **auto-organisée** révèle des **clusters** ou **macro-nœuds** $\{1, \dots, C\}$ qui regroupent des vidéos aux thématiques similaires, telles que “sports”, “comédie” ou “vlog personnel”, identifiables par la convergence locale des pondérations ω .

Conclusion

L'application d'un **DSL** multimodal à des **datasets** très larges (par ex. millions de vidéos) soulève des **problèmes d'échelle et de complexité** :

- **Stocker** ω en $O(n^2)$ devient impossible,
- Les mises à jour en $O(n^2)$ par itération ne sont plus réalisables,
- Il faut **distribuer** la charge et/ou **limiter** la mise à jour ω à un sous-ensemble de paires (k-NN, threshold).

En **infrastructure HPC**, on peut **partitionner** le SCN et synchroniser les sous-blocs, usant de mini-batches ou d'approximations. Sur le plan **mathématique**, ces méthodes garantissent que la **philosophie** du DSL (auto-organisation par renforcement local des synergies) demeure, mais sous une forme **sparse** et **distribuée**. Les perspectives incluent l'étude de **limites** asymptotiques ($n \rightarrow \infty$) et des **techniques** adaptatives, ouvrant la voie à des **réseaux** DSL “géants” capables d'auto-organiser des corpus multimodaux massifs.

8.10.4.2. Fusion Multi-Sensorielle en Robotique plus Avancée (Capteurs LiDAR, etc.)

La **fusion** de multiples **capteurs** (caméras, audio, LiDAR, etc.) constitue un volet crucial de la **robotique** moderne, où la diversité de sources (capteurs de profondeur, inertIELS, radars, etc.) accroît la **richesse** des informations mais soulève des **défis** d'intégration. Au sein d'un **SCN** (Synergistic Connection Network) piloté par un **DSL** (Deep Synergy Learning), les **entités** issues de capteurs variés peuvent être reliées par des **pondérations** ω , ajustées en continu par la **synergie** S . La présente section (8.10.4.2) s'attache plus spécifiquement à la **fusion** impliquant un **capteur LiDAR** (Light Detection and Ranging), soulignant ses apports et la manière dont un SCN multimodal peut en tirer parti pour une **compréhension** plus fine de l'environnement.

A. Rappels et Enjeux de la Fusion LiDAR

Le **LiDAR** envoie des impulsions laser pour mesurer la distance des obstacles via leur temps de vol. On obtient ainsi un **nuage de points** 3D décrivant la géométrie de l'environnement autour du robot. Sur le plan **mathématique**, on peut représenter chaque **scan** LiDAR comme un ensemble $\{\mathbf{p}_k\} \subset \mathbb{R}^3$. Pour alimenter un **SCN**, il est possible de **définir** une **entité** \mathcal{E}_i comme un cluster ou un voxel parmi ces points, en fonction d'une segmentation ou d'une agrégation 3D spécifique. Une fois cette entité formée, on lui associe des **features** pertinentes telles que la **position moyenne**, la **normale** dominante ou encore la **densité** de points, permettant ainsi d'enrichir la représentation et d'optimiser la synergie entre les différentes entités du réseau.

Un capteur LiDAR se révèle **complémentaire** des caméras (2D) en fournissant un **relief** 3D précis, tandis que l'image apporte la **couleur** ou la **texture**.

Le **DSL** (Deep Synergy Learning) opère sur un **réseau** $\{\omega_{i,j}\}$, où $\omega_{i,j}$ relie deux **entités** (par ex. segment LiDAR vs. patch visuel). La **synergie** S quantifie leur cohérence spatiale ou sémantique (ex. un nuage 3D localisé devant la caméra correspond à la forme perçue en 2D). L'**auto-organisation** $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ renforce ou affaiblit ces liens en continu, permettant au robot de **consolider** une représentation multimodale plus fiable.

B. Modélisation Mathématique de la Synergie LiDAR + Autres Capteurs

Plutôt que de traiter directement des milliers ou millions de points **LiDAR**, on effectue une **segmentation** 3D à l'aide d'algorithmes comme **DBSCAN** ou **RANSAC**, permettant d'identifier des volumes sous forme de clusters ou de surfaces homogènes. Une fois cette segmentation réalisée, on procède à une **agrégation** en entités $\{\mathcal{E}_i^{(LiDAR)}\}$, où

chaque entité représente un **objet partiel** ou un **voxel** volumique, facilitant ainsi l'analyse et l'exploitation des données au sein du SCN.

Chacune de ces entités \mathcal{E}_i dispose d'un vecteur $\mathbf{x}_i \in \mathbb{R}^d$ (features géométriques). Un algorithme plus fin peut extraire la **normale moyenne**, la **couleur estimée** (si un alignement avec la caméra est disponible), etc.

Si un robot embarque à la fois un LiDAR et une caméra, on peut définir :

$$S(\mathcal{E}_{\text{LiDAR}}, \mathcal{E}_{\text{cam}}) = \alpha \mathcal{C}_{\text{proj}}(\mathcal{E}_{\text{LiDAR}}, \mathcal{E}_{\text{cam}}) + \beta \mathcal{C}_{\text{color}}(\mathcal{E}_{\text{LiDAR}}, \mathcal{E}_{\text{cam}}) + \dots$$

- $\mathcal{C}_{\text{proj}}$ estime la **cohérence** spatiale en vérifiant que la projection 3D du cluster LiDAR vers l'image coïncide avec la forme 2D détectée.
- $\mathcal{C}_{\text{color}}$ compare éventuellement la **colorimétrie** si on a mappé la texture de la caméra sur le nuage LiDAR, etc.

Le SCN exploitera cette synergie pour **renforcer** ω entre entités “correspondantes” (même objet capté par la caméra et le LiDAR) et **éliminer** celles qui ne correspondent pas (faible S).

On peut étendre la logique à d'autres **capteurs**, en intégrant notamment des sources **audio**, où l'emplacement sonore est estimé par **beamforming**, ou en exploitant la position 3D d'un événement capté par **LiDAR**. De même, les capteurs **inertiels (IMU)** permettent d'évaluer la **pose** d'un robot et d'en déduire un **alignement spatio-temporel**, modélisé par une synergie $S(\text{LiDAR}_{t+1}, \text{LiDAR}_t)$ qui exprime la continuité entre deux instants successifs. Dans chaque cas, le **DSL** établit et ajuste dynamiquement les **liaisons** ω afin de renforcer la **cohérence** du réseau et d'optimiser l'agrégation des signaux issus de diverses modalités.

C. Avantages et Problématiques de la Fusion LiDAR + DSL

L'**auto-organisation** DSL procure une **flexibilité** en évitant une procédure figée de fusion. Au lieu d'une projection LiDAR vers caméra prédéfinie, le **réseau** ajuste dynamiquement ω si la configuration spatiale évolue ou si une partie du nuage LiDAR ne trouve pas d'équivalent visuel. Les entités \mathcal{E}_i LiDAR peuvent être partiellement **isolées** (faible ω vers les caméras), ou au contraire, fortement liées si l'**image** confirme l'objet 3D.

Inévitablement, un **nuage** LiDAR volumineux (plusieurs dizaines de milliers de points par scan) engendre beaucoup d'entités dans le SCN. Cela conduit à privilégier des approches telles que la **segmentation** ou la **voxelisation**, permettant de réduire la **granularité** des données et d'optimiser leur traitement. Une autre stratégie repose sur l'usage d'**heuristiques** basées sur les **k plus proches voisins** (k-NN), comme discuté en détail dans le **chapitre 8.10.1.2**. L'objectif est d'éviter un calcul exhaustif des pondérations ω entre chaque segment issu du **LiDAR** et l'ensemble des segments issus de la **caméra**, ce qui serait trop coûteux en termes de complexité computationnelle.

En **robotique**, le LiDAR scanne l'environnement en continu, donnant naissance à des entités $\{\mathcal{E}_{(\text{LiDAR}, t)}\}$. Le DSL agit donc de manière **incrémentale** (chap. 9.1), chaque nouveau scan actualisant ω . Sur le plan **mathématique**, la mise à jour $\omega_{(n+1), j} \leftarrow \omega_{(n+1), j} + \dots$ s'applique à un ensemble restreint de voisins si on adopte une approche k-NN ou un **gating**.

D. Extensions : Surfaces 3D et Sémantique Avancée

Le LiDAR est parfois converti en **maillage** (mesh) ou en **surface** polygonale. Dans un SCN, on peut représenter ces **surfaces** comme des entités plus haut niveau, facilitant la correspondance géométrique avec la caméra. La **synergie** $S(\text{mesh}, \text{cam})$ se définit alors via la superposition 2D/3D.

Par ailleurs, le robot peut disposer de **modèles** de reconnaissance (objets, classes sémantiques) alimentant le SCN d'un volet “symbolique” (chap. 8.7.4). On associe à chaque **cluster** LiDAR un label potentiel (ex. “voiture stationnée”). Le DSL vérifie la **cohérence** de ce label via la caméra ou la carte textuelle.

Conclusion

Dans le cadre d'une **robotique** avancée, la **fusion** multi-sensorielle intégrant **LiDAR**, caméras, audio, capteurs inertiels, etc. se prête bien à un **DSL** (Deep Synergy Learning) :

- **Chaque** capteur engendre des **entités** (clusters 3D, patchs images...),

- **Synergie** S mesure la coïncidence spatiale, sémantique ou temporelle,
- L'**auto-organisation** du SCN (ω) renforce ce qui est cohérent et isole les données aberrantes.

Sur le plan **mathématique**, un tel SCN applique la règle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta[S(i,j) - \tau \omega_{i,j}(t)]$ à chaque lien jugé pertinent, en flux continu. Les **cahiers** de charges particuliers (volume important de points LiDAR, scans successifs, multi-canal) demandent des **heuristiques** (k-NN, gating) pour gérer la complexité en temps réel. On obtient alors une **vision** plus riche et plus flexible de l'environnement, résiliente à des lacunes ou à des bruits dans un capteur isolé.

8.11. Conclusion et Ouverture

Après avoir parcouru ce chapitre 8 consacré au **DSL multimodal**, nous arrivons maintenant à la **conclusion** et aux pistes d'ouverture. L'objectif était de montrer comment le **Deep Synergy Learning** s'étend à des flux variés (images, audio, texte...), tout en préservant la philosophie d'**auto-organisation** au sein du **SCN** (Synergistic Connection Network). Dans ce qui suit (8.11.1 et 8.11.2), nous récapitulons les points marquants du chapitre, puis nous anticipons les liens avec les chapitres ultérieurs (9 et 10), avant de souligner la valeur générale du DSL multimodal (8.11.3).

8.11.1. Récapitulatif du Chapitre

8.11.1.1. On a défini comment le DSL peut s'étendre à plusieurs modalités (image, audio, texte)

Au terme de l'exploration menée dans ce **chapitre 8**, nous avons montré que le **DSL** (Deep Synergy Learning), jusqu'à présent appliqué à des environnements mono-modaux (par exemple, seulement images ou seulement texte), peut être **élargi** à des cadres où **plusieurs modalités** — visuelles, auditives, textuelles, voire d'autres types de capteurs — coexistent et sont traitées au sein d'un **SCN** (Synergistic Connection Network) unifié. Cette extension repose sur plusieurs aspects fondamentaux garantissant la robustesse et l'adaptabilité du modèle. Tout d'abord, elle intègre des **fonctions de synergie** spécifiques ou hybrides, permettant d'évaluer la **pertinence** et la **compatibilité** entre des entités provenant de différents canaux, qu'il s'agisse de relations image–image, image–texte ou encore audio–texte. Ensuite, la **dynamique** du DSL suit une règle de mise à jour uniforme, définie par

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)]$$

indépendamment de la modalité des entités concernées. Si la synergie $S(i,j)$ est jugée élevée, alors la pondération $\omega_{i,j}$ se trouve renforcée, tandis qu'en cas de faible synergie, elle décroît progressivement. Enfin, l'ensemble du réseau conserve une **cohérence** globale grâce aux mécanismes d'**inhibition**, détaillés dans le **chapitre 7.4**, ainsi qu'à la possibilité d'**insertion** incrémentale, discutée dans le **chapitre 9**. De plus, des techniques de **calibrage** des embeddings, présentées dans le **chapitre 8.10.2**, permettent d'assurer que les différentes modalités restent alignées et comparables au sein du SCN.

Cette conclusion met en évidence la **généralité** de l'approche DSL. Son cadre synaptique local, basé sur l'évolution des **pondérations** ω et la mesure de **synergie** S , s'adapte de manière **naturelle** à des **sources** de données variées. L'un des aspects clés de cette flexibilité réside dans la **définition appropriée** de la **fonction** $S(\mathcal{E}_i, \mathcal{E}_j)$, qui doit refléter la correspondance ou la distance entre entités, même lorsqu'elles appartiennent à des **espaces** ou **dimensions** fondamentalement dissemblables.

A. Principe d'Intégration Multimodale

L'**intégration multimodale** repose sur le fait qu'un **SCN** ne se limite plus à des entités issues d'un **espace unique** \mathbb{R}^d . Désormais, les entités **visuelles** $\mathcal{E}_i^{(img)}$ sont représentées par un **embedding visuel** $\mathbf{v} * i \in \mathbb{R}^{d*img}$, tandis que les entités **sonores** $\mathcal{E}_j^{(aud)}$ sont décrites par un **embedding acoustique** $\mathbf{a} * j \in \mathbb{R}^{d*aud}$. De même, les entités **textuelles** $\mathcal{E}_k^{(txt)}$ possèdent une représentation linguistique sous la forme d'un **embedding sémantique** $\mathbf{t} * k \in \mathbb{R}^{d*txt}$. D'autres modalités, telles que les **capteurs LiDAR** (voir chap. 8.10.4.2) ou les **représentations symboliques** (voir chap. 8.7.4), peuvent être incorporées à cette structure.

Le **DSL** organise alors un **réseau de pondérations** $\omega_{i,j}$ qui connecte ces entités, sans nécessiter un **alignement strict** de leurs espaces respectifs. Pour ce faire, il utilise des **fonctions de synergie** adaptées à chaque modalité, telles que la **similarité cosinus**, la **distance exponentielle**, ou encore une **co-occurrence probabiliste**. En présence de plusieurs modalités combinées, il est possible de recourir à un **assemblage pondéré** sous la forme d'une combinaison linéaire $\alpha, S_{vis} + \beta, S_{aud}$, permettant ainsi de moduler l'impact de chaque **canal d'information** selon son importance relative au sein du réseau.

Sur le plan **mathématique**, la même règle de descente

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(\mathcal{E}_i, \mathcal{E}_j) - \tau \omega_{i,j}(t)]$$

assure que, si plusieurs modalités **confirment** une association (par ex. l'image d'un chat, l'audio "meow", le mot "cat"), le **score** S en devienne élevé, renforçant $\omega_{i,j}$. Ainsi, un **cluster** auto-organisé se forme, réunissant entités {img_chat, audio_miaulement, texte_cat}. D'autres liaisons (incohérentes) finissent par s'**étioyer** (pondérations retombant vers 0).

B. Mise en Œuvre au Niveau du SCN

Le SCN demeure une **matrice** $\{\omega_{i,j}\}_{1 \leq i,j \leq n}$. La "nouveauté" réside dans le fait que $S(i,j)$ prend en compte l'identifiant de la modalité. Lorsque deux entités appartiennent au même canal, comme image–image ou audio–audio, la synergie S est calculée selon les méthodes classiques d'un **DSL mono-modal**, en fonction de la distance dans l'espace des embeddings ou d'autres critères de similarité propres à la modalité. En revanche, lorsqu'il s'agit d'une comparaison **cross-modal** (par exemple, image–texte ou audio–texte), on utilise une fonction S spécifique et adaptée à cette correspondance, telle que

$$S(\text{image}_i, \text{texte}_j) = f\left(\cos\left(\mathbf{v}_i^{(\text{img})}, \mathbf{v}_j^{(\text{txt})}\right)\right)$$

ou bien un mélange convexe, comme décrit dans la section **8.10.2.2**, lorsque plusieurs descripteurs sont disponibles pour affiner la mesure de synergie entre les modalités.

Une fois la mise à jour enclenchée, le **DSL** repère **naturellement** les associations fréquentes ou récurrentes. Un *macro-nœud* se construit progressivement, regroupant un sous-ensemble d'images, de segments audio et de tokens textuels qui partagent une forte synergie. Sur le plan **algorithmique**, on finit par voir un **bloc** dans la matrice ω où $\omega_{i,j}$ est élevé, indiquant une synergie inter-modalité solide (ex. "chien + aboiement + mot dog + mot puppy").

C. Gestion des Conflits et Incohérences

Il arrive que le SCN détecte une synergie contradictoire (ex. un segment audio "oiseau" mal aligné sur une image "voiture"). Si la **valeur** S reste faible ou fluctuante, la pondération $\omega_{i,j}$ ne se consolide pas. L'**auto-organisation** agit comme un **filtre** inhibant la croissance de liens incohérents.

Les règles d'**inhibition** (chap. 7.4) permettent de limiter la **prolifération** de liaisons entre modalités trop hétérogènes. S'il s'avère qu'un flux audio "n'a rien à voir" avec l'image, $\omega_{i,j}$ se voit réduit, évitant la formation d'un cluster erroné.

D. Pertinence pour l'Analyse et la Classification

Exemples d'applications

- **Indexation multimédia** : on peut ranger ou annoter des **vidéos** selon les sons correspondants et les mots-clés détectés. Le SCN classe naturellement ce qui converge (image + sous-titres + audio).
- **Recherche cross-modal** : on cherche dans des documents la correspondance (texte–image). Un SCN fusionne les entités, créant des macro-nœuds signifiant la parenté sémantique.
- **Clustering** auto-organisé : un ensemble de contenus (podcasts, transcriptions, images d'illustration) se groupent par thèmes émergents.

Au plan **théorique**, la **convergence** se comprend comme la minimisation d'une fonction d'énergie $J(\omega)$ (chap. 7.2), où l'on pèse la **synergie** $S(i,j)$ par $\omega_{i,j}$. Les entités dont la **similarité** (même modalité) ou **complémentarité** (modalités différentes, mais sémantiquement alignées) est forte forment un **minimum local** stable, c'est-à-dire un **cluster** consolidé dans la matrice $\{\omega_{i,j}\}$.

Conclusion

Nous avons ainsi **défini** comment le **DSL** peut :

- **S'étendre** naturellement à plusieurs modalités (image, audio, texte, etc.),

- **Conserver** la même dynamique de mise à jour ω (règle DSL), en y incorporant des **fonctions** S adaptées aux différents canaux,
- **Auto-organiser** les entités issues de modalités variées en **clusters** multimodaux, reflétant la convergence ou la cohérence cross-canaux.

L'**intégration** multimodale dans un **SCN** constitue une **alternative** souple aux pipelines rigides, en évitant l'imposition d'un alignement strict entre canaux. Plutôt que de contraindre chaque modalité à s'inscrire dans une structure préétablie, le réseau de pondérations ω permet des connexions dynamiques entre les différentes modalités en fonction de leur **pertinence** et de leur **cohérence**.

Cette **flexibilité** présente plusieurs avantages majeurs. D'abord, elle assure une **robustesse** accrue, car les liaisons incohérentes s'atténuent progressivement, tandis que les synergies réellement significatives se renforcent naturellement.

Ensuite, elle permet une **capacité d'adaptation** essentielle à l'évolution du système, en autorisant l'insertion progressive de nouveaux flux multimodaux sans nécessiter de restructuration complète, comme détaillé dans la section 9.

Enfin, elle procure une **vision unifiée** de l'information, où les entités audio, visuelles et textuelles convergent pour former des clusters autour de concepts communs, garantissant ainsi une représentation plus cohérente et interconnectée des données.

Cette **généralisation** du DSL marque une étape vers la **fusion** et l'**apprentissage** non supervisé multi-canals, où l'**auto-organisation** s'exprime sur un périmètre plus vaste qu'un domaine unique, tirant parti de la diversité des signaux pour construire des **clusters** et des **macro-nœuds** aux fondements plus riches et plus transversaux.

8.11.1.2. On a vu la dynamique auto-organisée gérer des liens hétérogènes, créant des clusters multimodaux

Au fil du **chapitre 8**, nous avons exploré la façon dont un **DSL** (Deep Synergy Learning) — appliqué à un **SCN** (Synergistic Connection Network) — traite simultanément des entités provenant de **differentes modalités** (image, audio, texte, etc.) en s'appuyant sur la **même** dynamique ω . Le **résultat** essentiel est la capacité à gérer des **liens** hétérogènes tout en **faisant émerger** des **clusters** multimodaux, c'est-à-dire des sous-groupes où des entités de canaux dissemblables (vision, son, mots, etc.) se retrouvent rassemblées selon leur **synergie**. La présente section (8.11.1.2) récapitule les aspects fondamentaux de cette auto-organisation multimodale et insiste sur l'hétérogénéité des liens traités par la mise à jour DSL.

A. Hétérogénéité des Liens et Mesures de Synergie

Un **SCN** multimodal regroupe des **entités** $\{\mathcal{E}_i\}$ issues de canaux variés, par exemple :

- $\mathcal{E}_i^{(\text{img})}$ pour les **images**,
- $\mathcal{E}_j^{(\text{aud})}$ pour l'**audio**,
- $\mathcal{E}_k^{(\text{txt})}$ pour du **texte**,
- Ou encore d'autres capteurs (LiDAR, signaux biométriques...).

Les **liens** $\omega_{i,j}$ reliant ces entités diffèrent selon qu'on considère un **couple** image–image, image–texte, audio–texte, etc. La **synergie** $S(i,j)$ doit donc s'adapter à la **nature** des deux modalités. Par exemple, pour audio–audio, on pourra prendre un score de corrélation spectrale ; pour image–texte, un embedding cross-modal ; pour texte–texte, une similarité sémantique, etc.

Formellement, si l'on note \mathcal{M} l'ensemble des **modalités** (image, audio, texte, etc.), pour deux entités $\mathcal{E}_i^{(\alpha)} \in \alpha$ et $\mathcal{E}_j^{(\beta)} \in \beta$ (avec $\alpha, \beta \in \mathcal{M}$), la **fonction** $S^{(\alpha, \beta)}(i, j)$ mesure leur **pertinence**. On insère alors la règle DSL :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S^{(\alpha, \beta)}(i, j) - \tau \omega_{i,j}(t)].$$

Même si S varie d'une paire de modalités à l'autre, la **dynamique** de mise à jour ω (fondée sur η et τ) demeure identique et **auto-organisée**.

B. Émergence de Clusters Multimodaux

Grâce à cette **auto-organisation** hétérogène, le **SCN** peut relier un **segment audio** à une **image** s'ils sont cohérents (ex. le son “aboiement” correspond à la scène “chien”), ou un fragment **texte** décrivant la même scène. Ces liens ω se renforcent, menant à un **cluster** où cohabitent entités image–audio–texte. Ce cluster se révèle **multimodal**, traversant les différents canaux et reflétant une **unité sémantique**, un concept, une scène ou un objet.

Les liens ne s'appuient pas seulement sur la **similarité** (proches dans la même modalité), mais aussi sur la **complémentarité** (par ex. le texte et l'image se répondent, ou l'audio “coïncide” avec la vidéo). Le DSL autorise cette **mixité** car S peut intégrer des scores de **co-occurrence**, de **correspondance** spatiale ou temporelle, etc.

C. Conséquences sur la Dynamique Auto-Organisée

En prenant en compte plusieurs types de liens (image–image, image–audio, audio–texte, etc.), le **SCN** acquiert une **structure** plus dense, où chaque entité peut être reliée à d'autres d'une **modalité** différente. Les **clusters** qui en émergent ne sont plus purement “une classe d'images” ou “un ensemble de phrases”, mais **un** sous-groupe **multicanal** (chap. 8.10.4.2 sur la robotique, par exemple).

Cette capacité à gérer la **diversité** des liens donne au DSL plus de **robustesse**, car si un flux, comme l'audio, se dégrade ou se révèle insuffisant, la complémentarité d'autres flux, tels que le texte ou la vision, peut **soutenir** la cohérence du cluster. Par ailleurs, ce mécanisme favorise la **découverte** de nouvelles associations “long-courrier” (ex. on détecte qu'un son “type jazz” se recoupe souvent avec des images de “concert nocturne”), renforçant une connaissance globale du contenu.

Conclusion

La **dynamique auto-organisée** décrite dans le **DSL** est capable de **gérer** des liens **hétérogènes** (cross-modalités variées) et, ce faisant, de **créer** des **clusters** multimodaux cohérents. Les points clés sont :

- La **fonction** $S(i, j)$ s'adapte à la nature des entités (même ou différentes modalités),
- La **mise à jour** $\omega_{i,j}$ opère uniformément, autorisant l'auto-organisation à **fusionner** des canaux divers,
- Les **clusters** finaux peuvent traverser plusieurs modalités, illustrant la force du **SCN** à l'heure de traiter des données complexes et disparates.

Ce mécanisme **multimodal** ouvre la voie à des applications de plus grande envergure, car il permet de lier images, sons, textes, capteurs comme le LiDAR ou les signaux biologiques, tout en conservant la **philosophie** DSL, fondée sur l'**auto-organisation** des liens ω selon la synergie S . C'est précisément cette **universalité** de la règle DSL, quelle que soit la modalité, qui permet l'émergence de **clusters** transversaux, conférant au réseau une **représentation** holistique du contenu.

8.11.2. Liens vers Chapitres Suivants

L'exploration que nous avons menée dans ce **Chapitre 8** sur le **DSL multimodal**, traitant de la fusion entre la vision, le langage, les sons et d'autres modalités, trouve des prolongements naturels dans les **chapitres** qui suivent. Nous examinerons la capacité du **DSL** à évoluer dans des contextes encore plus dynamiques dans le **Chapitre 9**, ainsi que la mise en œuvre de **boucles de feedback** plus élaborées dans le **Chapitre 10**.

8.11.2.1. Chap. 9 : Évolutions Temps Réel et Apprentissage Continu

Après avoir établi, dans le **Chapitre 8**, les fondements d'un SCN (Synergistic Connection Network) étendu à plusieurs **modalités** (image, audio, texte, etc.), la suite logique consiste à aborder le cas d'un **environnement** en constante évolution, où de **nouvelles** entités multimodales apparaissent (ou d'anciennes disparaissent). Le **Chapitre 9** traite précisément des mécanismes d'**apprentissage continu**, d'**insertion** de nouvelles données, et de **dynamique** jamais totalement figée dans le temps. Cette section (8.11.2.1) introduit la transition vers ce prochain chapitre en rappelant les grands enjeux et l'esprit des méthodes qui y seront discutées.

A. Flux Dynamiques et Contexte Évolutif

Lorsque l'on observe un SCN soumis à un **flux** incessant (comme un **stream** vidéo, audio, textuel, ou un ensemble hétéroclite de capteurs), le nombre d'**entités** \mathcal{E}_i croît au fil du temps, ou certaines entités sont jugées obsolètes et retirées. Sur le plan **mathématique**, la pondération $\omega_{i,j}(t)$ n'atteint pas nécessairement un **équilibre** final, mais se **modifie** constamment à chaque "tick" ou à chaque **itération** où de nouvelles entités arrivent. Le **Chapitre 9** se focalise donc sur :

- Des **protocoles** d'insertion incrémentale (`addEntity(\mathcal{E}_{new})`) dans le réseau,
- Des **stratégies** de mise à jour **online**, permettant d'éviter la nécessité de tout recalculer,
- Des **mécanismes** de "recuit local" ou de "mini-bursts de bruit" quand on craint une "cristallisation" trop rapide du SCN.

Ces aspects soulignent la **flexibilité** d'un **DSL**, qui ne se limite pas à une unique passe d'apprentissage mais entretient un **processus continu**, capable de gérer la **distribution** variable des flux sensoriels au fil du temps.

B. Adaptation Incrémentale pour le Multimodal

Les **scénarios** multimodaux "réels" impliquent souvent des **sources** distinctes qui arrivent à des rythmes différents. Par exemple, on reçoit 25 images par seconde pour la vidéo et un flux audio à 16 kHz, tandis que des blocs textuels peuvent être déclenchés de manière asynchrone en fonction d'événements spécifiques. Chaque **entité** nouvellement créée (frame, segment audio, chunk textuel) doit alors :

- **Déterminer** un **voisinage** dans le SCN existant (chap. 8.10.1.2 sur k-NN ou gating),
- **Calculer** la synergie $S(\text{new}, j)$ vis-à-vis de ces voisins,
- **Actualiser** $\omega_{\text{new},j}$ localement (insertion incrémentale),
- Éventuellement, **réajuster** quelques liens existants si la nouvelle entité modifie le paysage sémantique.

Le **Chapitre 9** décrit ces algorithmes incrémentaux, établissant les conditions pour **maintenir** la **cohérence** multimodale malgré l'arrivée d'entités multiples.

C. Intégration avec la Multi-Échelle

Nous avons précédemment (Chap. 8) évoqué la **fusion** à différents **niveaux** (features brutes, embeddings plus abstraits, concepts symboliques). Dans un environnement qui **change** en continu, il importe de **préserver** cette structure multi-niveau sans qu'un réapprentissage intégral (coût prohibitif) ne soit imposé à chaque nouveau lot de données. Le **Chapitre 9** montrera comment :

- On peut **gérer** un SCN "ouvert" où la **dynamique** ω s'élargit aux entités apparues plus tard.
- On évite la redondance ou la saturation via des **mécanismes** de suppression, d'inhibition (ex. liens sous un seuil),

- La **synergie** multimodale demeure un **fil** conducteur pour relier (ou non) les nouvelles entités.

D. Perspective

Le **DSL** multimodal gagne en **robustesse** lorsque, au fil du temps, il est capable d'adopter un **paradigme d'apprentissage continu**. Le réseau ne se fige pas, mais *se met à jour* avec un flux incluant la vidéo, l'audio, le texte ou des capteurs multiples. Du point de vue **mathématique**, on peut voir cela comme une **suite** $\{\omega(t)\}_{t \in \mathbb{N}}$ potentiellement infinie, où l'équilibre local évolue constamment avec les nouvelles **données**. Les petits **recuits** ou "mini-bursts de bruit" parfois introduits aident à éviter les **minima** locaux trop rigides. Le **Chapitre 9** discutera en détail de ces procédures, prolongeant la théorie multimodale du Chap. 8 par une **dimension de flux temps réel** et d'**adaptation** permanente.

Conclusion

La **suite** de ce **Chapitre 8** (scénarios multimodaux) aboutit donc naturellement au **Chapitre 9**, qui se focalise sur :

- La **dynamique en flux** : arrivée de nouvelles entités, retrait d'entités, gestion online,
- Les **heuristiques incrémentales** : k-NN, batch, inhibition adaptative,
- La **résilience** du SCN face aux **changements** de distribution ou aux données hétérogènes.

C'est une **extension** logique de la fusion multimodale en contexte **statique** (où l'on a un dataset figé) vers un contexte **dynamique** (où l'information multimodale ne cesse de se renouveler). De quoi approfondir les capacités "**évolutives**" du DSL.

8.11.2.2. Chap. 10 : Feedback Coopératif dans le DSL

Les sections précédentes (Chap. 8 sur la multimodalité, Chap. 9 sur l'apprentissage continu) ont montré comment un **SCN** (Synergistic Connection Network) peut intégrer **plusieurs modalités** (images, textes, audio, capteurs divers) et gérer l'**arrivée** ou la **disparition** d'entités en temps réel. Le **Chapitre 10** se consacre à un **niveau** nouveau niveau de complexité. Il traite de la **coopération** entre sous-SCN ou sous-modules via des **boucles de rétroaction** (feedback), permettant aux modalités de s'**influencer** mutuellement et d'obtenir une **harmonisation** globale plus fine qu'une simple juxtaposition. Le présent paragraphe (8.11.2.2) introduit la **logique** de ces "feedbacks coopératifs" et la façon dont ils s'inscrivent dans la dynamique du **DSL** (Deep Synergy Learning) multimodal.

A. Circulation de l'Information Multimodale et Boucles de Rétroaction

Dans un **DSL** multimodal, il est fréquent de disposer de **sous-SCN** ou de "modules" dédiés à chaque modalité. Un module **image**, un module **audio**, un module **texte**, etc. sont ainsi définis, chacun opérant sur ses propres entités et liens internes. Chacun opère sa propre **auto-organisation** locale (calcul de synergies S "intra-modal" et mise à jour des liaisons ω correspondantes). Cependant, ces modules ne travaillent pas en vase clos. Un **méta-niveau** ou un **super-nœud** global peut réunir leurs résultats, notamment les clusters partiels, les scores et les pondérations agrégées, afin de rendre un jugement "inter-modal".

Ce **niveau** global joue un **rôle** fondamental de **coordination** en orchestrant l'interaction entre les différents modules du **SCN**. Il fonctionne en trois étapes complémentaires.

Tout d'abord, il reçoit en **bottom-up** les informations issues de chaque module, incluant les **clusters internes**, les **liaisons fortes** et les **indices de confiance** associés aux entités détectées.

Ensuite, il procède à une **évaluation de la cohérence** entre ces modules, en vérifiant si les différentes modalités identifient un même objet ou événement à travers des canaux distincts, ce qui permet d'assurer une convergence interprétative.

Enfin, il agit en **top-down** en renvoyant un **feedback** correctif ou un **contrôle coopératif**, visant à ajuster localement les pondérations ω d'un module particulier.

Cet ajustement favorise une meilleure **harmonisation** des informations entre modalités, en maximisant la **concordance inter-modules** et en réduisant d'éventuelles contradictions ou incohérences.

On peut structurer cette **communication inter-modulaire** en deux **boucles complémentaires**, une **boucle ascendante** (*bottom-up*) et une **boucle descendante** (*top-down*), qui permettent une interaction continue entre les modules et le niveau global du SCN.

Dans la **boucle ascendante**, chaque module $\mathcal{M}_{\text{img}}, \mathcal{M}_{\text{aud}}, \mathcal{M}_{\text{txt}}, \dots$ transmet des informations au **noeud global** $\mathcal{N}_{\text{global}}$. Ces informations incluent les **liaisons internes**, les **clusters détectés** et des **scores résumés** sur la cohérence interne du module. Mathématiquement, cette étape correspond à la collecte des matrices de pondérations $\Omega^{(m)}$, ou d'autres indicateurs dérivés pour chacun des m modules.

Dans la **boucle descendante**, le **niveau global** analyse ces contributions et évalue la **cohérence inter-modules** à l'aide d'une fonction Ψ . Il génère ensuite un **feedback correctif** Δ_{down} à destination de chaque module, ce qui permet d'**affiner** les liaisons $\omega_{i,j}$ en fonction de la compatibilité observée entre les canaux. Sur le plan **DSL**, cette correction se traduit par un **terme additionnel** dans la mise à jour des pondérations, donné par :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i,j) - \tau \omega_{i,j}(t)] + \Delta_{\text{down}}^{(\text{global})}(i,j).$$

Le rôle du **terme correctif** $\Delta_{\text{down}}^{(\text{global})}(i,j)$ est de moduler dynamiquement l'évolution des liens. Un **terme positif** $\Delta_{\text{down}} > 0$ renforce la pondération $\omega_{i,j}$ lorsque plusieurs canaux confirment une association pertinente, améliorant ainsi la robustesse du SCN. À l'inverse, un **terme négatif** $\Delta_{\text{down}} < 0$ affaiblit le lien lorsqu'une incohérence est détectée, favorisant une meilleure spécialisation des clusters.

B. Coordination et Coopération entre Modules

Ce **feedback coopératif** permet aux canaux **image**, **audio**, **texte**, etc. de s'**aligner** plus efficacement. Si, à l'échelle globale, une correspondance cohérente est détectée entre plusieurs modalités, par exemple une **image identifiée comme "chien"** et un **segment audio reconnu comme "abolement"**, le **niveau global** du SCN peut alors émettre un **signal correctif** en direction des modules concernés, renforçant ainsi leur cohésion. Ce signal incite à **augmenter la pondération des liens pertinents** pour assurer une meilleure structuration du réseau.

Dans le **module vision**, ce signal se traduit par une **hausse des pondérations internes** impliquant le concept visuel "chien", ce qui renforce la liaison $\omega_{\text{chien}, \dots}$ avec d'autres images de chiens similaires.

Dans le **module audio**, un effet similaire se produit, amplifiant les connexions entre le son "abolement" et d'autres segments acoustiques apparentés, ce qui accroît la pondération $\omega_{\text{abolement}, \dots}$.

Enfin, le **feedback global** agit également au niveau des **liaisons cross-modales** reliant image et audio, consolidant ainsi la relation entre la représentation visuelle et le signal sonore correspondant. Cela conduit à un **renforcement automatique** de la **synergie multimodale** entre les entités associées, stabilisant progressivement la structure auto-organisée du SCN.

À l'inverse, un **conflict** (ex. l'audio prétend "miaulement", alors que la caméra et le texte concordent "chien") peut conduire le niveau global à pénaliser la liaison $\omega_{\text{audio}, \text{chien}}$. Le sous-module audio, recevant ce **feedback**, s'ajuste en conséquence. D'autres pondérations, comme $\omega_{\text{miaulement}, \dots}$, pourraient se voir inhibées, redirigeant ainsi l'**auto-organisation** locale.

C. Formulation Mathématique du Feedback

On peut formaliser ce **feedback descendant** comme un **terme** $\Delta_{\text{down}}^{(m)}(i,j)$ pour chaque module m . Celui-ci dépend de la **cohérence** globale repérée entre divers modules $\{m' \neq m\}$. Par exemple :

$$\Delta_{\text{down}}^{(m)}(i,j) = \alpha \sum_{m' \neq m} \text{compat}(\omega_{i,j}^{(m)}, \omega_{p,q}^{(m')}),$$

où compat évalue le degré de compatibilité entre la structure interne du module m et celle du module m' . Si la compatibilité est élevée, $\Delta_{\text{down}}^{(m)}(i, j)$ peut renforcer $\omega_{i,j}^{(m)}$. Sinon, on la réduit.

Cette mécanique rend le **SCN coopératif**, chaque module local recevant à la fois sa dynamique DSL standard et un **feedback** global, censé maintenir la cohérence inter-modules. Sur le plan **algorithmique**, on a une **boucle locale** qui correspond à la descente **DSL classique** et une **boucle globale** qui assure la réception, l'analyse et la distribution du feedback.

D. Avantages et Considérations

Sans ce **feedback** coopératif, chaque module peut demeurer cohérent en soi, mais ignorer les signaux corroborant ou réfutant ses hypothèses dans d'autres canaux. Le feedback **unifie** la perception en renforçant les clusters multi-canaux déjà plausibles et en corrigeant ou fragilisant ceux qui paraissent anormaux lorsqu'ils sont analysés dans leur globalité.

D'un point de vue **mathématique**, l'introduction de boucles de feedback rend la **dynamique** plus complexe, on ne manipule plus un simple $\omega(t)$, mais un ensemble $\{\omega^{(m)}(t)\}$ couplé par des interactions top-down. Il faut veiller à la **stabilité** et à éviter des oscillations ou divergences entre modules. Les heuristiques (recuit, inhibition, etc.) aident à canaliser ce phénomène.

Conclusion – Chap. 10

Le **Chapitre 10** approfondira :

- **Comment** s'organisent ces **boucles** ascendantes/descendantes,
- **Quelles** formules de **feedback** coopératif** permettent d'ajuster localement ω dans chaque module,
- **Pourquoi** ce modèle accroît la **robustesse** et la **performance** d'un DSL multimodal, où la simple auto-organisation *intra-modale* ne suffit pas toujours.

Ainsi, au-delà de la fusion multimodale vue dans le **Chap. 8** et l'apprentissage continu du **Chap. 9**, le **Chap. 10** introduit une **coopération** multi-niveau, actualisant la **synergie** S en fonction d'une **cohérence** ou d'un "contrôle" global. Cette **approche** permet un **SCN** véritablement "coopératif", coordonnant l'information entre canaux et résolvant les divergences, pour consolider des **clusters** multimodaux cohérents au sein d'un réseau réellement distribué et dynamique.

8.11.3. Synthèse sur la Valeur du DSL Multimodal

La **convergence** de flux multiples (vision, langage, audio, etc.) au sein d'un **Deep Synergy Learning** (DSL) multimodal présente un **intérêt** majeur pour la **cohésion** et la **richesse** des représentations. En effet, la synergie $S(\mathcal{E}_i, \mathcal{E}_j)$, lorsqu'elle englobe des **dimensions** hétérogènes (caractéristiques visuelles, sémantiques, acoustiques), autorise une **mise en correspondance** beaucoup plus fine et adaptative entre des entités supposément "différentes" (une image, une phrase, un extrait sonore), tout en préservant la **dynamique** d'auto-organisation propre au DSL.

8.11.3.1. Le DSL multimodal offre une "cohésion" adaptative entre vision, langage, son, tirant profit des principes d'auto-organisation

Les sections précédentes (Chap. 8) ont établi que le **DSL** (Deep Synergy Learning), appliqué à un **SCN** (Synergistic Connection Network), peut prendre en charge **plusieurs modalités** (image, texte, audio, etc.) en définissant pour chaque couple (i, j) une **synergie** $S(i, j)$ reflétant la correspondance ou la complémentarité entre les entités \mathcal{E}_i et \mathcal{E}_j . Cette **synergie**, lorsqu'elle est jugée forte, **renforce** la pondération $\omega_{i,j}$ via la règle DSL, ce qui conduit à la formation **auto-organisée** de **clusters** multimodaux. La présente section (8.11.3.1) récapitule la façon dont le DSL multimodal parvient à établir une **cohésion adaptative** entre différentes sources (vision, langage, son) et à exploiter les principes d'**auto-organisation** en leur conférant une structure unifiée et dynamique.

A. Principe de Base

Dans un **SCN** multimodal, chaque **entité** \mathcal{E}_i (image, segment textuel, extrait audio, etc.) se dote d'un vecteur (embedding) ou d'une représentation plus complexe. La **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ peut regrouper diverses composantes, chacune correspondant à une **modalité** donnée :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \alpha_{\text{vis}} S_{\text{vis}}(i, j) + \alpha_{\text{txt}} S_{\text{txt}}(i, j) + \alpha_{\text{aud}} S_{\text{aud}}(i, j) + \dots$$

Si, par exemple, \mathcal{E}_i est un **patch d'image** et \mathcal{E}_j un **segment textuel**, la composante S_{vis} n'intervient qu'à l'intérieur de la modalité "visuel" (image–image), et la composante S_{txt} s'applique à la modalité "texte". Pour le cross-modal "image–texte", on définit une **version** $S_{\text{vis,txt}}(i, j)$, etc. Chaque **pondération** $\omega_{i,j}$ est alors mise à jour par la règle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta [S(i, j) - \tau \omega_{i,j}(t)],$$

ce qui confère à la **dynamique** DSL (chap. 2.2.2) sa **flexibilité** d'extension à plusieurs canaux.

Cette **cohésion** entre modalités ne repose pas sur une procédure rigide, mais sur la **dynamique** locale d'**auto-organisation**. Dès que deux entités, comme une image représentant un "chat" et le mot "cat", se révèlent cohérentes par une similarité sémantique ou une co-occurrence avérée, leur pondération ω se **renforce**. À l'inverse, si d'autres paires, telles qu'une image de "voiture" et un audio de "miaulement", ne présentent pas de correspondance pertinente, leurs liaisons restent faibles ou s'annulent naturellement. Ainsi, le **réseau** DSL s'ajuste **en continu** à la cohérence perçue entre flux.

B. Auto-Organisation et Formation de Clusters

Au fil des itérations, des **sous-graphes** fortement connectés, appelés **clusters**, émergent progressivement. Ces clusters peuvent regrouper des images, des segments textuels, des morceaux audio ou d'autres entités multimodales, tous reliés par des pondérations ω élevées. Cela traduit une **unité** sémantique ou contextuelle (ex. "scène de plage", "concert de rock", "textes parlant de chat + images félines + bruits de miaulement"). Le **DSL** n'a pas besoin d'une supervision extérieure imposant ces regroupements. La **dynamique** elle-même, gouvernée par S et ω , les fait **surgir** de façon non supervisée.

Grâce à son **dynamisme**, le **DSL** multimodal demeure **robuste**. Si l'un des canaux, par exemple l'audio, est bruité ou manquant, les autres modalités, telles que le visuel et le textuel, prennent le relais et assurent la cohésion globale du cluster. On obtient souvent une **structure** plus stable que dans un cadre mono-modal, car chaque lien $\omega_{i,j}$ peut être soutenu par plusieurs composantes $\alpha_m S^{(m)}$ (fusion de scores). De plus, si de **nouvelles** entités surviennent (chap. 9.1), on recalcule localement les synergies, évitant un réapprentissage exhaustif de toute la base.

C. Illustration : un Score Composite

Pour illustrer, prenons la fusion tripartite **image–texte–audio**. La **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pourrait être :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \alpha_{\text{vis}} \text{vis_score}(\mathcal{E}_i, \mathcal{E}_j) + \alpha_{\text{txt}} \text{txt_score}(\mathcal{E}_i, \mathcal{E}_j) + \alpha_{\text{aud}} \text{aud_score}(\mathcal{E}_i, \mathcal{E}_j),$$

où :

- `vis_score` compare l'**embedding** d'images (ou de vidéos),
- `txt_score` compare l'**embedding** textuel (similitude cosinus, par ex.),
- `aud_score` compare l'**embedding** sonore (spectrogramme, MFCC).

Un **couple** (i, j) pourrait n'être pertinent que dans une seule modalité ou dans plusieurs. L'**auto-organisation** DSL, identique à la formule standard, assure la convergence vers des liaisons élevées lorsque les **entités** coïncident sur un ou plusieurs canaux.

D. Portée et Intérêt

On peut élargir ce **modèle** à d'autres capteurs (LiDAR, signaux inertIELS), ou à des **représentations** plus symboliques (chap. 8.7.4). Chacune génère une **composante** $S^{(m)}$ dans la synergie globale, permettant une **unification** graduelle de sources diverses. Le DSL s'avère donc apte à gérer une **multitude** de types d'entrées, pourvu que l'on définisse la **fonction** S propre à chaque canal ou paire de canaux.

De nombreux systèmes de **multimédia** (annotation, clustering, recommandation) ou de **robotique** (fusion sensorielle, perception 3D) bénéficient de la **cohésion** adaptative induite par le DSL. L'**absence** de supervision stricte ouvre des perspectives de **découverte** de correspondances, d'**auto-correction** en présence de bruit, et de **fusion** continue en cas de flux évolutifs (chap. 9).

Conclusion

Le **DSL multimodal**, dans lequel la synergie $S(i,j)$ agrège plusieurs canaux (vision, langage, audio, etc.), démontre une **cohésion** adaptative particulière. Les **pondérations** $\omega_{i,j}$ se renforcent lorsque plusieurs modalités **corroborent** ou se **complètent**, entraînant l'**émergence** de **clusters** véritablement multimodaux (images + sons + textes se reliant sur le même thème). Ce **mécanisme** auto-organisé garantit à la fois une **robustesse** (toute modalité peut pallier l'insuffisance d'une autre) et une **souplesse** (découverte de nouvelles associations, raffinement continu). La **dynamique** DSL, s'appliquant inchangée en présence de multiples flux, ouvre ainsi la porte à des scénarios riches où la vision, le langage, le son, etc. se **soutiennent** mutuellement pour **construire** (et mettre à jour) des **macro-nœuds** ou **clusters** sémantiquement cohérents.

8.11.3.2. Perspectives pour la Future IA Multimédia, la Robotique Sensorielle, etc.

En conclusion du **Chapitre 8**, il est clair que l'**approche** DSL (Deep Synergy Learning), et en particulier la mise en œuvre d'un **SCN** (Synergistic Connection Network) multimodal, ouvre des **perspectives** passionnantes pour un large éventail d'applications. Les **principes** d'**auto-organisation**, de **synergie** et de **multi-échelle** s'appliquent aussi bien à la **fusion** de données multimédia (texte, image, audio) qu'à des environnements plus "physiques" comme la **robotique sensorielle**, où de multiples capteurs (LiDAR, caméras, microphones, etc.) doivent coopérer.

A. IA Multimédia de Prochaine Génération

Les systèmes multimédia ne se contentent plus de **séparer** les canaux, tels que les images, les sons et les textes, pour les fusionner a posteriori. L'**avenir** envisage une **fusion évolutive** qui se construit progressivement au fil de la réception des données. Le **DSL multimodal** offre une **auto-organisation** organique où chaque entité, qu'il s'agisse d'un segment audio, d'un patch d'image ou d'une phrase textuelle, s'intègre progressivement au réseau, comme détaillé dans le chapitre 9 sur l'insertion incrémentale. La **cohérence** du système est maintenue par la règle DSL, qui renforce $\omega_{i,j}$ lorsque la synergie $S(i,j)$ est élevée, tandis que les liens faibles s'estompent naturellement au fil des itérations.

Cette **logique** évite le cloisonnement des canaux et permet de **déceler** des liens transversaux (ex. entre une vidéo donnée et un sous-titre partiellement correspondant, un timbre musical et une scène visuelle).

Au-delà de la simple correspondance surface (ex. cosinus d'embeddings), on peut imaginer des **niveaux** (micro → macro) où le SCN reconstruit des **conceptS** ou **thèmes** (chap. 6). Le DSL s'enrichit ainsi de :

- **Micro-liaisons** : patch d'image ↔ token textuel (façon "vision–langage local"),
- **Macro-liaisons** : grand cluster multimodal associant plusieurs flux autour d'un thème sémantique (ex. "actualité sur le climat").

Cette capacité multi-niveau profite à la **classification** (découvrir des catégories émergentes) et à la **navigation** (indexer des contenus, extraire des motifs).

Les **applications** du DSL multimodal sont nombreuses et diversifiées. En **réalité augmentée**, un SCN multimodal permet d'ajuster dynamiquement les annotations visuelles, textuelles ou sonores en fonction de l'image capturée par la caméra, en exploitant les liaisons ω qui unissent ces différentes modalités. Les **agents conversationnels enrichis**

bénéficient également de cette approche en intégrant simultanément les informations issues de la caméra, du son et du texte, ce qui améliore leur capacité à comprendre l'environnement et les gestes de l'utilisateur. Enfin, dans l'**analyse de grands corpus** multimédias, qu'il s'agisse de vidéos ou de documents, le DSL génère un **clustering auto-organisé unifié**, surpassant les méthodes traditionnelles qui compartimentent artificiellement les données en clusters distincts pour l'image, le texte ou l'audio.

B. Robotique Sensorielle et Coordination Synergique

En **robotique**, un système auto-organisé multimodal prend en charge les mesures issues de différentes sources, notamment le **LiDAR**, la **caméra** pour l'analyse visuelle, les **capteurs inertiels** et les **microphones**. Le **SCN** établit des liaisons entre les entités **LiDAR** et **visuelles** lorsque leur synergie spatiale et temporelle est confirmée. Il filtre ou ignore les associations incohérentes et ajuste progressivement la pondération des connexions ω en fonction de l'évolution du flux de données, conformément aux principes du **DSL** appliqués aux systèmes dynamiques (chap. 9).

La **dynamique** DSL confère au robot un mécanisme de **fusion** souple, sans imposer un pipeline figé (**LiDAR**↔**camera** obligatoire). Au besoin, si un capteur se dégrade, le réseau auto-organisé **bascule** plus de poids ω vers d'autres capteurs.

Plus ambitieusement, on peut coupler la **fusion** sensorielle à la **structure** motrice dans un **SCN** commun. Les nœuds représentant les **capteurs** tels que le **LiDAR**, la **caméra** ou les **microphones** s'auto-organisent en fonction de leur synergie intrinsèque. Parallèlement, les nœuds associés aux **moteurs**, qu'il s'agisse de joints ou d'actionneurs, reçoivent des retours établissant un lien direct entre la perception et l'action. L'**auto-organisation** assure ainsi un couplage **sensorimoteur flottant**, permettant à chaque entité motrice d'**apprendre** quels capteurs sont les plus pertinents pour sa commande et, inversement, aux capteurs d'adapter leur association aux actions qu'ils influencent.

On obtient ainsi une **architecture** fractale ou multi-niveau, unifiant la **perception** et l'**action** au sein d'un même **réseau** ω .

Dans le cadre de la **robotique multi-agent**, notamment pour des **floktes de drones** ou des **essaims de robots**, le **DSL multimodal** établit une répartition dynamique des **liens** $\omega_{i,j}$. Ces connexions ne se limitent pas aux capteurs d'un unique robot, mais s'étendent à **plusieurs** robots si une synergie pertinente est détectée. Lorsque des objectifs communs, des localisations proches ou des signaux partagés renforcent ces interactions, des **macro-clusters** inter-agents émergent naturellement. Ce mécanisme favorise une **coopération distribuée**, évitant ainsi la nécessité d'une supervision centrale qui imposerait un schéma rigide d'interaction.

C. Vers une IA Synergique et Fractale

Les **scénarios** de l'IA multimédia et de la **robotique sensorielle** convergent dans la nécessité de gérer simultanément **plusieurs flux**, qu'il s'agisse d'images, de sons, de capteurs 3D ou de textes. Cette gestion implique l'**assemblage** de ces flux à **différents niveaux**, allant des micro-liaisons locales aux macro-liaisons sémantiques. L'approche repose sur un **paradigme d'auto-organisation**, évitant toute contrainte a priori sur la manière dont ces informations doivent être fusionnées, permettant ainsi une intégration dynamique et adaptative des signaux issus de diverses modalités.

Le DSL multimodal satisfait ces critères grâce à l'**universalité** de la règle DSL (pondérations ω renforcées localement, inhibition et recuit optionnels, chap. 7.3–7.4).

Dans des environnements évolutifs (nouveaux flux multimédias, nouveau capteur, nouveau robot), le **DSL** s'adapte **progressivement** (chap. 9.1). L'**inhibition** évite la prolifération indiscriminée de liens, tandis que le **recuit** (ou mini-bursts de bruit) prévient l'enlisement dans de faux minima. Cette **combinaison** d'éléments mathématiques (synergie, inhibition, recuit, partition, etc.) ouvre la voie à des **réseaux** robustes face à la quantité et la variabilité de données.

Conclusion

Les **perspectives** pour la **Future IA Multimédia** (fusion évolutive des flux image, audio, texte...) et la **robotique sensorielle** (coordination multi-capteurs, multi-agents, couplage perception-action) sont vastes. Le **DSL** multimodal, avec sa dynamique d'**auto-organisation** (mise à jour ω localement, inhibition et recuit globaux), constitue un **cadre** pertinent :

- **Souple** : car il n'impose pas un pipeline statique, autorisant l'arrivée de nouvelles modalités,
- **Robuste** : la synergie multi-canaux permet de pallier la défaillance d'un flux et de découvrir de nouvelles associations,
- **Scalable** : via des heuristiques (k-NN, gating, chap. 8.10.1) et du calcul distribué (chap. 8.10.4).

En se tournant vers le **Chap. 9** et le **Chap. 10**, on verra comment les **mécanismes** d'apprentissage continu et de feedback coopératif renforcent encore cette vision, posant les fondations d'une **IA** plus **organique**, capable d'**unifier** les informations multimodales dans un réseau **auto-évolutif** et **adaptatif** — un pas de plus vers une intelligence vraiment **holistique**.