Par Mohammed Bouayoun, Janvier 2025 (bouyao@gmail.com)

Chapitre 7 : Algorithmes d'Optimisation et Méthodes d'Adaptation Dynamique

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-Échelle (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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ 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 :

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

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 paramétrer 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 **paliers** : un **niveau micro** (patchs 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 paramétrer (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 :

1. **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 $\{\omega_{i,j}\}$.
2. **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).
3. **Architecture** (Chap. 5) : la **modularité** du SCN autorise l'injection de mécanismes d'**optimisation** (bruit, recuit, inhibition) dans divers sous-systèmes.
4. **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 **surchage** 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

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

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 nœud). 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 nœud $\mathcal{E}_{n+1}$, 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

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau\, \omega_{i,j}(t)\right],$$

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 $\mathcal{J}\big(\{\omega_{i,j}\}\big)$. 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

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau\, \omega_{i,j}(t)\right],$$

on ajoute une **température** $\sigma(t)$ qui décroit dans le temps, plus un bruit $\xi_{i,j}(t)$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau\, \omega_{i,j}(t)\right] \; + \; \sigma(t)\, \xi_{i,j}(t).$$

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 ($\sigma(t) \to 0$ quand $t \to \infty$). Divers choix s'offrent pour $\sigma(t)$:

$$\sigma(t) = \frac{\sigma_0}{\log(t+2)}, \qquad \text{ou} \qquad \sigma(t) = \alpha^t \quad (\alpha < 1),$$

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 $\mathcal{J}$. 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** $\mathcal{J}(\{\omega_{i,j}\})$ 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\mathrm{DSL}}(i,j) + \Delta_{\mathrm{global}}(i,j),$$

où $\Delta_{\mathrm{DSL}}(i,j)$ se limite à la variation usuelle associée à la **descente** contrôlée par la synergie locale :

$$\Delta_{\mathrm{DSL}}(i,j) = \eta \left[S(i,j) - \tau\, \omega_{i,j}(t) \right].$$

Le terme $\Delta_{\mathrm{global}}(i,j)$ 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 $\Delta_{\mathrm{global}}(i,j) = \sigma(t)\, \xi_{i,j}(t)$. 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

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\mathrm{DSL}}(i,j) + \Delta_{\mathrm{global}}(i,j)$$

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 $\mathcal{J}$. 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 prématurée. 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 nœud 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

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

en y ajoutant un terme inhibiteur, conduisant à

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] - \gamma \sum_{k\neq j} \omega_{i,k}(t).$$

Ce terme d'inhibition, noté $-\gamma \sum_{k\neq j} \omega_{i,k}\,(t)$, induit une compétition latérale : plus un nœud $\mathcal{E}_i$ 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 surnombre. 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 $\omega_{\max}$ aux pondérations. Concrètement, une fois le calcul de $\omega_{i,j}(t+1)$ effectué, on applique

$$\omega_{i,j}(t+1) \;\leftarrow\; \min\big(\,\omega_{i,j}(t+1),\;\omega_{\max}\big).$$

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 $\omega_{\max}$ 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 $\omega_{\max}$, on la ramène simplement à $\omega_{\max}$.

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 $\omega_{\max}$. 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 $\omega_{\max}$ 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 $\omega_{\max}$ 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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] \;-\; \gamma \sum_{k \neq j} \omega_{i,k}(t),$$

ce qui contraint 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 $\omega_{\max}$ empêchant toute pondération de croître indéfiniment : après mise à jour, on applique

$$\omega_{i,j}(t+1) \leftarrow \min\big(\omega_{i,j}(t+1),\ \omega_{\max}\big).$$

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 signifiantes : 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

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

devient

$$\omega_{i,j}(t+1) = \omega_{i,j}(t)\eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] - \gamma \sum_{k \neq j} \omega_{i,k}(t)$$

puis est suivi d'un **clipping** :

$$\omega_{i,j}(t+1) \leftarrow \min\big(\omega_{i,j}(t+1),\ \omega_{\max}\big).$$

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 $\omega_{\max}$). 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 $\omega_{\max}$ 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 $\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \ldots$ 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 $\mathrm{O}(n^2)$), 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 :

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

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** $\mathrm{N}(n+1) \subset \{1, \ldots, n\}$, c'est-à-dire à un ensemble restreint de nœuds j pour lesquels on estime que la **synergie** $S\big(\mathcal{E}_{n+1}, \mathcal{E}_j\big)$ 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** $\Delta_{\mathrm{down}}(i,j)$ dans la mise à jour :

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

Si le macro-nœud (ou super-nœud) détecte qu'un cluster local n'est plus pertinent à la tâche courante, $\Delta_{\mathrm{down}}(i,j)$ 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 $\Delta_{\mathrm{down}}(i,j)$ 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

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

peut être comprise dans un **cadre** plus large de minimisation ou de maximisation d'une **fonction d'énergie** $\mathcal{J}(\boldsymbol{\Omega})$. 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 $\mathrm{O}(n^2)$ (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. α^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 $\omega_{\max}$.

  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** :

1. **Simulation** d'un SCN de petite taille (20–50 entités) avec des patterns artificiels, observant la convergence avec/sans recuit.

2. **Exemple** robotique (multi-capteurs, multi-tâches) : on voit comment l'inhibition aide à sélectionner un sous-ensemble de connexions utiles pour chaque scénario.

3. **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 nn, peut devenir considérable. Il en résulte une structure de taille O(n2)\mathcal{O}(n^2) pour la matrice de pondérations ωi,j\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)(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\omega 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'équilibrent. 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\omega 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'**incrémentation 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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \Delta_{\text{down}}(i,j)$. 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 scenarii 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 $\mathcal{J}(\boldsymbol{\Omega})$ (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 $\mathcal{J}(\boldsymbol{\Omega})$ 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 Détaillée : Rappel de la Notion de $\mathcal{J}(\boldsymbol{\Omega})$ (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 $\mathcal{J}(\boldsymbol{\Omega})$ (où $\boldsymbol{\Omega}$ 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 :

$$\mathcal{J}(\boldsymbol{\Omega}) = -\sum_{i,j} \omega_{i,j}\big(S(i,j)\big) \; + \; \mathcal{R}(\boldsymbol{\Omega}).$$

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 $\mathcal{J}(\boldsymbol{\Omega})$

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** $\boldsymbol{\Omega} \in \mathbb{R}^{n\times n}$ (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**" $\mathcal{J}(\boldsymbol{\Omega})$ selon deux composantes principales :

$$\mathcal{J}(\boldsymbol{\Omega}) = -\sum_{(i,j)} \omega_{i,j}\; S(i,j) + \mathcal{R}(\boldsymbol{\Omega}).$$

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, $\mathcal{R}(\mathbf{\Omega})$, 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 :

$$\mathcal{R}(\mathbf{\Omega}) = \lambda_1 \sum_{(i,j)} \left(\omega_{i,j}\right)^2, \quad \text{(ex. terme quadratique)}$$

pour modérer la croissance illimitée, ou bien

$$\mathcal{R}(\mathbf{\Omega}) = \lambda_2 \sum_{(i,j)} \omega_{i,j} \quad \text{ou} \quad \lambda_3 \sum_{(i,j)} \left|\omega_{i,j}\right|,$$

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 ↑ si trop de liaisons connectées à un même nœud s'avèrent très fortes.

En posant $\mathcal{J}(\mathbf{\Omega}) = -\sum \omega_{i,j}\, S(i,j) + \mathcal{R}(\mathbf{\Omega})$, on peut considérer la **configuration Ω** qui **minimise** $\mathcal{J}$. Cela revient à :

$$\min_{\mathbf{\Omega}} \mathcal{J}(\mathbf{\Omega}) = \min_{\mathbf{\Omega}} \left[-\sum_{(i,j)} \omega_{i,j}\; S(i,j) + \mathcal{R}(\mathbf{\Omega}) \right].$$

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 $1/2\ \tau \sum \left(\omega_{i,j}\right)^2$) 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

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau\, \omega_{i,j}(t) \right]$$

correspond, en omettant d'autres nuances, à

$$-\frac{\partial}{\partial \omega_{i,j}} \left(-\omega_{i,j}\, S(i,j) + 1/2\ \tau \left(\omega_{i,j}\right)^2 \right) = S(i,j) - \tau\, \omega_{i,j}.$$

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 $\boldsymbol{\Omega}$.

Conclusion : Vers un Cadre d'Optimisation Global

En guise de synthèse, la notion de fonction potentielle $\mathcal{J}(\boldsymbol{\Omega}) = -\sum\omega_{i,j}S(i,j) + \mathcal{R}(\boldsymbol{\Omega})$ offre une **lecture** globale de la **dynamique** locale : rechercher la configuration de $\boldsymbol{\Omega}$ 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 Détaillée : Cas Simple de l'Énergie $\mathcal{J} = -\sum\omega_{i,j}\,S(i,j) + \tau/2\,\sum\left(\omega_{i,j}\right)^2$

Dans la **Section 7.2.1.1**, on a rappelé comment la notion d'**énergie** (ou **fonction** potentielle) $\mathcal{J}(\boldsymbol{\Omega})$ 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** $\boldsymbol{\Omega}$ 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

$$\mathcal{J}(\boldsymbol{\Omega}) = -\sum_{i,j}\ \omega_{i,j}\ S(i,j)\ +\ \frac{\tau}{2}\sum_{i,j}\ \left[\,\omega_{i,j}\,\right]^2,$$

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 en 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \ \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}\Big(\omega_{i,j}(t)\Big).$$

En substituant,

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

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 :

4. **renforcer** $\omega_{i,j}$ si $S(i,j)$ est élevé, pour diminuer $-\sum\omega S$,
5. **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 $\mathcal{J}(\boldsymbol{\Omega}) = -\sum_{(i,j)}\omega_{i,j}\ S(i,j) + \tau/2\sum\big(\omega_{i,j}\big)^2$ 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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$. 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 Détaillée : 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 $\mathcal{J}(\mathbf{\Omega})$ 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 $\mathcal{J}(\mathbf{\Omega};\, t)$ ou $\mathcal{J}(\mathbf{\Omega}, \mathbf{\Omega})$ 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 $\mathbf{\Omega}$ 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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\, \omega_{i,j}(t)\right]$ 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 $\sum\left(\omega_{i,j}\right)^2$ comme une somme d'éléments indépendants. Mais si la "régularisation" prenait la forme $\sum\left(\omega_{i,j}\, \omega_{i,k}\right)$ (compétition pour un nœud $\mathcal{E}_i$), la fonction $\mathcal{R}(\mathbf{\Omega})$ 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** $\mathcal{J}(\mathbf{\Omega})$ 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 Détaillée : Descente de Gradient Implicite via $\omega_{i,j}(t+1) = \omega_{i,j}(t) \; - \; \eta \; \partial\mathcal{J}/\partial\omega_{i,j}$

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) $\mathcal{J}(\mathbf{\Omega})$. Dans le cas le plus simple (tel que décrit en **Section 7.2.1.2**), cette $\mathcal{J}$ prend la forme :

$$\mathcal{J}(\mathbf{\Omega}) = -\sum_{i,j} \; \omega_{i,j}\, S(i,j) + \frac{\tau}{2} \sum_{i,j} \; (\omega_{i,j})^2,$$

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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \; \frac{\partial\, \mathcal{J}}{\partial\, \omega_{i,j}}\big(\mathbf{\Omega}(t)\big).$$

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 $\mathcal{J}(\mathbf{\Omega})$ sur l'espace $\{\omega_{i,j}\}_{(i,j)}$. Si on recherche une **minimisation** de $\mathcal{J}$ par une descente de gradient explicite, on écrit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta \; \frac{\partial \mathcal{J}}{\partial \omega_{i,j}}\big(\mathbf{\Omega}(t)\big),$$

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 pente **descendante** dans l'espace $\{\omega_{i,j}\}$.

Dans le **DSL**, on constate que la **règle** :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau\, \omega_{i,j}(t)\right]$$

peut se réécrire :

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

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 :

$$\mathcal{J}(\mathbf{\Omega}) = -\sum \omega_{i,j}\, S(i,j) \;+\; \text{(termes de régularisation sur } \omega_{i,j}),$$

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** $\mathcal{J}(\mathbf{\Omega})$ se reconfigure simultanément ; on ne suit plus exactement la descente d'un **paysage** figé, mais d'un paysage qui peut bouger (ce qui n'invalide pas la logique, mais la rend plus dynamique).

On peut schématiser la règle DSL comme un **schéma** :

$$\Delta\omega_{i,j}(t) = -\,\eta\,\nabla_{\omega_{i,j}}\mathcal{J}\big(\mathbf{\Omega}(t)\big) \quad \Rightarrow \quad \omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta\omega_{i,j}(t).$$

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 surpasse 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 $\mathbf{\Omega}$ 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

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) - \eta\,\frac{\partial\mathcal{J}}{\partial\omega_{i,j}}\big(\mathbf{\Omega}(t)\big)$$

à une **descente** de gradient (implicite) sur la fonction $\mathcal{J}(\mathbf{\Omega})$ 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** $\mathcal{J}(\mathbf{\Omega})$ (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 $\mathcal{J}(\mathbf{\Omega})$ 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 $\mathbf{\Omega}^*$ tel que :

$$\nabla \mathcal{J}(\mathbf{\Omega}^*) = 0 \quad \text{et} \quad \mathcal{J}(\mathbf{\Omega}^*) > \mathcal{J}\big(\mathbf{\Omega}^{(\text{global min})}\big).$$

En d'autres termes, on ne peut plus abaisser $\mathcal{J}$ en effectuant un petit pas local autour de $\mathbf{\Omega}^*$, mais il existe malgré tout un **point** $\mathbf{\Omega}^{(\text{global min})}$ 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 :

$$\mathcal{A}_k = \{\mathbf{\Omega}(0) \ \mid \ \lim_{t\to\infty} \mathbf{\Omega}(t) = \mathbf{\Omega}^{(k)}\},$$

où $\mathbf{\Omega}^{(k)}$ est le point fixe (minimum local) correspondant. Chaque $\mathcal{A}_k$ est appelé **bassin** d'attraction de $\mathbf{\Omega}^{(k)}$. Cela signifie que le **résultat** final dépend **fortement** de la condition initiale $\mathbf{\Omega}(0)$.

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 :

6. **Secouer** la configuration (stochastiquement, recuit simulé, etc.),
7. **Explorer** plusieurs initialisations (multi-run),
8. **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 Détaillée : 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 $\{\omega_{i,j}\}$.

9. **Sauts Non Localisés**

   Les algorithmes de **descente** (gradient ou variations) ne s'autorisent que de petits pas dans la direction locale de $-\nabla\mathcal{J}(\mathbf{\Omega})$. 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.

10. **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 $\nabla \mathcal{J}(\mathbf{\Omega})$ annule la progression en un point, un “coup” aléatoire suffisant peut pousser $\mathbf{\Omega}$ en dehors du puits, d’où une chance d’atteindre un minimum plus profond.

11. **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 $\mathcal{J}(\mathbf{\Omega})$. 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 :

12. **Recuit Simulé**

L’idée (détaillée en **Chap. 7.3**) consiste à assimiler la configuration $\mathbf{\Omega}$ à un “état” physique, et la fonction $\mathcal{J}$ à une “énergie”. Au début, on impose une **température** élevée, permettant des **mouvements** 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 ($\sigma(t) \to 0$), 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 :

$$\Delta\omega_{i,j}(t) = -\eta\,\frac{\partial \mathcal{J}}{\partial \omega_{i,j}}\big(\mathbf{\Omega}(t)\big) \;+\; \sigma(t)\,\xi_{i,j}(t),$$

où $\sigma(t)$ décroît en fonction de t et $\xi_{i,j}(t)$ est un bruit (souvent gaussien).

13. **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.

14. **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 "saute" 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 :

15. **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.

16. **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.

17. **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 7.2.2.2**). Les **méthodes** stochastiques (recuit simulé, injection de bruit) et les **heuristiques** globales (algorithmes génétiques, colonies de fourmis, etc.) constituent une **extension** incontournable pour :

18. **Élargir** la recherche de solutions dans l'espace $\{\omega_{i,j}\}$,

19. **Franchir** les barrières d'énergie qui empêchent la descente locale d'accéder à un bassin plus profitable,

20. **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 Détaillée : 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

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

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 $\omega_{i,j}(t+1) \leftarrow \omega_{i,j}(t) + \cdots$ 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 éparse.

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 Exploser 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 $\{1, \ldots, n\}$ 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

$$\mathcal{J}(\boldsymbol{\Omega}) \;=\; -\sum_{i,j} \omega_{i,j}\, S(i,j) \;+\; \text{(termes d'inhibition ou de régularisation)},$$

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 $\mathcal{J}(\boldsymbol{\Omega})$, 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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\,\big[S(i,j) - \tau\,\omega_{i,j}(t)\big],$$

mais pour un ensemble restreint de paires (i,j). Le coût par itération s'établit en $\mathrm{O}(n\,k)$ au lieu de $\mathrm{O}(n^2)$, 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 $\| \mathbf{\Omega}(t+1) - \mathbf{\Omega}(t) \| < \varepsilon$ 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 :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;-\; \eta\,\nabla_{\omega_{i,j}}\mathcal{J}\big(\mathbf{\Omega}(t)\big) \;+\; \sigma(t)\,\xi_{i,j}(t),$$

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 $\mathrm{O}(n\,k)$ ou en un temps $\mathrm{O}(n^2)$ très partiellement employé (puisqu'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 $\mathrm{O}(n^2)$ 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 $\mathrm{O}(n^2 \times T)$ 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 $i \in \{1, \ldots, n\}$, un ensemble $\mathrm{N}_k(i)$ de **cardinalité** k correspondant aux entités j qui maximisent la synergie $S(i,j)$ ou la pondération $\omega_{i,j}(t)$. On force ainsi :

$$\omega_{i,j}(t) = 0 \quad \text{si} \quad j \notin \mathrm{N}_k(i).$$

Le coût total chute alors de $\mathrm{O}(n^2)$ à $\mathrm{O}(n\,k)$, 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 $\mathrm{O}(m)$ 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 Restrictif"

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 $\mathrm{C}(i)$ de candidats (quelques centaines parmi n) où la liaison $\omega_{i,j}$ est autorisée ou recalculée. On obtient alors un coût $\mathrm{O}(\sum_i |\mathrm{C}(i)|) \approx \mathrm{O}(n\,k)$. L'**objectif** est de balayer un sous-graphe localement pertinent : l'entité $\mathcal{E}_i$ n'entretient d'interactions que dans $\mathrm{C}(i)$. 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 $\mathrm{C}(i)$ 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 $\mathrm{O}(n^2)$ 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 clarsemé de taille $\mathrm{O}(n\,k)$ ou $\mathrm{O}(m)$ (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 :

$$\widetilde{\omega}_{i,j}(t) = \begin{cases} \omega_{i,j}(t), & \text{si } (i,j) \in \Lambda, \\ 0, & \text{sinon,} \end{cases}$$

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 $\{\widetilde{\omega}\}$. 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 $\Delta\mathcal{J}$ entre la solution restreinte $\tilde{\omega}$ et la solution hypothétique $\omega^{(\text{complet})}$. Tant que $\Delta\mathcal{J}$ 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 draconiennement 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 sparce, 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. Appliqué aux réseaux **DSL** (Deep Synergy Learning) et à la minimisation d'une **énergie** $\mathcal{J}(\mathbf{\Omega})$, 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 :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\right] \;+\; \sigma\big(T(t)\big)\,\xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ désigne un **bruit** (gaussien, uniforme…) d'amplitude gouvernée par la température $T(t)$. Lorsque $T(t)$ est **élevée**, 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 $\omega_{i,j}(t+1) \approx \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ 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 $\exp\big(-\Delta\mathcal{J}/(k_B T)\big)$. 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 :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] \;+\; \sqrt{\alpha\,T(t)}\;\mathcal{Z}_{i,j}(t),$$

où $\alpha > 0$ est un coefficient d'échelle, $\mathcal{Z}_{i,j}(t) \sim \mathcal{N}(0,1)$, 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 $\omega_{i,j} \leftarrow \omega_{i,j} + \eta\big[S(i,j) - \tau\,\omega_{i,j}\big]$.

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 $\big[S(i,j) - \tau\,\omega_{i,j}\big]$ — 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) précisas 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 $\mathcal{J}(\boldsymbol{\Omega})$. 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 $\mathcal{J}$) 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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ se comporte comme une **descente** locale, on n'accepte pas de sauts qui augmentent l'énergie $\mathcal{J}(\Omega)$. 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 :

$$P(\text{accepter } \Delta\Omega) = \min(1,\ \exp[-\Delta\mathcal{J}/T(t)]),$$

ce qui signifie que si $\Delta\mathcal{J} > 0$ (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 $\sigma\big(T(t)\big)\,\xi_{i,j}$ 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 $\boldsymbol{\Omega}$ (toutes les $\omega_{i,j}$), la fonction d'énergie $\mathcal{J}(\Omega)$ 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 espace 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 $\mathcal{J}(\Omega)$ 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, $\max\big(S(i,j)\big)$, 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 $\Longrightarrow$ refroidissement **très lent** $\Longrightarrow$ exploration riche, mais temps de convergence potentiellement prohibitif,

- α trop bas $\Longrightarrow$ on "refroidit" trop vite $\Longrightarrow$ 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 ($\mathrm{O}(n^2)$ connexions). Sinon, le recuit n'explore 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 :

21. **Refroidissement trop rapide** : on n'explore pas assez, la dynamique revient à une quasi-descente locale,

22. **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 **temperature** $T(t)$ 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 : $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] + \xi_{i,j}(t)$

Le **recuit simulé** dans un **Synergistic Connection Network (SCN)** se formalise en **complétant** la mise à jour habituelle $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ 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 :

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

Cette petite modification rend la **dynamique** du SCN stochastique. Elle permet, lorsqu'on "chauffe" (température haute), de **monter** localement en énergie $\mathcal{J}(\boldsymbol{\Omega})$, 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 :

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

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 :

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

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** $\Delta\omega \propto -\nabla\mathcal{J}$ interdit d'aller vers un état de plus haute énergie. Or, franchir un "col" requiert momentanément $\Delta\mathcal{J} > 0$. 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 $\Delta\mathcal{J}$ 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 $\mathcal{J}(\Omega)$. Ainsi, **plus** $T(t)$ est élevé, **plus** la probabilité d'accepter un saut énergétiquement défavorable est grande.
On choisit souvent :

$$\xi_{i,j}(t) \sim \mathcal{N}\left(0,\ \sigma^2\big(T(t)\big)\right),$$

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 $\eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \xi_{i,j}(t),$$

avec $\xi_{i,j}(t) \sim \mathcal{D}\big(0, \sigma^2(t)\big)$ 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j)\tau\,\omega_{i,j}(t)\big] + \xi_{i,j}(t).$$

Le $\xi_{i,j}(t)$ est alors le bruit, dit **"thermique"**, dont la distribution centrée ($\mathrm{E}[\xi] = 0$) 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 :

23. Une **loi gaussienne** : $\xi_{i,j}(t) \sim \mathcal{N}\big(0, \sigma^2(t)\big)$,
24. Une **loi uniforme** : $\xi_{i,j}(t) \sim \mathrm{Unif}\big(-\Delta(t), +\Delta(t)\big)$.

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. α^t, $\frac{1}{\ln t}$, etc.) :

$$T(t+1) = \alpha\, T(t) \quad (0 < \alpha < 1), \quad \text{ou} \quad T(t) = \frac{T_0}{\log(t+2)}, \quad \ldots$$

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 $\Delta\mathcal{J} > 0$, 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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ 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 :

25. **Phase chaude** : $T(t)$ élevé, **fortes** fluctuations $\xi_{i,j}$, exploration large,

26. **Phase froide** : $T(t)$ décroît, les **perturbations** se réduisent, on converge.

Cette **équation** unifiée $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots + \xi_{i,j}(t)$ 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 $t = 0 \ldots T_{\max}$), 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 $t \to 0$, 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 $\boldsymbol{\Omega}$, dont la probabilité de franchir un mur d'énergie $\Delta\mathcal{J} > 0$ 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 :

27. **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.

28. **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 ⇒ grande **exploration**,
- Basse température ⇒ **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** discrétisé. 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 :

$$\xi_{i,j}(t) \sim \mathcal{N}\big(0,\ \sigma^2(t)\big),$$

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 :

$$\xi_{i,j}(t) \sim \text{Unif}\big(-\Delta(t),\ +\Delta(t)\big),$$

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.

D. 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 appliqué.

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 :

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

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 $\boldsymbol{\Omega}(t)$ 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 nœuds 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** :

29. **Gaussienne** $\mathcal{N}(0,\sigma^2)$: forme continue, symétrique, plus usuelle,

30. **Uniforme** Unif$(-\delta, +\delta)$: bornée, simple, contrôle direct de l'amplitude maximale,

31. **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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ est tiré selon une distribution $\mathcal{D}\big(0, \sigma^2(t)\big)$. 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 :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right], & \text{si } t \bmod K \neq 0, \\ \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] + \xi_{i,j}(t), & \text{si } t \bmod K = 0. \end{cases}$$

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 :

32. **À chaque** itération (recuit "full stochastique"),
33. **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 remodèle 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \xi_{i,j}(t),$$

où $\xi_{i,j}(t)$ suit une distribution $\mathrm{Distrib}\big(0,\ \sigma^2(t)\big)$, 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 $\mathcal{J}$. 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 compromettraient 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, $\mathrm{Var}(\{\omega_{i,j}\})$, 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 $\sigma(t) \to 0$, 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 :

1. **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 précoces.

2. **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.

3. **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 $\Delta_{\mathrm{DSL}}\omega_{i,j}$, on injecte le bruit, puis on refroidit la température. Le tout se poursuit sur un nombre d'itérations donné.

Algorithm: DSLRecuitSimule$\left(\{\omega_{i,j}^{(0)}\},\ S(\cdot,\cdot),\ \eta,\ \tau,\ T_0,\ \alpha,\ N_{\max}\right)$

Initialisation: $\omega_{i,j}(0) \leftarrow \omega_{i,j}^{(0)}, \quad T(0) \leftarrow T_0, \quad t \leftarrow 0.$

Tant que $t < N_{\max}$:

1) $\Delta_{\mathrm{DSL}}\omega_{i,j}(t) = \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right].$

2) Générer un bruit $\xi_{i,j}(t) \sim \mathcal{N}(0,\,1)$ (par exemple), puis poser $\xi_{i,j}^{(\mathrm{eff})}(t) = T(t)\,\xi_{i,j}(t).$

3) $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\mathrm{DSL}}\omega_{i,j}(t) + \xi_{i,j}^{(\mathrm{eff})}(t).$

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 $\eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$. 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 aisé 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** $\mathcal{E}_1, \ldots, \mathcal{E}_{10}$. 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** :

  $\{\mathcal{E}_1, \ldots, \mathcal{E}_5\}$ vs. $\{\mathcal{E}_6, \ldots, \mathcal{E}_{10}\}$.

- **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 $\mathcal{J}(\Omega^A)$ égale ou presque à $\mathcal{J}(\Omega^B)$. 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 :

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

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** $\xi_{i,j}(t) \sim \mathcal{N}\big(0, \sigma^2(t)\big)$, la mise à jour devient :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(i,j) - \tau\, \omega_{i,j}(t) \right] + \sigma(t)\, \xi_{i,j}(t).$$

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" $\Delta\mathcal{J} > 0$ 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 $\sigma(t) \to 0$ 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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] \;+\; \xi_{i,j}(t),$$

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 retarde 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** appliqué 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 imposerait 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 schématise 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 **coefficient** 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

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

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 $\eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$ 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 économe 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] - \gamma \sum_{k\neq j} \omega_{i,k}\,(t).$$

Le résultat est un **réseau** (SCN) plus **sparse** et plus **lisible**, 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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\big] \;-\; \gamma(t)\sum_{k\neq j}\omega_{i,k}(t).$$

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 ($\mathrm{Densite}(\omega)$, 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 $0 \le t < T_1$, 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_{final} 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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] \;-\; \gamma(t) \sum_{k\neq j} \omega_{i,k}(t)$$

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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[\mathbf{S}(i,j) \;-\; \tau\,\omega_{i,j}(t)\right] \;-\; \boldsymbol{\gamma}\sum_{k\neq j}\omega_{i,k}(t),$$

dans laquelle le **coefficient** $\gamma > 0$ 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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[\mathbf{S}(i,j) \;-\; \tau\,\omega_{i,j}(t)\right] \;-\; \gamma(t)\sum_{k\neq j}\omega_{i,k}(t),$$

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 :

4. 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 $\mathbf{1}$ la fonction indicatrice.

5. Une **moyenne des pondérations**,

$$\Delta(t) \;=\; \frac{1}{n(n-1)}\sum_{i\neq j}\omega_{i,j}(t).$$

6. 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 $\gamma_{\min}$ et $\gamma_{\max}$, 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 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 :

$$I(t) = \frac{1}{\binom{n}{2}} \sum_{1 \le i < j \le n} \omega_{i,j}(t),$$

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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] \;-\; \gamma(t)\sum_{k\neq j}\omega_{i,k}(t)$$

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** :

$$H(t) \;=\; -\sum_{i,j}\; p_{i,j}(t)\,\ln\Big(p_{i,j}(t)\Big).$$

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* :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] \;-\; \gamma(t)\sum_{k\neq j}\omega_{i,k}(t),$$

et on y associe un **feedback** :

$$\gamma(t+1) \;=\; \gamma(t) \;+\; f\big(\mathrm{MesureGlobale}(t)\big),$$

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** :

$$\begin{cases} \omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] - \gamma(t)\displaystyle\sum_{k\neq j}\omega_{i,k}(t), \\ \gamma(t+1) = \gamma(t) + f\big(\text{MesureGlobale}(t)\big), \end{cases}$$

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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\right] \;-\; \gamma\sum_{k\,\neq\,j}\omega_{i,k}(t),$$

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 $\eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$ 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

$$\gamma(t+1) \;=\; \gamma(t) \;-\; \alpha\left[\kappa \;-\; \overline{d}(t)\right],$$

dès lors que $\overline{d}(t) < \kappa$. 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 :

7. Un γ *suffisamment positif* pour **restreindre** la croissance simultanée de multiples liaisons "moyennes" et **forcer** un choix sélectif,

8. 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 **régule** 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 **parsimonie** 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 :

9. **Met à jour** la matrice $\{\omega_{i,j}(t)\}$ à partir du **DSL** (chap. *7.4.1* et *7.4.2*).
10. **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 **parsimonie** : 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 éparse.

- **Avantage** : Lisibilité et calcul facilité. On ne s'occupe plus des liens < θ, 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 :

11. Les liens de très faible amplitude **n'apportent** pas un gain notable dans la fonction d'**énergie** ou d'**affinité** globale.

12. 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** $\mathcal{P}_\theta$:

$$\widetilde{\omega}_{i,j}(t+1) \;=\; \max\{\,\omega_{i,j}(t+1),\,0\,\}\,\mathbf{1}\Big(\omega_{i,j}(t+1) \;\geq\; \theta(t)\Big).$$

(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 :

13. **Seuil fixe** $\theta > 0$: on choisit dès le départ une valeur (petite ou moyenne) qui convient à la dimension $\| \omega \|$.

14. **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 :

15. **Calcul** : on met à jour $\omega_{i,j}(t+1)$ selon la règle de **plasticité** (ex. inhibition, saturation).

16. **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 :

17. **Logarithmique**

$$\theta(t) \;=\; \theta_0\,\big(1 + \alpha \ln(1+t)\big),$$

assurant un **resserrement** lent (la fonction ln croît modérément).

18. **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 :

1. **Exploration** des synergies potentielles.
2. **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'accroît**, 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 :

1. **Mise à jour** :

$$\omega^*_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] \pm \ldots \quad \text{(inhibition, etc.).}$$

2. **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 :

1. **Phase initiale** : $\theta(t)$ faible, ⇒ beaucoup de liaisons au-dessus du seuil, *exploration large* du réseau.
2. **Phase intermédiaire** : $\theta(t)$ commence à s'élever, ⇒ élimination des connexions qui ne se renforcent pas assez.
3. **Phase finale** : $\theta(t)$ est devenu nettement plus strict, ⇒ 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**.

1. **Linéaire** : la *tension* monte régulièrement, précipitant un tri plus *brusque* à mesure que t s'élève.
2. **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 :

3. **Un contrôle** plus fin de la structure du SCN (par exemple, en limitant explicitement le nombre de liens),
4. **Des solutions** plus globales (en combinant la descente DSL avec des méthodes d'exploration plus large, comme les algorithmes génétiques),
5. **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 locale** : 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** :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t+1), & \text{si } j \in \mathrm{TopK}(i, t+1), \\ 0, & \text{sinon.} \end{cases}$$

Ici, $\mathrm{TopK}(i, t+1)$ 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 $O(n\,k)$ 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 nœud "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 alourdissant 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 nœud $\mathcal{E}_i$ 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 :

6. **Trie** $\{\omega_{i,j}\}_{j=1..10}$ par ordre décroissant.
7. **Conserve** les 3 plus grands liens.
8. **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 :

9. Elle **limite** la densité, passant de $O(n^2)$ à $O(n\,k)$.
10. Elle **clarifie** le SCN, renforçant la **sélectivité** et accélérant la formation de **clusters**.

11. 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

$$\omega_{i,j}(t+1) \;=\; \begin{cases} \omega_{i,j}(t+1), & \text{si } j \in \mathrm{TopK}(i, t+1), \\ 0, & \text{sinon.} \end{cases}$$

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 $\mathrm{O}(n^2)$ arêtes mais plutôt $\mathrm{O}(n\,k)$. 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

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\,\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] \pm \cdots$$

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 nœud 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 W** résumant toutes les composantes $\omega_{i,j}$ pour $1 \le i < j \le n$.

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 discrétiser 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

$$\mathbf{W} = \left(\underbrace{b_{1,2}^1, \dots, b_{1,2}^q}_{\text{bits de } \omega_{1,2}}, \dots, \underbrace{b_{i,j}^1, \dots, b_{i,j}^q}_{\text{bits de } \omega_{i,j}}, \dots \right),$$

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 $\dim \approx n^2$. 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 **W**, on définit une **fonction de coût** $\mathcal{J}(\mathbf{W})$ 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 :

$$\text{Fitness}(\mathbf{W}) = \frac{1}{1+\mathcal{J}(\mathbf{W})} \quad \text{ou} \quad -\mathcal{J}(\mathbf{W}),$$

de sorte que **minimiser** $\mathcal{J}(\mathbf{W})$ équivaut à **maximiser** la fitness.

Les **individus** de l'algorithme génétique se voient ainsi **comparés** selon la valeur de $\mathcal{J}(\mathbf{W})$, 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 W** dans ce GA se présentera sous la forme :

$$\mathbf{W} = \big(\omega_{1,2},\ \omega_{1,3},\ \omega_{1,4},\ \omega_{2,3},\ \omega_{2,4},\ \omega_{3,4}\big).$$

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** :

12. On part d'un individu **W**.
13. On applique quelques pas de la règle **DSL** ($\omega_{i,j}(t+1) = \cdots$) pour obtenir un **minimum local** $\mathbf{W}^*$.
14. 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** $\boldsymbol{\omega} = \{\omega_{i,j}\}$, 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 $\mathrm{Fitness}(\mathbf{v}) = -\mathcal{J}(\mathbf{v})$, 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 :

15. **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).

16. **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.

17. **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** :

18. Chaque individu subit quelques itérations de la descente DSL, pour parfaire localement sa matrice ω.

19. Les meilleurs individus sont ensuite pris pour opérer un crossover, une mutation.

20. 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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$. Puis, à la fin de ces T itérations, un **module génétique** vient "relire" l'état $\boldsymbol{\Omega}$ 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 $\boldsymbol{\Omega}$ (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 $\boldsymbol{\Omega}$), 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 $\mathcal{J}\big(\boldsymbol{\Omega}(t)\big)$ 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 $\boldsymbol{\Omega}(t)$ entière comme un **chromosome C**.

L'**AG** (algorithme génétique) sélectionne deux "individus" (matrices $\boldsymbol{\Omega}^{(p)}, \boldsymbol{\Omega}^{(q)}$) et opère un "crossover" :

$$\mathbf{\Omega}_{i,j}^{(\text{child})} = \begin{cases} \mathbf{\Omega}_{i,j}^{(p)}, & \text{avec probabilité 0.5,} \\ \mathbf{\Omega}_{i,j}^{(q)}, & \text{avec probabilité 0.5,} \end{cases}$$

ou un schéma plus sophistiqué (moyenne, bloc...). Puis on **mute** certains liens $\mathbf{\Omega}_{i,j} \leftarrow \mathbf{\Omega}_{i,j} + \delta$ aléatoirement.

On peut *remplacer* la configuration courante du SCN par $\mathbf{\Omega}^{(\text{child})}$ 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 $\mathbf{\Omega}$ 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 $\mathcal{J}(\mathbf{\Omega})$ 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** :

21. **Descente locale** : le SCN, via sa mise à jour $\omega(t+1) = \cdots$, affine la configuration,
22. **Exploration globale** : l'AG (ou la colonie de fourmis, ou tout autre) brasse l'espace $\{\omega\}$ de façon plus large,
23. **Phase d'interaction** : régulièrement (ou en parallèle), l'AG propose une "mutation" ou un "crossover" qui **remanie** la matrice $\mathbf{\Omega}$, 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 $\mathcal{J}(\mathbf{\Omega})$ (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) :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] \;\pm\; \ldots$$

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 $\mathbf{\Omega}^{(k)}_{\text{final}}$. On répète ainsi :

$$\text{Run } k \quad \rightarrow \quad \{\omega_{i,j}(t)\}_{t=0}^{T_k} \quad \text{avec initialisation} \quad \omega_{i,j}^{(k)}(0).$$

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 $\mathbf{\Omega}^{(k)}_{\text{final}}$. 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** $\{\mathbf{\Omega}^{(1)}_{\text{final}}, \ldots, \mathbf{\Omega}^{(R)}_{\text{final}}\}$. On **compare** alors leurs valeurs de la fonction $\mathcal{J}(\mathbf{\Omega})$:

$$\boldsymbol{\Omega}_{\text{best}} = \arg\min_{k\in\{1,\dots,R\}} \mathcal{J}\left(\boldsymbol{\Omega}_{\text{final}}^{(k)}\right),$$

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 $\boldsymbol{\Omega}_{\text{best}}$ 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)}}\mathrm{final}} $ 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 **descente 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 $\mathcal{J}(\Omega)$, 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** $\mathcal{J}\left(\Omega^{(r)}\right)$ (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** $\mathcal{J}$ ou **maximise** la modularité. Si l'on définit un **score** $Q(\Omega)$, comme un indicateur de qualité du clustering, la sélection s'effectue selon :

$$\Omega^{(r^*)} \quad \text{tel que} \quad r^* = \arg\max_{r\in\{1,\dots,R\}} Q\left(\Omega^{(r)}\right).$$

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 :

$$\omega_{i,j}^{(\text{consensus})} = \frac{1}{R}\sum_{r=1}^{R} \omega_{i,j}^{(r)} \quad \text{ou} \quad \text{med}\left\{\omega_{i,j}^{(1)}, \dots, \omega_{i,j}^{(R)}\right\}.$$

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 atténuées 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 $\mathbf{\Omega}^*$ 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 $\mathbf{\Omega}^{(k)}$, 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 $\mathbf{\Omega}^{(k)}$ 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 $\mathcal{J}\left(\mathbf{\Omega}^{(k)}\right)$, 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 $\boldsymbol{\Omega}^*$ 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 :

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

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 $\mathrm{N}\big((n+1), k\big)$, 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 :

$$\text{While new data arrives:} \quad \text{AddEntity}(\mathcal{E}_{n+1}), \quad \text{PartialUpdate}\big(\{\omega_{(n+1),j}\}\big),$$

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 $\mathrm{N}\big((n+1),k\big)$ désigne le **voisinage** de $\mathcal{E}_{n+1}$, il se peut que $\omega_{(n+1),\,j}$ demeure à 0 pour tout $j \notin \mathrm{N}\big((n+1),k\big)$. 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 **incréments** de données.

E. Avantage Comparé à un Batch Complet

L'**approche en flux** présente plusieurs atouts. Elle évite une **réinitialisation complète** ou un recalcul intégral à chaque nouvelle entité, garantissant ainsi une **scalabilité accrue**, puisqu'elle ne nécessite que le traitement de O(k)\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** $S\big(\mathcal{E}_i, \mathcal{E}_j\big)$ 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 :

$$\omega_{(n+1),j}(t+1) = \omega_{(n+1),j}(t) + \eta\left[S\left(\mathcal{E}_{n+1}, \mathcal{E}_j\right) - \tau\, \omega_{(n+1),j}(t)\right] \quad \text{uniquement pour } j \in \mathcal{V}(\mathcal{E}_{n+1}).$$

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 $\mathrm{O}(|\mathcal{V}|)$ au lieu de $\mathrm{O}(n)$.

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 $\mathrm{dist}\left(\mathcal{E}_{n+1}, \mathcal{E}_j\right)$ (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

$$\mathcal{V}(\mathcal{E}_{n+1}) = \left\{j \mid \mathrm{dist}\left(\mathcal{E}_{n+1}, \mathcal{E}_j\right) < \epsilon\right\}.$$

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** $\mathrm{O}(n)$ 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 $\mathrm{O}(n^2)$. Cela conserve le **principe** de mise à jour :

$$\omega_{(n+1),j}(t+1) = \omega_{(n+1),j}(t) + \eta\left[S\left(\mathcal{E}_{n+1}, \mathcal{E}_j\right) - \tau\, \omega_{(n+1),j}(t)\right].$$

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 nœud $\mathcal{E}_j$ 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 nœuds 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** $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ 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** $\mathcal{E}_{\text{new}}$ 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) :

$$\omega_{(\text{new}),\,j}(t+1) = \omega_{(\text{new}),\,j}(t) + \eta\big[S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) - \tau\,\omega_{(\text{new}),\,j}(t)\big],$$

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 $\mathcal{E}_{\text{new}}$ 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 $\mathcal{E}_{n+1}, \mathcal{E}_{n+2}$, ... 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 $\mathcal{E}_{\text{old}}$.

Mathématiquement, on introduit une **règle** de désactivation : si un nœud $\mathcal{E}_{\text{old}}$ 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 éparse (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 $\mathcal{J}(\omega)$ 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 $\mathcal{E}_{\text{new}}$ 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,

Persistance 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** $\mathcal{E}_{\text{new}}$ 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 $\mathcal{E}_{\text{new}}$ de façon cohérente au réseau préexistant, tout en maîtrisant la **complexité** et en préservant la **stabilité** des clusters déjà formés. La fonction 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 $\mathcal{E}_{\text{new}}$, 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 $\mathcal{E}_{\text{new}}$ 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 $\mathcal{E}_{\text{new}}$ 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 $\alpha\, n$ 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 $\mathcal{E}_{\text{new}}$. 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 $\mathcal{E}_{\text{new}}$ 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 $\mathcal{E}_{\text{new}}$ poursuit sa trajectoire dans la **dynamique** DSL : pour chaque lien retenu (ou éventuellement créé a posteriori), on itère

$$\omega_{(\text{new}),\,j}(t+1) = \omega_{(\text{new}),\,j}(t) + \eta\big[S(\mathcal{E}_{\text{new}},\, \mathcal{E}_j) - \tau\, \omega_{(\text{new}),\,j}(t)\big].$$

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 $\mathcal{E}_{\text{new}}$. 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 ($\omega_{\mathcal{E},j} \to 0$ 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 :

$$\omega_{\mathcal{E},j}(t+1) = \omega_{\mathcal{E},j}(t) + \eta\big[S(\mathcal{E},j) - \tau\,\omega_{\mathcal{E},j}(t)\big] + \Delta_{\text{remove}}(t),$$

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 $\Delta_{\text{remove}}(t)$:

24. 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.

25. 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 nœuds $\{j\}$ 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 nœuds 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 $\{\mathcal{E}_{n+1}, \ldots\}$ 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($\mathcal{E}_{\text{new}}$), voir § 7.6.2.1) : on calcule et initialise $\omega_{(\text{new}),j}(0)$ pour un sous-ensemble de nœuds $\{j\}$, puis on enclenche la dynamique DSL sur ces liaisons.

- **Retrait** (removeEntity($\mathcal{E}_{\text{old}}$), voir § 7.6.2.2) : on réduit $\omega_{(\text{old}),j}$ vers 0 sur plusieurs itérations, impliquant une réorganisation des clusters précédemment liés à $\mathcal{E}_{\text{old}}$.

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 $\boldsymbol{\omega}(t)$ 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 nœuds,
- 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 $\ell_{\max}$ 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 $O\big((20+10)\times k\big)$ 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 "sèquencée" (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 localement 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 appliqué 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \Delta_{\text{noise}}(i,j,t),$$

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 $\boldsymbol{\omega}$. 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 $\boldsymbol{\omega}(t)$ 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 paramétrer. 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 $\mu\, n$ (quelques pourcents de l'ensemble total). On exécute la règle DSL usuelle :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big] + \Delta_{\text{noise}}(i,j,t).$$

D'un point de vue **théorique**, on espère qu'avec une amplitude σ "raisonnable", cette perturbation **conduit** la suite $\{\boldsymbol{\omega}(t)\}$ 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 nœuds) 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 $\mathcal{C}_\alpha$ (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 $\mathcal{C}_\alpha$ reste identique ou quasi identique, sans entité entrant ni sortant.

Une **variance** ou une **énergie** interne $\mathrm{Var}(\mathcal{C}_\alpha, t)$ 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 $\mathcal{C}_\alpha$ 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** $\mathcal{C}_\alpha$ 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 reconsidération 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** $\mathcal{C}_\alpha$. Le cluster concerné peut être effacé ou perturbé, mais sans compromettre l'**historique des liaisons** dans le reste du réseau. Cette approche **sélective**, 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 **détecter cette non-réactivité**, de réinitialiser $\mathcal{C}_\alpha$, 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 $\mathcal{C}_\alpha$ 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 nœuds) se **fige** trop longtemps — en dépit de l'évolution du reste du réseau ou de l'arrivée de nouveaux nœuds —, 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 nœuds 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 $\mathcal{C}_\alpha$ 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 $\mathcal{C}_\alpha$ 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 $\mathcal{C}_\alpha$ 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 :

26. **Injections stochastiques** périodiques (voir § 7.6.3.1),
27. 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 :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] \;+\; \ldots$$

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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \Delta_{\text{reward}}(i,j,t),$$

avec

$$\Delta_{\text{reward}}(i,j,t) = f\big(\mathcal{R}(t),\, \omega_{i,j}(t),\, \dots\big).$$

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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \alpha\,\big(\mathcal{R}(t)\big) \times g(i,j),$$

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 $g(i,j) = 1/2\,\big(\mathcal{R}_i(t) + \mathcal{R}_j(t)\big)$ 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 $\big(S(i,j),\, \omega_{i,j},\, \mathcal{R}(t)\big)$, 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 **paramétrer 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

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

peut être étendue en ajoutant un **terme** associant la **récompense** $\mathcal{R}(t)$. Par exemple :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \alpha\,\Delta_{\text{reward}}(i,j,t).$$

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 **paramétrer** 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 $\mathcal{J}_{\mathcal{R}}$ à la fonction de coût, conduisant à un **potentiel** global $\mathcal{J}_{\text{total}} = \mathcal{J}_{\text{Synergy}} + \mathcal{J}_{\mathcal{R}}$. 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 **oriente** 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 $\{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$. 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 ≡ Agents, Liaisons ≡ 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 :

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

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** $\mathcal{R}_i(t)$. 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 $\big(\alpha_i^{(t)}, \alpha_j^{(t)}\big)$. Cela peut être une fonction $f\big(\alpha_i, \alpha_j, \dots\big)$ dépendant de la situation courante.

Au fil de l'itération temporelle $t \to t+1$, on applique :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S_{i,j}(t) - \tau\,\omega_{i,j}(t)\big] \quad \text{(forme de base).}$$

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 prédéfinies**, 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) context, 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** $(\mathcal{A}_1, \ldots, \mathcal{A}_n)$, 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 prédéfinis. 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** :

28. **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.

29. **Inhibition** : on accentue la compétition entre liaisons, poussant le SCN à "choisir" seulement quelques connexions par agent, rendant la topologie plus éparse (cf. chap. 7.4).

30. **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** $\mathrm{state}_m(t)$, 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 $\mathrm{state}_m(t)$: 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 :

$$Q_m(\mathrm{state}_m, \alpha_m) \leftarrow Q_m(\mathrm{state}_m, \alpha_m) + \alpha \left[\mathcal{R}_m(t) + \gamma \max_{\alpha'} Q_m(\mathrm{state}_m{}', \alpha') - Q_m(\mathrm{state}_m, \alpha_m)\right].$$

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 :

$$\omega_{m,n}(t+1) = \omega_{m,n}(t) + \eta\big[S_{m,n}(t) - \tau\,\omega_{m,n}(t)\big],$$

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, $\omega_{m,n}$ décroit.

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-$\mathcal{C}_\alpha$ 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\mathrm{DSL}}\big(\omega_{i,j}(t),\ \{\alpha_i, \alpha_j\},\ \mathcal{R}\big),$$

$$\pi_i(t+1) = \pi_i(t) + \Delta_{\mathrm{RL}}\big(\pi_i(t),\ \mathrm{Reward}_i,\ \omega_{i,\cdot}\big).$$

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** $(\boldsymbol{\omega}^*, \pi^*)$.

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 $\omega_{i,\cdot}$ 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 nœuds 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 **coefficient** 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 influe-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 **estim**er 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** $\mathcal{J}(\Omega)$ finale : on compare la valeur finale $\mathcal{J}\big(\Omega(t_{\text{final}})\big)$. 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 prématurée 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 $\mathcal{J}$ 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** $\mathcal{J}\left(\Omega(t_{\text{final}})\right)$ 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 prématurée).

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

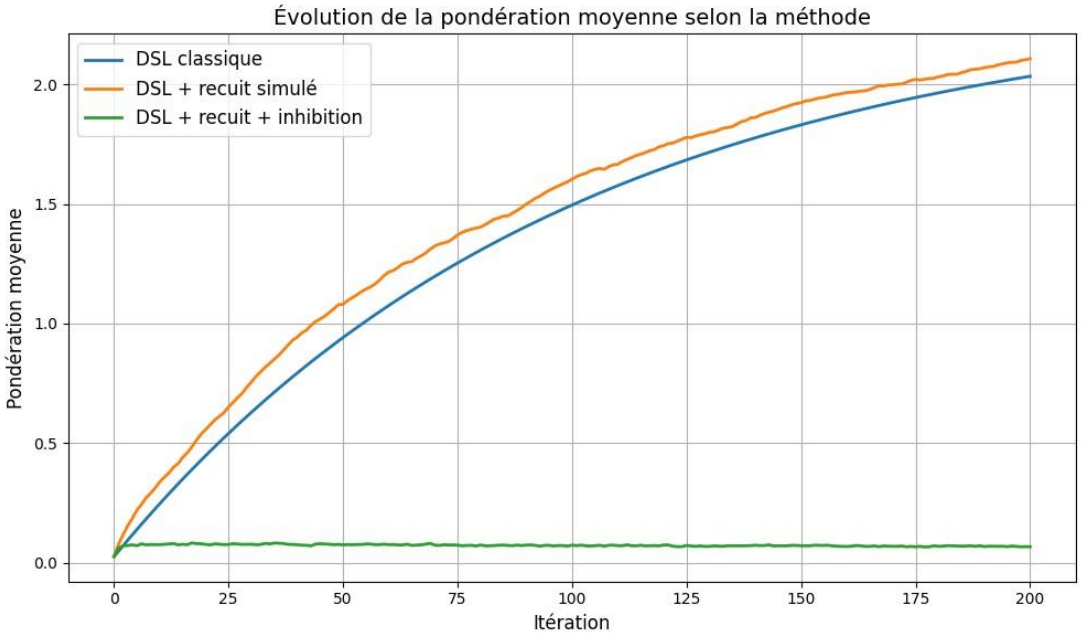
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Modèle simplifié de Deep Synergy Learning (DSL)

Nous considérons un réseau de n = 30 entités, chacune représentée par un vecteur
dans ℝ². La fonction de synergie entre deux entités est définie par
    S(E_i, E_j) = exp(-α * ||x_i - x_j||)
pour les entités du groupe 0 et, pour celles du groupe 1, on ajoute un bonus
    S(E_i, E_j) = exp(-α * ||x_i - x_j||) + δ_bonus.
La mise à jour des pondérations est donnée par :
    ω_{ij}(t+1) = ω_{ij}(t) + η [ S(i,j) - τ ω_{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
```

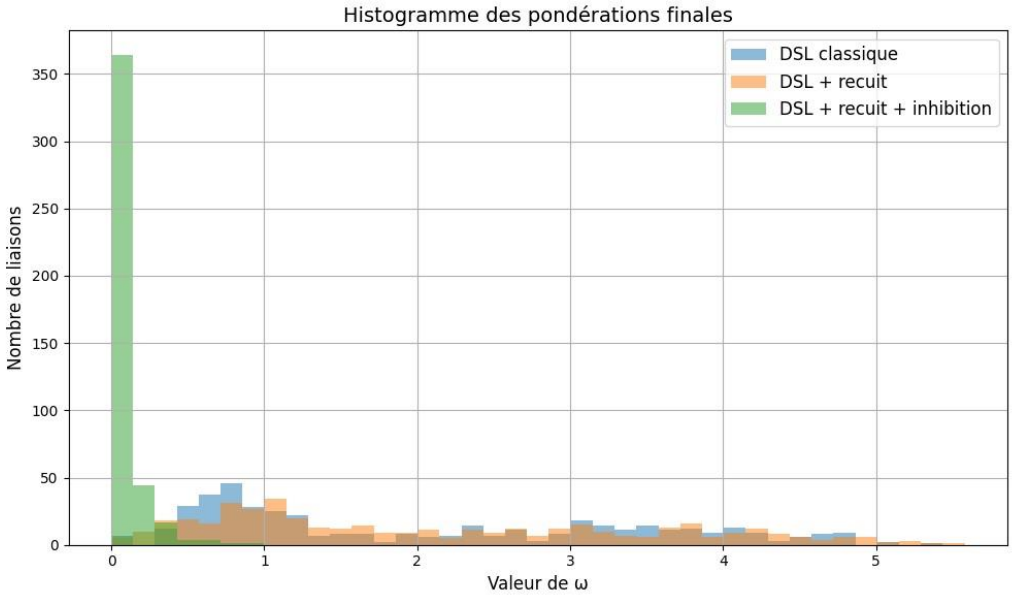
```
np.random.seed(42)

### 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 (Ω^(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 ℝ².
# 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 ω_{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(𝔼_i, 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):
```

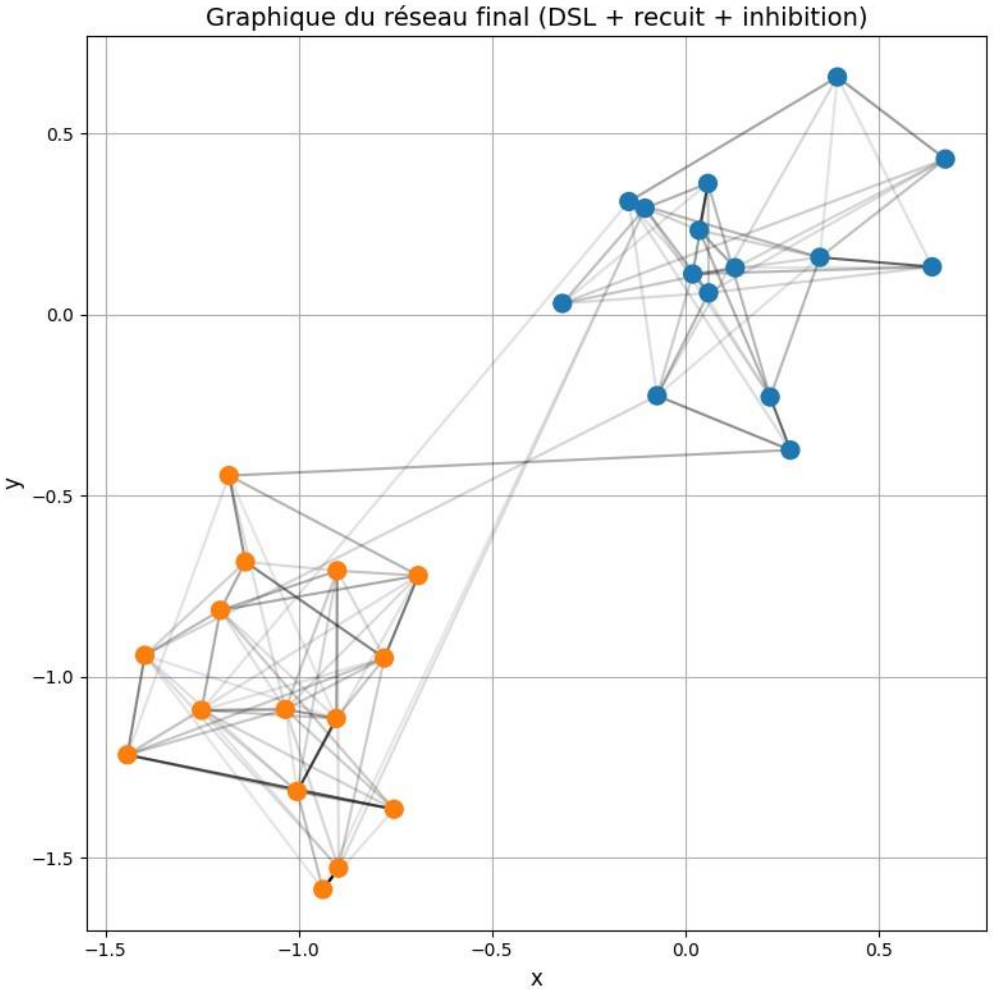
```
    """
    Construit la matrice de synergie S de taille (n,n).
    """
    n = X.shape[0]
    S_mat = np.zeros((n, n))
    for i in range(n):
        for j in range(i+1, n):
            s_val = synergy(i, j, X, labels, alpha, delta_bonus)
            S_mat[i, j] = s_val
            S_mat[j, i] = 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_recuit, bins=bins, alpha=0.5, label="DSL + recuit")
plt.hist(weights_recuit_inhib, bins=bins, alpha=0.5, label="DSL + recuit + inhibition")
plt.xlabel("Valeur de ω", 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 ℝ², en colorant selon le groupe, et on affiche les liaisons avec ω > 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_recuit_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_recuit_inhib[i,j]/final_W_recuit_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()
```




Histogramme des pondérations finales
DSL classique
DSL + recuit
DSL + recuit + inhibition
Nombre de liaisons
Valeur de ω
0
50
100
150
200
250
300
350
0
1
2
3
4
5


Graphique du réseau final (DSL + recuit + inhibition)
y
x
0.5
0.0
−0.5
−1.0
−1.5
−1.5
−1.0
−0.5
0.0
0.5

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 signifiantes.

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 éparse.

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 $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right]$, 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 $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ 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 :

31. **Énergie finale** $\mathcal{J}\left(\mathbf{\Omega}^{(\text{final})}\right)$ (ou pseudo-énergie),
32. **Temps** (ou nombre d'itérations) de **convergence**,
33. **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 : $\mathcal{J}(\boldsymbol{\Omega}^{(final)})$

Dans le cadre du DSL (Chap. 7.2.1), on définit une **énergie** $\mathcal{J}$ (ou un coût) reflétant :

$$\mathcal{J}(\boldsymbol{\omega}) = -\sum_{i,j} \omega_{i,j}\, S(i,j) + \frac{\tau}{2}\sum_{i,j} \left[\omega_{i,j}\right]^2 + \cdots$$

(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** $\boldsymbol{\omega}^{(\text{final})}$. On évalue $\mathcal{J}\left(\boldsymbol{\omega}^{(\text{final})}\right)$. 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 $\mathcal{J} \approx \mathcal{J}\left(\Omega^{(2)}\right)$ 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** (recalcul 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 :

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

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 $\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$. 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

$$\Delta\omega_{i,j}(t) = \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$$

risque d'être **excessive**, engendrant des dépassements 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 nœuds 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),

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

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** $\tau \Longrightarrow$ amortissement trop ténu, liens susceptibles d'**osciller** et de se comporter de façon saccadée,
- **Trop fort** $\tau \Longrightarrow$ 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 nœud i : plus γ est grand, plus un nœud 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\, \omega_{i,j}(t)\big] - \gamma \sum_{k \neq j} \omega_{i,k}(t).$$

Le **terme** $-\gamma \sum_{k \neq j} \omega_{i,k}(t)$ reflète l'**inhibition latérale** : lorsqu'un nœud 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 :

34. **Phase initiale tolérante** : commencer avec γ faible pour permettre au réseau d'explorer de nombreux liens.
35. **Phase finale plus sélective** : augmenter γ afin de forcer la compétition, aboutissant à un **réseau** plus économe.

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 nœud) assez faibles, mettant en évidence un **petit** nombre de connexions vraiment "utiles". Une γ élevée encourage cette parcimonie : chaque nœud 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 nœud 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 nœud 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** $\gamma \Longrightarrow$ réseau **dense**, chaque entité peut cultiver plusieurs liaisons.
- **Fort** $\gamma \Longrightarrow$ **parcimonie** plus marquée, conduisant un nœud 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 chronométrer 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 : $\mathcal{J}(\boldsymbol{\Omega})$

La **descente** DSL s'interprète fréquemment comme la **minimisation** (locale) d'une **fonction** d'énergie. Un schéma générique serait :

$$\mathcal{J}(\boldsymbol{\Omega}) = -\sum_{i,j} \omega_{i,j}\, S(i,j) \;+\; \frac{\tau}{2}\sum_{i,j} \left[\omega_{i,j}\right]^2 \;+\; \ldots$$

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** $\mathcal{J}(\mathbf{\Omega})$ 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 $\mathcal{J}$ à l'issue de la convergence (la **version** DSL classique vs. une version enrichie en recuit ou heuristiques). Une **différence** notable de $\mathcal{J}$ 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 $\mathcal{J}\left(\mathbf{\Omega}^{(\text{final})}\right)$. Si la dynamique simple reste piégée dans un arrangement à $\mathcal{J} \approx -100$ et que la version recuit aboutit à $\mathcal{J} \approx -120$, 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 $\{\mathcal{C}_1, \ldots, \mathcal{C}_k\}$ 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 $\mathcal{J}$ 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 $\mathcal{J}$ 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 :

- $\mathcal{J}$ indique si on a atteint un minimum local/globalseLon la descente DSL,
- 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 $\mathcal{J}$ forme la base "interne" pour évaluer la **descente** 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** $\Delta\mathcal{J}(\boldsymbol{\Omega}(t))$ 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 notoire 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…), 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 $\omega_{\max}$ 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 épaisse. On voit ainsi la formation de **clusters** dans l'espace projeté.

Tracer la courbe $\mathcal{J}\big(\omega(t)\big)$ 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 prédéterminée. La **dynamique DSL** est ensuite exécutée avec ou sans **recuit simulé** ou heuristique d'optimisation, et la règle d'évolution des poids suit :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \xi_{i,j}(t),$$

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\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 $\mathcal{J}^{(1)}$.

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 $\mathcal{J}^{(2)}$ est souvent proche de $\mathcal{J}^{(1)}$, 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** $\mathcal{J}(\boldsymbol{\Omega}^*)$ est l'un des principaux critères permettant de juger l'optimalité d'un état atteint. Elle peut être définie par la fonction $\mathcal{J}$:

$$\mathcal{J}(\boldsymbol{\Omega}) = -\sum_{i,j} \omega_{i,j}\, S(i,j) + \frac{\tau}{2} \sum \omega_{i,j}^2.$$

Lorsque plusieurs exécutions de la dynamique aboutissent à des valeurs similaires $\mathcal{J}^{(1)}, \mathcal{J}^{(2)}, \mathcal{J}^{(3)}$, 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 :

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

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 décroit, 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 {**ω**}, 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 $\mathcal{J}$ 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 $\mathcal{J}(\boldsymbol{\Omega}(t))$ 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'**évitement 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. $t = 0{,}50{,}100, \ldots$), on met en évidence la manière dont les blocs de **connexions fortes** se structurent et comment certaines **liaisons disparaissent progressivement**. Un SCN bien organisé révèle 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 $\mathcal{J}(t)$ 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 agressive** 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 $\mathcal{J}$ 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 $\mathcal{J}$ (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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta \left[S(R_i, R_j) - \tau\, \omega_{i,j}(t) \right] + \Delta_{\text{(inhibition ou bruit)}}$$

où η correspond au taux d'apprentissage, τ reflète la décroissance basique, et $\Delta_{\dots}$ 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(R_i, R_j) - \tau\,\omega_{i,j}(t)\big].$$

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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] + \sigma(t)\,\xi_{i,j}(t)$$

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 $-\sum\omega_{i,j}S(i,j) + \cdots$. 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'explore 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\omega_{i,j}(t)\big] - \gamma \sum_{k\neq j} \omega_{i,k}\,(t).$$

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 prématuré** 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 piégeages 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-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 ωi,j\omega*{i,j} devient trop élevée, l'augmentation de la somme* ∑*kωi,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 **concepts** 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** $\mathcal{E}_{\text{sub}}$, 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** $\mathcal{E}_{\text{sym}}$, par exemple un ensemble de **règles**, **axiomes** ou **concepts** 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" $\mathcal{E}_{\text{mix}}$, 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S_{\text{hyb}}(i,j) - \tau\,\omega_{i,j}(t)\right] + \Delta_{(\text{inhibition, recuit, etc.})}$$

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 $S_{\text{sub}}(\mathcal{E}_i, \mathcal{E}_j) = \exp(-\alpha \parallel \mathbf{x}_i - \mathbf{x}_j \parallel)$, 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 $\mathcal{J}(\omega)$. La dynamique DSL (décrite 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 $\mathcal{J}$ 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. $\mathcal{R}_\alpha$ dit "si A alors B", $\mathcal{R}_\beta$ 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 $\mathcal{R}_\alpha$ et $\mathcal{R}_\beta$ 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 $(i, j) \in \mathcal{B} \times \mathcal{B}$:

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S_{\text{hyb}}(i,j) - \tau\,\omega_{i,j}(t)\right] + \sigma(t)\,\xi_{i,j}(t).$$

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 :

$$\omega_{\alpha,\beta}(t+1) = \omega_{\alpha,\beta}(t) + \eta\left[S_{\text{sym}}(\alpha,\beta) - \tau\,\omega_{\alpha,\beta}(t)\right] - \gamma \sum_{\beta' \neq \beta} \omega_{\alpha,\beta'}(t).$$

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 **ambiguïtés** 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'ancrage 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

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S(i,j) \;-\; \tau\,\omega_{i,j}(t)\right]$$

) 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 unifié**, 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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] + \kappa\,\mathcal{R}(t)\,\Phi(i,j,t).$$

- $\mathcal{R}(t)$: un **signal** extrinsèque, pouvant être global (même valeur pour tous) ou local (une portion de la récompense associée à certains nœuds).

- $\Phi(i,j,t)$: 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. À 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 $\Phi(i,j,t) \approx \left[\mathcal{R}_i(t) + \mathcal{R}_j(t)\right]$ 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 :

$$\omega_{(n+1),j}(0) = \delta \times S\left(\mathcal{E}_{n+1}, \mathcal{E}_j\right)$$

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 $\mathcal{E}_{\text{old}}$ 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 Incrémentale (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** $\omega_{\max}$ 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-modale** 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 ($\mathcal{E}$_vision, $\mathcal{E}$_audio, $\mathcal{E}$_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 :

$$\omega_{(n+1),j}(t+1) = \omega_{(n+1),j}(t) + \eta \left[S\left(\mathcal{E}_{n+1}, \mathcal{E}_j\right) - \tau\, \omega_{(n+1),j}(t)\right].$$

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 evolutionnaires" 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 métaheuristiques 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 dimensionalité 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** $\mathbf{\Omega} = \{\omega_{i,j}\}$ 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 $\mathbf{\Omega} \in \mathbb{R}^d$. À chaque itération :

$$\mathbf{\Omega}^{(p)}(t+1) = \mathbf{\Omega}^{(p)}(t) + \mathbf{v}^{(p)}(t+1),$$

où $\mathbf{v}^{(p)}(t+1)$ 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 $\mathbf{\Omega}$. On définit une **fonction d'évaluation** :

$$\text{Fitness}(\mathbf{\Omega}) = -\mathcal{J}(\mathbf{\Omega}),$$

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 :

$$\omega_{i,j}(t) \quad \leftrightarrow \quad \text{Taux de phéromone sur le lien } (i,j),$$

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 $\eta\big[S(i,j) - \tau\omega_{i,j}\big]$, 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. Hybridations 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 contraint un nœud 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 $\eta\big[S(i,j) - \tau\omega_{i,j}\big]$) puis, toutes les K itérations, on lance un cycle "PSO global" (ou "Fourmis") manipulant $\boldsymbol{\Omega}$ 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 $\mathbb{P}(R \mid a)$ 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

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

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 $\eta\left[S(\mathcal{E}_i,\mathcal{E}_j) - \tau\,\omega_{i,j}\right]$, 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,

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S(\mathcal{E}_i,\mathcal{E}_j) \;-\; \tau\,\omega_{i,j}(t)\big]_{\text{DSL local}} \;+\; \Omega_{\text{RL}}\Big(\omega_{i,j}(t),\,\mathcal{R}(t)\Big)_{\text{action RL}},$$

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 $\mathcal{J}(\omega)$. 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

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

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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\left[S(i,j) - \tau\,\omega_{i,j}(t)\right] + \underbrace{\Psi(\mathcal{H}, \boldsymbol{\Omega}, t)}_{\text{terme heuristique global}},$$
$$\text{composante DSL}$$

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 *méta*correction (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 :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}\left(\omega_{i,j}(t)\right) + \Delta_{\text{Global}}\left(\omega_{i,j}(t), \Omega_{\text{all}}\right),$$

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 ($\eta\left[S(i,j) - \tau\omega_{i,j}\right]$) 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.