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

Chapitre 9 : Évolutions Temps Réel et Apprentissage Continu

9.1. Introduction

Dans ce neuvième chapitre, nous abordons la **dimension temps réel** du Deep Synergy Learning (DSL) et l'**apprentissage continu** au sein du Synergistic Connection Network (SCN). Les chapitres antérieurs (3 à 8) ont exploré tour à tour la **représentation** des entités (ch. 3), la **dynamique** d'auto-organisation (ch. 4), l'**architecture** générale (ch. 5), la **multi-échelle** (ch. 6), les **optimisations** (ch. 7) et l'**approche multimodale** (ch. 8). Nous allons désormais étendre ces principes au contexte où le **réseau** ne se limite pas à un instant figé, mais évolue **en continu**, confronté à des **flux** de données, des **ajouts** ou **retraits** d'entités, et des changements de contexte permanents.

9.1.1. Contexte Général

9.1.1.1. Rappel : après avoir posé les bases d'un SCN (Ch. 5) et exploré le multi-échelle (Ch. 6), les optimisations (Ch. 7) et le multimodal (Ch. 8), on s'intéresse désormais à la capacité du DSL à fonctionner en continu

Dans le cadre du **Deep Synergy Learning** (**DSL**), il a été nécessaire, au fil des chapitres précédents, de bâtir une structuration solide du **Synergistic Connection Network** (**SCN**). Ce réseau s'articule autour d'un **noyau central** qui stocke et met à jour les pondérations $\{\omega_{i,j}\}$. L'architecture modulaire présentée au **Chapitre 5** a rendu possible la configuration d'**interfaces**, de **mécanismes d'inhibition** et de **calculs de synergie**, de telle sorte que l'on puisse introduire ou retirer des entités $\mathcal{E}_k$ sans devoir entièrement déconstruire le réseau. Cette souplesse structurelle se révèle cruciale pour toute évolution en continu, puisqu'un SCN "monolithique" impliquerait un recalcul intégral à chaque insertion ou suppression.

Dans la poursuite de cette logique, le **Chapitre 6** a mis en évidence la notion de **multi-échelle** et la survenue de configurations potentiellement fractales, où des **clusters** locaux peuvent former des **macro-structures** plus globales. Cette **hiérarchisation** facilite une gestion localisée du réseau lorsque de nouvelles entités apparaissent ou lorsque certaines disparaissent. Ainsi, au lieu de modifier la totalité de la matrice ω, on se concentre sur les structures impactées. Cette approche modulaire et multi-échelle aide à limiter les recalculs et à maintenir, en temps réel, la **cohérence** de l'ensemble des pondérations.

L'étape suivante, décrite au **Chapitre 7**, a consisté à approfondir les **méthodes d'optimisation**. Ces techniques incluent des processus de **recuit simulé**, des **heuristiques globales** et des schémas d'**inhibition avancée**, l'objectif étant d'éviter qu'un SCN ne tombe prématurément dans un **minimum local**. Dans un environnement en évolution continue, la nécessité de conserver une certaine **plasticité** est encore plus marquée. Une fois que le réseau commence à "vieillir" ou à se stabiliser, il faut savoir réinjecter un **bruit contrôlé** ou ajuster la force de la **compétition** pour s'adapter à de nouvelles entités ou à des variations de **synergie**.

Le **Chapitre 8**, pour sa part, a montré à quel point l'intégration **multimodale** (vision, texte, audio, etc.) accroît la complexité des règles d'**auto-organisation**, surtout si l'on envisage des flux de données qui arrivent de manière ininterrompue. L'hétérogénéité des types de signaux s'avère potentiellement variable dans le temps . De nouveaux capteurs ou de nouveaux formats apparaissent, et le réseau doit accepter et traiter cette diversité de manière incrémentale. Les mécanismes décrits dans les chapitres 5 à 8 restent donc valables, mais ils doivent être

complétés par des procédures d'**insertion** et de **retrait** d'entités en temps réel, ainsi que par une mise à jour dynamique du calcul de **synergie**.

À la lumière de ces éléments, le **Chapitre 9** se consacre à la capacité du DSL à se maintenir opérationnel et **flexible** dans un cadre de **fonctionnement en continu**. On s'intéresse, par exemple, à la règle d'évolution

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \Delta\omega_{i,j},$$

où $\Delta\omega_{i,j}$ dépend à la fois de $\omega_{i,j}(t)$ et de $S(i,j)$, mais aussi des phénomènes d'**arrivées** et de **disparitions** d'entités, de l'existence de **coûts de vieillissement** (ou **forgetting**), et d'éventuels **signaux de rétroaction** macro-structurelle (voir références au Chap. 10). Cette formalisation vise une situation d'**online learning**, ou d'**apprentissage continu**, où la donnée n'est plus simplement un ensemble fixe, mais un **flux** dont les caractéristiques peuvent évoluer au fil du temps.

Dans un contexte de **robotique**, d'**IoT** ou d'**analyse de flux multimédia**, il est rare de pouvoir conserver un **dataset** statique. Le SCN doit donc réaliser l'**auto-organisation** tout en incorporant les **nouveaux** signaux et en retravaillant la matrice $\{\omega_{i,j}\}$ de manière localisée. En ce sens, on identifie deux **exigences fondamentales**. D'une part, il est essentiel de **maintenir une mémoire** pour les entités anciennes dont les **synergies** restent pertinentes. D'autre part, le réseau doit être capable d'une **évolution plus radicale** lorsque se produit un **changement profond** dans la structure ou la nature des données. Cette logique d'incrémentation et d'adaptation continue prolongera naturellement les enseignements des précédents chapitres en les inscrivant dans une optique de **streaming** ou de **scénario temps réel**.

En conclusion, l'ensemble des briques théoriques et pratiques présentées dans les chapitres 5, 6, 7 et 8 fournit un **socle** pour comprendre comment le SCN peut être **construit** et **stabilisé** dans un environnement statique ou faiblement changeant. Le **Chapitre 9**, quant à lui, montrera comment ce **réseau** peut rester "vivant" et s'adapter à des variations incessantes, en absorbant de nouvelles entités $\mathcal{E}_k$, en **actualisant** les synergies $S(i,j)$ de façon incrémentale et en procédant à des réorganisations locales ou globales. Cette perspective se révèle incontournable dans de nombreux domaines de l'intelligence artificielle, de la **robotique** et des **systèmes distribués**, où la structure des données ne peut être considérée comme figée et où l'**auto-organisation** doit opérer dans un flux dynamique, potentiellement multimodal et en perpétuelle expansion.

9.1.1.2. Motivations : environnements changeants, afflux de nouvelles entités, "online learning" ou streaming de données

Les contextes d'application du **Deep Synergy Learning (DSL)** s'étendent souvent à des **environnements** instables, soumis à un flux ininterrompu de nouvelles données et de nouvelles entités. Le **Synergistic Connection Network (SCN)** doit alors gérer l'arrivée ou la disparition de ces entités, et adapter **incrémentalement** ses liaisons $\omega_{i,j}$. Cette section expose les raisons pour lesquelles un simple entraînement statique ne peut suffire, en soulignant la nécessité d'un apprentissage en continu ou en **online learning**.

Dans la pratique, la plupart des systèmes évoluent dans un cadre où les conditions externes ou internes se modifient au cours du temps. Un **robot**, par exemple, affronte des **contextes variés** tels que la **luminosité**, la **présence d'obstacles**, la **nature des tâches** à accomplir ou encore les **préférences des utilisateurs**. Dans un marché financier, les signaux contextuels (valeurs,

corrélations, volatilité) changent de structure au fil des événements économiques. Si l'on se bornait à un apprentissage ponctuel et figé, le **SCN** resterait calé sur d'anciens modèles, incapables de capturer l'émergence de nouvelles **synergies** ou la disparition d'interactions précédemment établies. La formule

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

rend explicite la dépendance temporelle de la **synergie** $S\big(\mathcal{E}_i, \mathcal{E}_j, t\big)$. Dans un monde changeant, cette fonction peut être amenée à varier dans le temps, reflétant l'évolution de la distribution des données ou des caractéristiques des entités. Pour rester pertinent, le réseau n'a pas d'autre choix que de s'adapter en continu, sans "redémarrer" tout le processus.

Dans de nombreux secteurs, l'ensemble des entités $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ croît ou fluctue. De nouveaux utilisateurs arrivent sur un site de recommandation, de nouveaux documents se créent dans une base de connaissances, de nouveaux modules capteurs sont ajoutés à un système IoT. L'**online learning** demande alors d'étendre **localement** la matrice $\{\omega_{i,j}\}$ aux entités ajoutées, de leur attribuer des pondérations initiales $\omega_{(\text{new}),\,j} \approx 0$, puis de laisser la règle de mise à jour les faire évoluer. L'enjeu principal est de ne pas réentraîner depuis zéro l'intégralité des pondérations existantes, mais plutôt de limiter les calculs à ce qui concerne l'entité nouvellement insérée. Cette insertion incrémentale doit cependant demeurer cohérente avec les **clusters** ou **macro-structures** déjà en place. Une fois la période initiale d'adaptation passée, la pondération $\omega_{(\text{new}),\,j}$ se stabilise aux valeurs dictées par la **synergie** $S\big(\mathcal{E}_{\text{new}}, \mathcal{E}_j\big)$. D'un point de vue algorithmique et mathématique, cela empêche que le coût de reconstruction du **SCN** ne devienne prohibitivement élevé à mesure que le nombre total d'entités n s'accroît.

Streaming de données et flux infini. Dans certains environnements, comme la surveillance continue de signaux (capteurs, logs, séquences conversationnelles), le flux de données est potentiellement **infini**. Il devient impossible de tout stocker ou de procéder à un recalcul complet de la matrice ω à chaque nouvelle arrivée de données. Le **SCN** doit alors incorporer chaque échantillon ou mini-lot de données de manière incrémentale. En notation mathématique, on peut décrire un processus stochastique $\{\omega(t)\}_{t\geq 0}$ qui, à chaque itération t, reçoit un nouvel événement ou un nouveau paquet X_t et met à jour de façon locale les pondérations associées. Si la **synergie** $S(i,j)$ s'appuie elle-même sur des statistiques d'occurrences ou de co-occurrences, ces estimateurs doivent être maintenus sous forme glissante ou partielle, sans recomptage global. Cette perspective de "**online learning**" confère au réseau un caractère perpétuellement adaptable. À chaque instant, il est prêt à recevoir l'information suivante, à mettre à jour ω, puis à poursuivre son rôle d'**auto-organisation** dans un flot de données qui ne s'arrête jamais.

Les **motivations** qui précèdent (environnements à forte variabilité, arrivée dynamique d'entités, flux potentiellement infini) forment un socle conceptuel justifiant l'exigence d'un **SCN** résolument "vivant". Au-delà de l'intérêt théorique, cette capacité se révèle cruciale dans les systèmes de recommandation en temps réel, la gestion de tâches évolutives en robotique ou l'exploitation de grands volumes de données multimédias où l'utilisateur souhaite s'appuyer sur l'**auto-organisation** pour extraire des clusters ou des patterns révélateurs. Les mécanismes décrits dans les parties ultérieures du chapitre détailleront la manière dont un **SCN** peut mettre en œuvre des stratégies d'inhibition, de recuit partiel ou de mise à jour locale de ω pour ne pas se laisser submerger par l'instabilité de l'environnement et la croissance potentiellement illimitée du nombre d'entités.

9.1.2. Objectifs

Au sein de ce **chapitre 9**, l'accent est mis sur la **capacité** du DSL (Deep Synergy Learning) à **s'adapter** de façon permanente, dans des situations où le réseau n'est jamais figé, mais continuellement sollicité par de **nouvelles entités** ou de **nouvelles données** en flux. Il s'agit d'un enjeu crucial pour des **systèmes temps réel**, qu'il s'agisse de robotique, de capteurs en ligne, de traitements multimédias, etc. Nous visons notamment :

- **Décrire l'adaptation permanente** : (9.1.2.1)
- **Montrer comment éviter la "catastrophic forgetting"** ou les minima locaux figés : (9.1.2.2)
- **Illustrer des applications** concrètes en robotique, multimédia, etc. : (9.1.2.3)

9.1.2.1. Décrire l'adaptation permanente : mise à jour incrémentale, insertion/suppression d'entités, auto-réorganisation

L'**adaptation permanente** du **Synergistic Connection Network** (**SCN**) en contexte **DSL** implique la capacité de mettre à jour la matrice $\{\omega_{i,j}\}$ de manière **incrémentale**, d'**insérer** ou de **supprimer** des **entités** $\mathcal{E}_i$ au fil du temps, et de déclencher une **auto-réorganisation** lorsque la structure du réseau se trouve modifiée. Cette approche se distingue nettement d'un apprentissage statique. Au lieu de recalculer entièrement les pondérations existantes, on se limite à des ajustements **locaux**, gage de **flexibilité** et de **résilience** du SCN dans des environnements dynamiques (voir également [9.1.2.2] et [9.1.2.3]).

Un **SCN** où le nombre d'**entités** $\{\mathcal{E}_i\}_{i=1,\dots,n}$ varie au cours du temps requiert des règles d'**insertion** et de **suppression** adaptées. Lorsqu'une entité $\mathcal{E}_{n+1}$ survient, on procède à un **calibrage initial** des nouvelles pondérations $\omega_{(n+1),\,j}$ et $\omega_{j,\,(n+1)}$ (pour tout j). Ces valeurs, souvent **faibles** ou déterminées par un **k plus proches voisins**, alimentent ensuite la **règle** d'évolution incrémentale habituelle, par exemple

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

où la **synergie** $S\left(\mathcal{E}_i, \mathcal{E}_j\right)$ peut dépendre de mesures de similarité (cf. Chap. 5 et 8) ou de statistiques de co-occurrence. L'important est de n'**impacter** que les liaisons associées à l'entité ajoutée, en évitant de **recalculer** l'intégralité de la matrice ω, ce qui serait prohibitif pour de grands réseaux.

En **flux continu**, il arrive qu'une entité $\mathcal{E}_k$ devienne **obsolète** (capteur défaillant, document périmé, etc.). La **suppression** consiste alors à **diminuer** progressivement les pondérations associées $\omega_{k,j}$ et $\omega_{j,k}$, voire à les annuler afin de libérer de la ressource et de clarifier la **structure** émergente. On assiste alors à une **auto-réorganisation** où les **clusters** ou **groupes** du réseau s'adaptent pour refléter l'absence de l'entité retirée. Sur le plan **algorithmique**, cette opération reste locale et n'affecte directement que la composante liée à $\mathcal{E}_k$, tandis que les autres liaisons poursuivent leur évolution selon le schéma principal.

Par ailleurs, lorsque le **nombre** d'entités s'accroît sans limite, le risque d'une **complexité** en $O(n^2)$ sur la mise à jour est réel. Pour y remédier, on recourt à des **mécanismes de parcimonie**, tels que la conservation des seuls **k** plus forts liens ou l'application d'un rayon ϵ en distance (cf. Chap. 7). Ainsi, la mise à jour ne concerne qu'un **voisinage** restreint plutôt que l'ensemble

des paires $\omega_{i,j}$. Une telle **sparsification** sécurise la **scalabilité** du SCN, tout en préservant la majorité des **interactions** jugées essentielles.

A. Exemple symbolique

On peut illustrer ces mécanismes par un **système de recommandation** dans lequel de **nouveaux utilisateurs** apparaissent en continu. Lors de l'arrivée d'un utilisateur $\mathcal{E}_{\text{new}}$, la matrice ω s'étend d'une ligne et d'une colonne ; on initialise alors $\omega_{\text{new},k} \approx 0$ pour tous les items k inconnus. On peut aussi spécifier des pondérations un peu plus élevées pour les items jugés proches selon un critère de similarité. Ensuite, chaque **mise à jour** incrémentale (fondée sur la synergie entre $\mathcal{E}_{\text{new}}$ et les items connexes) consolide les **liens** formés, sans altérer outre mesure la structure déjà présente dans le SCN. Ainsi, la **réorganisation** demeure **localisée** et économise des ressources de calcul.

B. Motivation forte en temps réel

Dans un contexte de **temps réel** (ex. robotique, flux vidéo, IoT), on ne peut pas **stopper** l'activité du réseau pour "redémarrer" un entraînement complet à chaque insertion ou retrait d'entité. Grâce au principe d'**adaptation incrémentale**, le **DSL** perpétue la **philosophie** d'auto-organisation. Les pondérations $\omega_{i,j}$ continuent leur évolution

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \ldots$$

en considérant simplement que les indices (i,j) s'élargissent ou se restreignent au fil du temps. L'adaptabilité est alors **permanente**, assurant la **cohérence** du SCN face à des perturbations successives et maintenant la structure en phase avec l'évolution des données ou des entités en présence.

Conclusion (9.1.2.1). L'**adaptation permanente** d'un **SCN** dans le cadre du **DSL** repose sur une **mise à jour incrémentale** de la matrice ω, l'**insertion** et la **suppression** d'entités de manière locale, ainsi que sur une **auto-réorganisation** assurant la continuité de l'**auto-organisation** globale. Ces mécanismes permettent d'aborder des **environnements** dynamiques sans réapprentissage complet, tout en conservant un **contrôle** de la **dimension** (via la parcimonie) et une **souplesse** de recalibrage indispensable pour des applications en flux ou en temps réel. La suite (sections [9.1.2.2] et [9.1.2.3]) examinera plus avant la gestion de la mémoire et la question du "catastrophic forgetting" dans ce cadre évolutif.

9.1.2.2. Montrer comment éviter le "catastrophic forgetting" ou les minima** locaux statiques**

Dans un scénario d'**apprentissage continu**, le **Synergistic Connection Network** (**SCN**) se confronte à un flux potentiel de **nouvelles entités** $\{\mathcal{E}_{n+1}, \ldots\}$, ou à des **changements de contexte** (apparition de nouveaux patterns de similarité, modification des conditions externes). On redoute deux **écueils majeurs**. Le premier est le **catastrophic forgetting**, où la **structure** bâtie précédemment s'effondre sous l'effet de données récentes trop envahissantes, conduisant à une perte d'information acquise. Le second est la stagnation dans un **minimum local statique**, où le réseau néglige les signaux inédits et peine à s'adapter à de nouvelles configurations, compromettant ainsi la flexibilité du **DSL** face à l'évolution des flux de données. Cette section [9.1.2.2] détaille quelques **mécanismes** mathématiques et algorithmiques visant à **protéger** la mémoire acquise tout en **encourageant** l'adaptation aux nouveautés.

A. Techniques pour Éviter le "Catastrophic Forgetting"

Une première classe d'approches vise à éviter que les liens $\omega_{i,j}$ précédemment acquis ne s'évaporent quand un flux de nouvelles données ou entités arrive.

On peut doter la **règle de mise à jour** d'un **terme de régularisation** qui "rappelle" la configuration ancienne. Soit $\omega_{i,j}^{(\text{old})}$ les valeurs de $\omega_{i,j}$ apprises sur l'historique passé. Une forme simple de régularisation est alors

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S_{\text{new}}(i,j) \;-\; \tau\,\omega_{i,j}(t)\right] \;-\; \lambda\Big(\omega_{i,j}(t) \;-\; \omega_{i,j}^{(\text{old})}\Big),$$

où $\lambda > 0$ paramètre la **force** de préservation. La partie $\lambda\Big(\omega_{i,j}(t) - \omega_{i,j}^{(\text{old})}\Big)$ joue un rôle de **rappel**, empêchant le **SCN** de s'éloigner excessivement des liens stabilisés auparavant, et donc de **détruire** la structure acquise.

Il est aussi possible, dans un **réseau modulaire**, de réserver des **sous-blocs** dédiés à la réception des nouvelles données, afin d'éviter que celles-ci ne réécrasent intégralement le noyau existant. Ainsi, les nouveaux **clusters** se forment dans un espace de liens partiellement découplé, et les **liens historiques** $\omega_{i,j}^{(\text{old})}$ restent relativement **protégés**. On retrouve cette philosophie dans certaines architectures "multi-têtes" ou dans des partitions auto-organisées (voir chap. 7 et 8).

Dans un SCN **multi-niveau** (cf. chap. 6), un **macro-nœud** (ou un niveau supérieur de la hiérarchie) peut exercer un **feedback** préservant les clusters anciens. Concrètement, lorsque la mise à jour locale menace de diluer un ensemble de liaisons réputées cruciales, le macro-nœud impose un seuil minimal $\omega_{\min}$ ou un léger "forçage" de la pondération, évitant la disparition rapide d'un cluster clé. Cela limite l'effet du "**catastrophic forgetting**" car la structure mémorisée dans ces clusters reste protégée par cette rétroaction top-down.

B. Échapper aux Minima Locaux Statiques

Un second risque apparaît lorsque le **SCN** se trouve **trop stable**, ignorant les nouveautés. Plusieurs **outils** permettent de lever ce blocage.

Si la dynamique

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

est trop "paisible", il peut se produire une **saturation** de certains clusters. On introduit alors un **bruit** $\sigma(t)\,\xi_{i,j}(t)$, analogue à un "recuit simulé" (voir chap. 7.3), donnant :

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

Le coefficient $\sigma(t)$ module la "température". De faibles valeurs protègent la stabilité, tandis que des valeurs plus grandes réinjectent du **hasard**, cassant les attracteurs trop figés. Le **SCN** peut ainsi **repartir** à la recherche d'une nouvelle configuration plus adéquate lorsque le contexte évolue.

La **croissance** de liens "moyennement forts" peut enfermer le réseau dans un **état** où aucune liaison ne parvient à émerger clairement. En introduisant un **mécanisme** d'inhibition compétitive (chap. 7.4), on favorise les liens qui deviennent très élevés au détriment des autres. Cela peut se formuler par un **terme d'inhibition** $I\big(\omega_{i,j}, \omega_{i,k}\big)$ dans la mise à jour, incitant un nœud à conserver seulement un sous-ensemble restreint de connections réellement pertinentes.

Ce procédé évite que le **SCN** reste bloqué dans un **minimum local**, incapable de former de nouveaux clusters en réponse aux entités entrantes.

On peut, en outre, prévoir des **tests** de stagnation. Dès lors qu'un sous-ensemble de liaisons $\omega_{i,j}$ n'a plus évolué depuis un certain nombre d'itérations (faible vigilance), on applique un "**reset**" partiel, telle une réduction par un facteur $\alpha < 1$. Cette opération brise la **fixité**, rendant de nouveau possibles les ajustements selon les nouveaux signaux. Dans une équation globale, on modifie ponctuellement

$$\omega_{i,j}(t+1) \;\leftarrow\; \alpha\,\omega_{i,j}(t),$$

pour des groupes de nœuds identifiés comme "non évolutifs". Cet effacement partiel permet de **réallouer** la capacité de clusterisation là où elle est requise.

C. Synthèse Mathématique et Algorithmique

Les mesures décrites ci-dessus se traduisent par des **termes complémentaires** insérés dans la mise à jour. Un **terme mémoire** $-\lambda\left(\omega_{i,j}(t) - \omega_{i,j}^{(\text{old})}\right)$ permet de contrer l'oubli en maintenant une trace des pondérations passées. Un **terme de bruit** $\sigma(t)\,\xi_{i,j}(t)$ introduit une perturbation contrôlée pour éviter que le réseau ne se fige dans un état sous-optimal. Enfin, un **reset** ponctuel peut être appliqué si la vigilance tombe sous un certain seuil, permettant ainsi de réactiver des liens potentiellement sous-exploités. L'équation DSL "étendue" prenant en compte ces mécanismes s'écrit alors :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\left[S_{\text{new}}(i,j) - \tau\,\omega_{i,j}(t)\right] \;-\; \lambda\left(\omega_{i,j}(t) - \omega_{i,j}^{(\text{old})}\right) \;+\; \delta_{\text{noise}}(t),$$

où $\delta_{\text{noise}}(t)$ agrège recuit, resets et toute autre perturbation utile. Les **chapitres** précédents (par ex. chap. 6 pour la multi-échelle, chap. 7 pour les heuristiques d'optimisation) ont montré que ce type de **dynamique** peut être géré de façon distribuée ou hiérarchique, grâce à la modularité du **SCN**.

Conclusion (9.1.2.2)

Pour éviter à la fois le **catastrophic forgetting** et les **minima** locaux dans un **SCN**, on s'appuie sur :

- **Des mécanismes de "mémoire"** (régularisation ou allocation partielle) préservant la structure antérieure,
- **Des sources de "perturbation"** (bruit, recuit, resets) libérant le réseau de configurations obsolètes,
- **Des contrôles multi-niveaux** (inhibition, feedback top-down) qui garantissent la plasticité locale sans trahir la cohérence globale.

Ces principes, implémentés dans l'**équation** DSL via des **termes** additionnels et des **protocoles** de vigilance, offrent un **apprentissage continu** plus robuste, ménageant l'**héritage** des clusters stables tout en assurant la **réactivité** aux nouveautés. La prochaine section [9.1.2.3] précisera les enjeux pratiques de la plasticité et du "meta-control" pour maintenir l'équilibre entre mémoire et adaptabilité.

9.1.2.3. Illustrer des applications** : **robotique temps réel**, **systèmes en flux** (capteurs, multimédia)

L'évolution **incrémentale** d'un **SCN** (Synergistic Connection Network) décrite aux sections [9.1.2.1] et [9.1.2.2] se révèle particulièrement précieuse dans au moins deux grands domaines d'application :

- La **robotique temps réel**, où un robot mobilise en permanence des **capteurs** et **actionneurs** et doit ajuster ses liaisons $\omega_{i,j}$ au fil des observations.
- Les **systèmes en flux** (capteurs distribués, multimédia, etc.), caractérisés par des données qui arrivent à un rythme soutenu (souvent continu).

Après avoir identifié les pièges du "**catastrophic forgetting**" et de la stagnation dans un minimum local (voir [9.1.2.2]), on explique ici comment le **DSL** (Deep Synergy Learning) parvient à exploiter ses mécanismes d'**auto-organisation** pour maintenir un **SCN** "vivant" dans ces contextes dynamiques.

A. Robotique Temps Réel : intégration continue des capteurs et actionneurs

De nombreux systèmes robotiques intègrent simultanément plusieurs **capteurs**, tels que des caméras, des ultrasons, des capteurs de force, des unités de mesure inertielle (IMU) comprenant des accéléromètres et des gyroscopes, ainsi que des capteurs LiDAR. Chacun émet un **flux** $\{\mathbf{x}_t\}$ dont la fréquence peut atteindre plusieurs dizaines ou centaines de **Hz**. L'objectif est de **fusionner** ces informations pour la navigation, la planification de trajectoire ou la manipulation d'objets en temps réel.
Le **SCN**, à travers sa **matrice** $\{\omega_{i,j}(t)\}$, doit pouvoir s'**adapter** à ces flux sans recalculer l'intégralité des liaisons à chaque instant. La formulation incrémentale (voir [9.1.2.1]) :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \eta\big[S_t(i,j) \;-\; \tau\,\omega_{i,j}(t)\big] \;+\; \Delta_{\text{inhibition/bruit}},$$

permet, à chaque nouvelle mesure $\mathbf{x}_t$, de **recalculer** la **synergie** $S_t(i,j)$ (corrélation, co-occurrence, etc.) et de l'appliquer **localement**. Cette approche favorise une mise à jour **rapide**, sans latence excessive qui pénaliserait le contrôle du robot.

Lorsqu'un robot passe d'une phase de locomotion (par ex. avancer dans un couloir) à une phase de saisie d'objet, la **synergie** entre les capteurs de vision $\mathcal{E}_{\text{cam}}$ et de force $\mathcal{E}_{\text{force}}$ peut prendre plus d'importance, tandis que la synergie avec d'autres capteurs (GPS, ultrasons) devient secondaire. Le **SCN** reflète ce phénomène en renforçant les liaisons $\omega_{\text{cam,force}}$ lorsqu'il détecte, par exemple, que ces signaux sont **conjoints** dans l'atteinte d'un certain objectif :

$$\omega_{\text{cam,force}}(t+1) \;=\; \omega_{\text{cam,force}}(t) \;+\; \eta\big[S_t(\text{cam},\text{force}) \;-\; \tau\,\omega_{\text{cam,force}}(t)\big].$$

Ce **renforcement** local, rendu possible par l'auto-organisation incrémentale, confère au robot une **réactivité** essentielle. Il mobilise les capteurs utiles au moment voulu, puis redirige ses ressources lorsque la tâche change (cf. [9.1.2.1] pour les principes de mise à jour incrémentale).

Considérons un **bras robotique** cherchant à assembler des pièces. Lorsque le bras déploie son "pinceur" ($\mathcal{E}_{\text{pince}}$) et exerce une pression (capteur de force $\mathcal{E}_{\text{force}}$), la **synergie** $S_t(\text{pince},\text{force})$ augmente, se traduisant par un **gain** de $\omega_{\text{pince,force}}$. Simultanément, la **caméra** $\mathcal{E}_{\text{cam}}$ peut confirmer la bonne position de la pièce, solidifiant un cluster $\{\text{pince},\text{force},\text{cam}\}$. Si la phase suivante requiert un mouvement libre du bras (sans saisie), la synergie $\omega_{\text{pince,force}}$ redescend, réorientant l'activité vers d'autres capteurs (ex. vision de l'environnement).

D'un point de vue **algorithmique**, cette dynamique se poursuit en temps réel, suivant le principe de "**mise à jour incrémentale**" (voir [9.1.2.1]) et les mécanismes de **préservation** ou de **bruit** décrits en [9.1.2.2].

B. Systèmes en Flux : multimédia, capteurs distribués, etc.

Dans les systèmes **multimédia** (vidéo, audio, flux de texte ou de métadonnées), il est fréquent de recevoir en **continu** des paquets ou des segments. Le **SCN** propose de **connecter** chaque nouveau segment $\mathcal{E}_{n+1}$ aux segments antérieurs via la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$. Cette synergie peut reposer sur des mesures de similarité de contenu (features visuelles, signatures audio, sémantique textuelle), de proximité temporelle ou encore de co-occurrence.

Le même raisonnement vaut pour des **capteurs distribués** (capteurs d'humidité, de température, de trafic, etc.) déployés en masse. Les lectures successives forment un flux potentiellement infini. Chaque nouvelle entité (ou nouvel état capteur) s'insère dans la matrice $\{\omega\}$ par :

$$\omega_{(n+1),\,j}(t+1) \;=\; \text{UpdateLocal}\Big(\omega_{(n+1),\,j}(t),\, S_t(n+1, j)\Big),$$

où UpdateLocal représente une **fonction** appliquant la même **formule** DSL mentionnée plus haut. L'important est de ne **mettre à jour** qu'un **voisinage** pertinent (k plus proches voisins, ou ϵ-rayon), pour ne pas exploser la **complexité**.

Supposons qu'on reçoive des **segments vidéo** successifs $\{\mathcal{E}_{\text{vid}_k}\}$. À l'arrivée de $\mathcal{E}_{\text{vid}_{m+1}}$, on calcule une similarité $S(\mathcal{E}_{\text{vid}_{m+1}}, \mathcal{E}_{\text{vid}_k})$ pour quelques segments représentatifs (ceux jugés *proches* en contenu ou en index temporel). Si cette similarité S est élevée, la **pondération** $\omega_{\text{vid}_{m+1},\,\text{vid}_k}$ et son symétrique $\omega_{\text{vid}_k,\,\text{vid}_{m+1}}$ augmentent. Progressivement, le **SCN** peut faire émerger des **clusters** de segments formant un épisode ou un thème cohérent.

Simultanément, des segments anciens (au-delà d'un horizon temporel) peuvent perdre de leur pertinence. On applique alors un **seuil** $\omega_{\min}$ ou un mécanisme de "forgiveness" (ex. *forgetting factor*) pour relâcher ou **supprimer** ces liaisons (cf. [9.1.2.1]). Ainsi, on évite qu'un trop grand nombre de liens ne sature la matrice $\{\omega\}$.

Parce que chaque entité $\mathcal{E}_{n+1}$ est traitée par **incréments**, il n'est plus nécessaire de réaliser un **recalcul global** (coût $O(n^2)$) à chaque nouvelle arrivée. Le **DSL** préserve ainsi le **principe** de l'**auto-organisation** tout en s'adaptant. Des **micro-clusters** se forment localement, la structure globale se met en place sous l'effet des flux successifs, et l'on intègre des stratégies de **parsimonie** ou d'**inhibition compétitive** (voir chap. 7) pour limiter la croissance indésirable de liens "tièdes".

Conclusion (9.1.2.3)

Les cas de la **robotique temps réel** et des **systèmes en flux** montrent à quel point un **SCN** incrémental (cf. [9.1.2.1–2]) constitue un **levier** essentiel dans des environnements où la stabilité doit côtoyer la **réactivité** :

- **Robotique** : Ajuster en direct les synergies entre capteurs et actionneurs (force, vision, etc.) permet de moduler les phases de saisie, de déplacement ou de détection d'obstacles. Le **SCN** agit comme un noyau synaptique évoluant à la cadence des missions du robot.

- **Flux multimédia ou capteurs** : Chaque nouveau segment (vidéo, audio) ou nouvel état de capteur est inséré dans la matrice $\{\omega\}$, ajusté **localement** via la synergie avec les

entités proches. Le réseau évite un recalcul massif, préserve les clusters existants et laisse émerger de nouveaux regroupements pertinents.

Ainsi, l'**apprentissage continu** (chap. 9) apporte au **DSL** toute son utilité pratique dans des domaines réclamant un fonctionnement "**vivant**", c'est-à-dire capable de **s'auto-organiser** et d'**évoluer** conjointement à l'arrivée de nouvelles informations, sans se figer dans un état obsolète ni s'effondrer en cas de surcharge de données. Les aspects de **reconfiguration**, de **mémoire** et de **prévention** du "catastrophic forgetting" (voir [9.1.2.2]) se conjuguent alors pour construire un **SCN** non seulement stable, mais aussi **plastique** et **scalable**, à même de gérer les défis du temps réel et du flux incrémental.

9.1.3. Plan du Chapitre

Après avoir situé le **contexte** (9.1.1) et les **objectifs** (9.1.2) de ce chapitre consacré aux **évolutions temps réel et à l'apprentissage continu** dans le DSL, nous proposons une vue d'ensemble de la structure (sections 9.2 à 9.10). Cela permettra de percevoir la logique de progression, en passant des **fondements** de l'adaptation en flux à la **mise en œuvre** concrète, jusqu'aux **exemples** d'application.

9.1.3.1. Aperçu (sections 9.2 à 9.10)

Afin de guider la lecture du **Chapitre 9** et de clarifier la logique sous-jacente à l'**apprentissage continu** et aux **évolutions temps réel** dans le **DSL** (Deep Synergy Learning), cette section propose un **aperçu** des différents volets qui seront examinés dans les sections [9.2] à [9.10]. L'idée est de souligner la **cohérence** du cheminement. On commence par formaliser la dynamique en flux d'un **SCN** (Synergistic Connection Network), on discute ensuite de l'insertion ou suppression d'entités, de la plasticité de la **synergie**, puis on illustre avec des **cas pratiques** (robotique, conversation). Les comparaisons expérimentales et les éléments de **conclusion** viennent enfin compléter l'ensemble, ouvrant sur les notions de **robustesse** et de **sécurité** (cf. chap. 11).

1. Section 9.2 : Principes de l'Évolution Temps Réel

La section [9.2] pose les bases mathématiques d'un **SCN** mis à jour en continu, c'est-à-dire dans un contexte **non stationnaire**. Nous y explicitons comment la mise à jour $\omega_{i,j}(t)$ ne dépend plus simplement d'un lot de données fixes, mais bien de ce qui advient "**à la volée**" :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \Delta\omega_{i,j}^{(\text{online})}(t),$$

où $\Delta\omega_{i,j}^{(\text{online})}(t)$ est calculé au fur et à mesure que de **nouvelles entités** ou **signaux** apparaissent. Cette section [9.2] insiste sur la nécessaire **localité** de la mise à jour, afin que l'on ne retombe pas dans un recalcul global hors de portée pour de grands volumes de données.

2. Section 9.3 : Stratégies d'Insertion et de Retrait d'Entités

Au fur et à mesure que le temps s'écoule, le **DSL** peut devoir **intégrer** de nouvelles entités $\mathcal{E}_{\text{new}}$ (capteurs, documents, tokens, événements) ou **retirer** celles qui sont obsolètes. La section [9.3] détaille plusieurs méthodes :

- **Approches k-NN** : Initialisation de $\omega_{(\text{new}),\,j}$ par les **k plus proches voisins**, garantissant un **ancrage** local dans le SCN.
- **Voisinage restreint** : Mise en lien de la nouvelle entité avec un **sous-ensemble** de nœuds pertinents, tout en évitant l'explosion du nombre de pondérations.
- **Heuristiques globales** : Parfois, un module "macroscopique" (cf. chap. 6) évalue la meilleure façon de **connecter** l'entité $\mathcal{E}_{\text{new}}$ ou de "débrancher" un nœud vieilli.

Ces stratégies visent à maintenir la **structure** du **SCN** sans engendrer un surcoût énorme, assurant la **scalabilité** de l'approche.

3. Section 9.4 : Adaptation Incrémentale de la Synergie

Lorsque la **fonction** $S(i,j)$ évolue elle-même au cours du temps (par exemple, si le contexte ou les conditions expérimentales changent), il est impératif d'actualiser :

$$S_t(\mathcal{E}_i, \mathcal{E}_j) \quad \text{à chaque itération } t.$$

La section [9.4] examine les implications mathématiques d'une **synergie** $S(i,j,t)$ dépendant de l'instant t. Les **difficultés** portent notamment sur la définition de mesures de similarité ou de co-information adaptées à un **contexte mouvant**. On verra aussi comment intégrer des **estimations en ligne** (ex. moyennes glissantes, statistiques adaptatives) pour alimenter le calcul de la synergie dans un flux.

4. Section 9.5 : Mini-Recuit Périodique et Évitement de Stagnation

Comme déjà discuté dans [9.1.2.2], le **SCN** peut **stagner** dans un **minimum local** si aucune perturbation ne vient périodiquement "casser" les liens quasi stables. La section [9.5] propose la **technique** du "**mini-recuit**" :

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

où $\sigma\,\xi_{i,j}(t)$ constitue un **bruit** de faible amplitude injecté à intervalles réguliers. Cette stratégie, inspirée du **recuit simulé**, empêche la structure de se figer durablement et favorise l'émergence de nouveaux clusters lorsque les données changent de nature.

5. Section 9.6 : Contrôle Top-Down Continu

La section [9.6] explore les mécanismes de **feedback** descendant (top-down) qui, dans un **SCN** multi-niveau (chap. 6), permettent à un **macro-nœud** :

$$\Delta_{\text{down}}(i,j,t)$$

d'influencer localement la mise à jour $\omega_{i,j}$. Par exemple, si le **macro-nœud** détecte une violation d'une contrainte globale (surcharge, cluster trop large, etc.), il envoie un signal $\Delta_{\text{down}}(i,j,t)$ forçant la **réduction** de $\omega_{i,j}$ ou, au contraire, imposant son maintien au-dessus d'un certain seuil. Ce contrôle top-down se superpose au principe d'**auto-organisation** locale, enrichissant la gestion du réseau en temps réel.

6. Section 9.7 : Cas Robotique et Multi-Agent

Dans la section [9.7], nous approfondissons l'exemple déjà effleuré en [9.1.2.3], à savoir un **essaim de robots** collaboratifs. Chaque robot représente un **nœud** $\mathcal{E}_i$, et les liaisons $\omega_{i,j}$ traduisent le niveau de coopération ou d'échange d'informations. On analyse :

- Comment le réseau s'**auto-réorganise** quand un robot tombe en panne,
- Comment on **incorpore** un robot supplémentaire sans perturber outre mesure la structure existante,
- Les **indicateurs** quantitatifs (temps de convergence, robustesse, utilisation des ressources).

7. Section 9.8 : Domaines Conversationnels et Linguistiques

La section [9.8] projette ces mécanismes dans un **cadre conversationnel** où un agent dialogue avec un utilisateur. Les **mots**, **expressions** ou **actes de dialogue** constituent autant d'entités connectées. Au fil du dialogue, on voit apparaître de **nouveaux** segments lexicaux, de nouvelles intentions, etc. Le **SCN** doit donc effectuer un apprentissage "en flux" du **vocabulaire**, et la synergie entre termes évolue en fonction de la conversation :

$S_{\text{ling}}(w_i, w_j, t)$ (similitude sémantique dynamique, co-occurrences temporelles…).

On illustre alors l'intérêt d'un **SCN** incrémental pour s'adapter à la **variabilité** des sujets, prévenir le **catastrophic forgetting** (voir [9.1.2.2]) et, dans certains cas, "conserver" la mémoire des thèmes passés si l'utilisateur y revient plus tard.

8. Section 9.9 : Comparaisons Expérimentales

Les techniques développées depuis la section [9.2] jusqu'à [9.8] sont soumises à des **comparaisons empiriques** dans [9.9]. Les points clés abordés :

- **Qualité** des solutions "online" vs. "batch" : évalue-t-on la cohérence des clusters, la performance de la recommandation, etc.
- **Coût de calcul** : mesure-t-on le temps CPU requis pour intégrer une nouvelle entité vs. refaire un apprentissage global.
- **Robustesse** : test d'éventuelles perturbations, réinitialisations partielles (mini-resets).
- **Graphiques**, **tableaux** : évolution de $\omega_{i,j}(t)$ dans un flot continu, nombre moyen de liens actifs, taux de clustering.

9. Section 9.10 : Conclusion

La section [9.10] boucle le chapitre en rappelant :

- Les **bénéfices** de l'**apprentissage continu** résident dans son adaptation, sa souplesse et sa capacité à éviter la surcharge de recalcul.

- Les risques (catastrophic forgetting, stagnation) et comment ils sont **contenus** grâce aux techniques de recuit, de mémorisation partielle, d'inhibition, etc. (voir [9.1.2.2] et [9.5]).

- Les **perspectives** vers le chapitre 11, où se posent des questions de **sécurité** et de **fiabilité** dans des environnements réellement changeants (protection contre la dérive, détection d'attaques, etc.).

Conclusion (9.1.3.1)

Ce **plan** (sections [9.2] à [9.10]) couvre pas à pas les **thématiques** essentielles pour faire fonctionner un **SCN** en **temps réel** ou dans un **flux continu** :

- **Base théorique** : mise à jour incrémentale des pondérations $\omega_{i,j}$, traitement du contexte non stationnaire ([9.2], [9.4]).

- **Insertion/retrait** d'entités ([9.3]) : modalités, heuristiques k-NN, voisinage restreint.

- **Évitement** de la stagnation et contrôles top-down ([9.5], [9.6]).

- **Cas pratiques** : robotique, multi-agent ([9.7]) et conversationnel ([9.8]).

- **Comparaisons** expérimentales ([9.9]) et **conclusion** générale ([9.10]).

L'ensemble fournit la structure nécessaire pour comprendre la **dynamique** du **DSL** en **apprentissage continu**, préparant les discussions ultérieures (chap. 11) sur la **robustesse** et la **sécurité** des systèmes synergiques en environnement variable.

9.1.3.2. Liens avec chapitres suivants (robustesse/sécurité en Ch. 11, etc.)

Le **Chapitre 9** s'attache à expliciter l'**apprentissage continu** et les **évolutions en temps réel** d'un **SCN** (Synergistic Connection Network) dans le cadre du **DSL** (Deep Synergy Learning). Les sections [9.2] à [9.10] décrivent comment le réseau s'auto-adapte, s'auto-organise et gère l'afflux de nouvelles entités, de nouvelles synergies, ou la disparition d'anciennes composantes. Toutefois, ces mécanismes ne peuvent être envisagés isolément des considérations relatives à la **robustesse**, à la **sécurité** et à la **fiabilité**, qui feront l'objet du **Chapitre 11**. Dans des **environnements dynamiques** ou même hostiles, la faculté d'**adapter** ω de manière fluide (voir [9.1.2]) soulève inévitablement la question de **vulnérabilités** potentielles (données malveillantes, pannes, corruption) et de la nécessité de **contrôles** plus stricts.

A. Convergences entre l'adaptation en temps réel (Chap. 9) et la robustesse (Chap. 11)

L'**auto-organisation** décrite au **Chapitre 9** vise la **flexibilité** : le SCN doit s'ajuster aux modifications (insertion/retrait d'entités, changements de synergie) sans "redémarrer" un apprentissage complet. Or, cette même capacité d'**adaptation** peut être mise à rude épreuve par des perturbations malveillantes ou des séquences de données anormales. Le **Chapitre 11** (sur la robustesse/sécurité) explique comment ajouter des **verrous** ou des **contrôles d'intégrité**. Par exemple, détecter qu'un flux massif d'entités suspectes tente de forcer des liaisons ω fictives, corrompant le réseau.

La **convergence** est donc la suivante. Un SCN "adaptatif" ([9.1.2]) aura besoin de routines de **sécurité** (Ch. 11) pour s'assurer que les micro-changements successifs (cf. [9.1.2.1–2]) n'aboutissent pas à un "envahissement" progressif du réseau par des liens trompeurs.

Les méthodes d'**incrémentation** (voir [9.1.2.1]) et d'**évitement du catastrophic forgetting** (voir [9.1.2.2]) doivent s'accompagner d'une **résilience** vis-à-vis de pannes (p. ex. un nœud capteur défaillant) ou de données bruitées. Le **Chapitre 11** apportera des techniques (surcouche de redondance, watchers ou protocoles de vérification) pour garantir que, même en cas d'erreur sur certains liens, la structure globale ne s'effondre pas. On voit ainsi un **complément** entre la logique d'apprentissage continu (s'adapter au fil de l'eau) et la logique de robustesse (survivre aux incidents ou malveillances).

B. Sécurité et Fiabilité : Quand l'Adaptation devient un Vecteur de Vulnérabilité

Le **Chapitre 9** montre comment, à chaque nouvelle entité, on réévalue $\omega_{(n+1),\,j}$. Si un adversaire (ou des conditions extrêmes) introduit un grand nombre d'entités "piégées", on risque de **détourner** les clusters ou de **fausser** la synergie ω. Le **Chapitre 11** précisera comment détecter ces agressions :

$$\Phi(\mathcal{E}_{\text{new}}) \quad \text{(test d'anomalie, vérification de signature)},$$

avant d'autoriser la mise à jour des poids. Cette étape permet de **filtrer** les entités entrantes et de maintenir la **fiabilité** du SCN sur la durée.

L'**apprentissage continu** ([9.1.2]) implique un horizon temporel vaste, où le réseau enchaîne des milliers (voire plus) d'itérations incrémentales. De petits écarts accumulés (bruit, liens parasites) peuvent, sans garde-fou, se cristalliser en **erreurs majeures** (clusters factices, dérive de la pondération). Le **Chap. 11** discute de **rétrocontrôles** (contrôle d'intégrité, réinitialisations localisées) qui, superposés aux mécanismes de "mini-recuit" ou d'"inhibition" ([9.1.2.2]), assurent une cohérence et préviennent la dérive insidieuse.

C. Autres Perspectives : Chap. 12, 13, etc.

Le **Chapitre 12** introduira les réseaux **n-aires** ou d'ordre supérieur, où la synergie peut impliquer des **groupes** (trois entités et plus). L'**auto-organisation** continue (Ch. 9) s'appliquera alors dans des **configurations** plus complexes, ce qui accentue la nécessité de **contrôles** robustes (Chap. 11) pour repérer des schémas malicieux.

Quand le DSL s'étend vers des fonctions cognitives élevées (voir Chap. 13), il hérite de l'**apprentissage continu** décrit en [9.x] tout en devant faire face à un environnement d'autant plus **imprévisible**. Les thèmes de robustesse et de sécurité (Ch. 11) s'intègrent alors dans une vision holistique de l'intelligence distribuée, où l'on tient compte de la **responsabilité** et de la **fiabilité** face aux dérives potentiellement graves.

Conclusion

Le **Chapitre 9** se consacre à l'**adaptation en temps réel** du SCN et à la **mise en flux continu** de l'**apprentissage**. Cependant, un tel dispositif, s'il n'est **pas** encadré par des **mécanismes** de robustesse (Chap. 11), peut être exposé à des vulnérabilités :

- **Attaques** : flux adversarial saturant ou manipulant la matrice ω.
- **Dérives** : accumulation lente d'erreurs ou de bruit, détériorant la structure sur le long terme.

- **Pannes** : si certains nœuds (capteurs, modules) cessent de fonctionner, il faut une **résilience** au niveau du SCN.

Le **Chapitre 11** proposera donc des **solutions** pour préserver la **cohérence** et la **sécurité**, en articulation avec les idées d'**apprentissage continu** et de **mini-recuit** (voir [9.1.2.2, 9.5]). Ainsi, l'ensemble (Ch. 9 + Ch. 11) dessine un **DSL vivant**, capable de **s'auto-réorganiser** face aux conditions changeantes et **protégé** contre les attaques ou les corruptions, pour une exploitation à grande échelle dans des environnements potentiellement hostiles.

9.2. Principes de l'Apprentissage Continu dans le DSL

Dans un cadre **classique**, l'entraînement d'un **SCN** (Synergistic Connection Network) se ferait "offline". On dispose d'un lot d'entités $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ relativement fixe, on calcule ou initialise la matrice $\{\omega_{i,j}\}$, puis on itère la dynamique DSL (mise à jour des liens) jusqu'à la stabilisation. **Or**, de nombreux **scénarios** (robotique, systèmes de recommandation, flux sensoriels, conversations en évolution, etc.) exigent que le **réseau** s'adapte en **continu**. De **nouvelles** entités apparaissent, les entités existantes peuvent **changer**, ou certaines disparaissent. Le **DSL** doit donc **intégrer** ces arrivées (et disparitions) en temps réel, sans tout "redémarrer" depuis zéro.

C'est dans cette perspective que s'inscrit le **chapitre 9** : il s'agit de **formaliser** l'**apprentissage continu** dans le DSL et d'en décrire les **mécanismes** (flux incrémental, équilibre entre stabilisation et plasticité, etc.). Nous abordons ci-dessous (§9.2.1) la **notion** de flux évolutif, puis (§9.2.2) la comparaison avec l'apprentissage offline, et enfin (§9.2.3) les tensions entre la stabilisation (convergence) et la plasticité (adaptation perpétuelle).

9.2.1. Notion de Flux Évolutif

Le **terme** "flux évolutif" décrit la situation où les **entités** $\mathcal{E}_i$ (et/ou leurs caractéristiques) ne sont pas figées dans le temps, mais arrivent ou se **modifient** selon un **courant** incessant de données.

9.2.1.1. Scénario : des entités $\mathcal{E}_i$ apparaissent dans le temps, ou leurs caractéristiques changent

Il est fréquent, dans un **SCN** (Synergistic Connection Network) évoluant en **apprentissage continu**, de devoir intégrer de nouvelles entités $\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \ldots$ ou de prendre en compte des modifications progressives de certaines caractéristiques. Cette dynamique, explicitée ici en [9.2.1.1], rompt avec le cadre statique et impose une réactualisation permanente des pondérations $\omega_{i,j}$. Les sections A, B et C décrivent tour à tour l'**apparition** incrémentale, le **changement** de propriétés et la gestion d'un **flux** d'entités. Les sections D, E et F en précisent les conséquences, donnent un exemple mathématique minimal et situent la pertinence de la démarche dans divers contextes. La section G conclut ce panorama.

A. Apparition Incrémentale

L'idée d'**apparition** incrémentale se rencontre dans des domaines où, au fil du temps, le réseau se voit enrichi d'entités supplémentaires. Dans un **système** de recommandation, il peut s'agir d'un **nouvel** utilisateur ou d'un **nouveau** document. Il est alors nécessaire de créer les pondérations $\omega_{(\text{new}),j}$ et $\omega_{j,(\text{new})}$ sans recourir à un recalcul exhaustif. Sur le plan mathématique, la taille de la matrice ω passe alors de $O(n^2)$ à $O((n+1)^2)$. Dans la plupart des cas, il demeure impraticable d'évaluer toutes les synergies $S(i,j)$ associées aux paires précédemment établies ; il faut privilégier une **règle** locale. Si l'on note $\mathcal{E}_{n+1}$ la nouvelle entité, on initialise en général $\omega_{(n+1),j}(0)$ à un niveau faible (typiquement ε), puis on applique une **équation** du type

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

la mise à jour se limitant aux index j situés dans un voisinage pertinent de $\mathcal{E}_{n+1}$ (voisinage k-plus-proches-voisins ou rayon ϵ, par exemple).

B. Changement de Propriétés ou Caractéristiques

On rencontre aussi le cas où une entité $\mathcal{E}_k$ existante voit ses **caractéristiques** $\mathbf{x}_k(t) \in \mathbb{R}^d$ évoluer au cours du temps. Il peut s'agir d'un utilisateur révisant ses préférences, ou d'un capteur modifiant sa configuration, voire d'un module qui se calibre différemment. La **synergie** $S(\mathcal{E}_k, \mathcal{E}_j)$ ne saurait alors demeurer figée, car la quantité $\mathbf{x}_k(t)$ détermine l'expression analytique de S. On écrit donc

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

pour signifier la dépendance temporelle. À chaque changement notable de $\mathbf{x}_k(t)$, il est nécessaire de corriger ou de réinitialiser localement les pondérations $\omega_{k,j}(t)$. Dans un **cadre** DSL, on reste cohérent avec la règle de mise à jour habituelle, mais on conserve un œil sur la variabilité de $S(\cdot,\cdot,t)$.

C. Flux d'Arrivée

Les deux situations précédentes peuvent s'inscrire dans un **flux** temporel plus large. On modélise alors la venue de $\mathcal{E}_{n+1}$ à un instant t_1, puis de $\mathcal{E}_{n+2}$ à un instant t_2, et ainsi de suite. Le **SCN** doit s'adapter au rythme du flux, sans exiger un recalcul en $O(n^2)$ à chaque insertion. Le schéma "online" se traduit par une équation DSL locale de la forme

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

où new fait référence à l'entité nouvellement ajoutée. Dans cette logique, la **taille** du réseau grandit de manière progressive, et les mises à jour s'opèrent au fil de l'eau.

D. Conséquence sur la Dynamique

La dynamique DSL décrite au **Chapitre 9** (voir [9.2.3] sur la stabilisation–plasticité) ne saurait aboutir à une **convergence** absolue si des entités s'agrègent en continu. Les clusters peuvent se réorganiser chaque fois qu'une entité nouvelle génère une synergie notable avec un sous-groupe existant. Les sous-ensembles de ω non affectés par ces changements peuvent se stabiliser localement, mais le réseau dans son ensemble conserve un caractère ouvert. Les fluctuations incessantes dessinent un **régime** de stabilisation partielle, plutôt qu'un unique point fixe.

E. Exemple Mathématique Minimal

Il est éclairant de considérer un flux de deux entités successives $\mathcal{E}_{n+1}$ puis $\mathcal{E}_{n+2}$. À l'instant t_1, on insère $\mathcal{E}_{n+1}$. On ajoute dans ω les pondérations $\omega_{(n+1),j}$ et on calcule

$$S(\mathcal{E}_{n+1}, \mathcal{E}_j),$$

pour un ensemble restreint j. On applique la règle DSL locale jusqu'à un pseudo-équilibre. Plus tard, à l'instant t_2, $\mathcal{E}_{n+2}$ se greffe au réseau et réitère un processus analogue. On obtient ainsi un **SCN** dont la dimension croît lentement, sans blocage ni recalcul complet.

F. Utilité dans Différents Contextes

En **robotique**, il est courant de rajouter de nouveaux capteurs ou modules d'actionneur ; le **SCN** peut alors "brancher" ces éléments en insérant leurs pondérations, tout en conservant la structure des liens antérieurs. Dans un **système** de recommandation, un nouveau livre ou un nouveau lecteur se voit intégrer sans rebasculer dans un apprentissage batch. Enfin, tout **flux**

IoT ou logs massifs bénéficie de la même approche. L'entité insérée se connecte localement au réseau, et la règle DSL s'occupe d'en ajuster les coefficients au fil des itérations.

G. Conclusion

La sous-section [9.2.1.1] montre que l'**apparition** d'entités $\mathcal{E}_{n+1}, \mathcal{E}_{n+2}$ ... et la **modification** des caractéristiques d'entités existantes définissent un **scénario** de flux non stationnaire, où la **dynamique** DSL ne vise plus un unique état d'équilibre, mais se maintient dans une plasticité continue. D'un point de vue mathématique, la **matrice** ω grandit et s'adapte localement, sans repasser par un entraînement global, en appliquant la **formule** DSL restreinte à la zone impactée. Ce principe, détaillé plus avant dans [9.3] (stratégies d'insertion/suppression), [9.4] (adaptation incrémentale de la synergie) et [9.5] (éviter la stagnation), s'avère essentiel pour tout **apprentissage continu** ou temps réel envisagé au **Chapitre 9**, et s'inscrit dans la perspective plus large d'un **SCN** capable de "vivre" dans un environnement en constante évolution.

9.2.1.2. Le SCN ne peut pas se recalculer "from scratch" à chaque arrivée ; il faut des règles incrémentales

L'introduction d'entités **nouvelles** (capteurs, images, segments de données, etc.) dans un **SCN** (Synergistic Connection Network) soulève le problème d'un éventuel **recalcul** global de la **matrice** $\{\omega_{i,j}\}$. Cette section [9.2.1.2] montre en quoi cette approche massive serait incompatible avec un **flux** d'arrivée rapide, et pourquoi des **règles incrémentales** s'avèrent nécessaires pour assurer une **mise à jour** locale tout en préservant l'**organisation** existante. Les sous-sections A, B et C exposent successivement les **limites** d'un recalcul total, l'**intérêt** de règles locales, et quelques **exemples** concrets. La conclusion (D) synthétise ces éléments.

A. Limites d'un Recalcul Total

Dans un **SCN** de taille n, la **matrice** ω compte $O(n^2)$ pondérations. Si l'on opte pour un recalcul "**from scratch**" dès qu'une entité $\mathcal{E}_{n+1}$ fait son entrée, il devient nécessaire de réexaminer toutes les **paires** (i, j) de nœuds, au total environ $n^2 + n$. En situation de **flux** à haute fréquence, ce coût $O(n^2)$ se répète à chaque insertion, ce qui engendre une charge inacceptable pour de grands ensembles. D'un point de vue **mathématique**, cette explosion de complexité $O(n^2)$ par nouvelle entité se révèle inabordable dans des domaines tels que la **robotique temps réel** ou la gestion de **logs** massifs.

Par ailleurs, un redémarrage complet du **SCN** à chaque ajout annulerait les bénéfices de la dynamique itérative acquise. L'**historique** accumulé dans ω, les **clusters** déjà formés, et la stabilisation partielle du réseau seraient perdus. Cette interruption de la **dynamique** contradictoire envers l'esprit même du **Deep Synergy Learning**, qui cherche à raffiner ω de manière graduelle et distribuée.

B. Pourquoi des Règles Incrémentales ?

Une **règle** incrémentale circonscrit la mise à jour aux seules zones directement impactées par l'entité $\mathcal{E}_{n+1}$. Lorsqu'un **SCN** reçoit une nouvelle entité, on doit surtout actualiser

$$\omega_{(n+1),\,j} \quad \text{et} \quad \omega_{j,\,(n+1)}, \quad j = 1, \ldots, n,$$

ainsi que quelques liaisons potentiellement affectées via un **feedback** global, si le **macro-nœud** détecte un bouleversement de clusters. Les autres pondérations $\omega_{i,k}$ (pour $i, k \neq n+1$) n'ont

pas de raison fondamentale d'être recalculées dans leur intégralité.
Le **maintien** de l'**organisation** déjà acquise se fait naturellement, puisque seuls les liens attachés à $\mathcal{E}_{n+1}$ sont insérés dans la **matrice** et synchronisés via la formule **DSL** :

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

Cette approche évite le "**reboot**" intégral et garantit une **économie** en temps de calcul. En effet, on ne révise pas $\mathrm{O}(n^2)$ synergies, mais seulement un **voisinage** de taille bornée. Dans le cas où la **synergie** prend la forme $S(i,j) = \exp\big(-\|\,\mathbf{x}_i - \mathbf{x}_j\,\|^2\big)$, il est trivial de constater que la nouvelle entité $\mathbf{x}_{n+1}$ n'interagit réellement qu'avec ses plus proches points.

C. Exemples de Règles Incrémentales

La première étape consiste souvent à **initialiser** $\omega_{(n+1),\,j}(t)$ à une faible valeur (un $\varepsilon > 0$), ou à la calculer directement comme une fonction monotone de la **synergie** :

$$\omega_{(n+1),\,j}(0) = f\left(S\big(\mathcal{E}_{n+1},\,\mathcal{E}_j\big)\right) \quad \text{pour } j \in \mathrm{N}(n+1).$$

Les pondérations pour $j \notin \mathrm{N}(n+1)$ peuvent être nulles, évitant la sur-connexion. On peut ensuite filtrer pour conserver seulement les k plus forts liens sortants, conformément aux stratégies de **parsimonie** usuelles (cf. chap. 7).

Dans certaines architectures **multi-niveau**, un **macro-nœud** pilotant l'ensemble du réseau peut émettre un **retour descendant** pour réorienter ou scinder un cluster si la venue de $\mathcal{E}_{n+1}$ modifie l'équilibre local. Tout cela demeure dans l'esprit d'une **mise à jour** localisée, sans recourir à un recalcul massif de ω.

D. Conclusion

La **nécessité** de règles incrémentales apparaît dès lors que l'on envisage un **SCN** en **flux continu**. Un recalcul global "**from scratch**" à chaque nouvelle entité serait d'un **coût** prohibitif ($\mathrm{O}(n^2)$ par insertion) et détruirait la **dynamique** déjà acquise par la règle DSL. Les **règles** incrémentales, centrées sur les liens $\{\omega_{(n+1),\,j}\}$, conjuguent **économie** de calcul, **cohérence** d'organisation et **préservation** de l'historique. Elles constituent le fondement d'un **apprentissage continu** efficace, où le **SCN** se structure sans cesse, de manière graduelle, conformément aux idées développées en [9.2.1].

9.2.2. Comparaison avec l'Apprentissage Offline

Dans le cadre de l'**évolution temps réel** et de l'**apprentissage continu** (chap. 9.2.1), il est utile de se pencher sur la **distinction** fondamentale entre une **approche offline** — où l'on dispose d'un ensemble de données **fixe** et dont le SCN reste par la suite **inchangé** — et une **approche online**, en flux continu, où la **matrice** $\{\omega_{i,j}\}$ est **actualisée** au fur et à mesure que de **nouvelles** entités ou de **nouvelles** mesures apparaissent.

9.2.2.1. Offline : on dispose d'un dataset fixe, on construit un SCN puis on le fige

Il est souvent utile de considérer un **scénario** où l'on dispose d'emblée d'un **dataset fixe**, c'est-à-dire que l'ensemble des entités $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$ est connu à l'avance et qu'aucune nouvelle information ne viendra compléter ce jeu de données. Dans ce cas, on peut construire le **SCN**

(Synergistic Connection Network) en mode **batch**, puis le **figer** une fois la structure finale atteinte. Les sections A, B, C et D ci-dessous précisent cette démarche, tandis que la section E propose une conclusion (9.2.2.1).

A. Dataset fixe

Le concept de **dataset fixe** suppose qu'un ensemble $\mathcal{D}$ de données, ou de descripteurs associés à des entités, est entièrement disponible avant l'apprentissage. Mathématiquement, on possède $\{\mathcal{E}_i\}_{i=1,\dots,n}$ et on peut donc calculer la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ pour tous les couples (i, j). Dans ce cadre, on ne s'attend pas à l'arrivée d'une entité $\mathcal{E}_{n+1}$ ultérieure. La matrice des **pondérations** $\omega_{i,j}$ peut ainsi s'initialiser et évoluer de façon séquentielle jusqu'à **convergence**, en s'appuyant sur les formules usuelles du **DSL**.

B. Construction statique du SCN

L'étape de **construction** consiste à appliquer la **dynamique** DSL (ou un algorithme d'optimisation approprié) à l'ensemble des paires d'entités. On part de pondérations initiales $\omega_{i,j}(0)$, puis on itère :

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

Les sections précédentes (chapitres 3 et 4) ont déjà décrit cette équation et ses variantes (inhibition, recuit, etc.). L'objectif est de laisser s'opérer une **auto-organisation** interne, jusqu'à ce que la matrice ω atteigne un état quasi-stable. Le **temps** de calcul peut être important car on peut se permettre, dans ce contexte offline, un traitement en $O(n^2)$ si nécessaire.

C. Figeage de la matrice ω

Au terme de la mise à jour itérative, on aboutit à un **SCN** stabilisé. On note alors $\omega^*_{i,j} \approx \omega_{i,j}(\infty)$. Cette matrice, considérée comme "optimale" ou du moins convergée, n'évolue plus. Dans de nombreuses applications, c'est précisément ce dont on a besoin. Un **réseau** de liaisons figé, qu'on peut exploiter pour la détection de clusters, la classification ou la visualisation. Le **Synergistic Connection Network** ainsi construit se compare à n'importe quel résultat d'algorithme batch (k-means, clustering hiérarchique, etc.). On dispose d'un **modèle** qu'on ne prévoit pas de réentraîner sur d'éventuelles données futures.

D. Limites et analogie

Cette philosophie "offline" est **analogue** à des procédures non supervisées classiques, où le dataset ne bouge pas et où l'apprentissage produit un modèle unique et finalisé. S'il est plus simple à implémenter et à analyser, ce mode de fonctionnement présente l'inconvénient de ne pas reconnaître l'existence de **nouvelles** entités ou d'un changement de distribution dans le futur. D'un point de vue **mathématique**, on ferme en quelque sorte la possibilité d'adaptation : un **SCN** offline ne réagira pas à l'insertion d'une entité $\mathcal{E}_{n+1}$, à moins de relancer l'intégralité de l'apprentissage sur un ensemble agrandi.

E. Conclusion

Dans un cadre **offline**, on dispose d'un **dataset fixe** pour lequel on construit le **SCN** et on fige la matrice ω. Cette approche "batch" est stable et familière, mais elle se révèle **rigide**. Aucune nouvelle information n'est intégrée par la suite, au contraire de la version **online** ou **en flux** évoquée en [9.2.2.2] et [9.2.2.3]. Ainsi, lorsque le **Chapitre 9** discute de l'**apprentissage**

continu, il s'agit précisément de pallier la limite de ce "figeage" complet en offrant à la matrice ω la possibilité de s'ajuster aux **changements** de contexte ou d'entités.

9.2.2.2. Online (continu) : on modifie la matrice ω au fil des nouvelles données

Lorsqu'un **Synergistic Connection Network** (**SCN**) est appelé à fonctionner dans un **flux continu** de données, l'approche **batch** décrite en [9.2.2.1] ne suffit plus. Au lieu de figer la matrice ω après un seul apprentissage, on adopte un **mode online**, laissant ω évoluer **en permanence** au rythme de l'arrivée de nouvelles entités ou de nouveaux signaux. Les sections A, B et C analysent les principes de cette mise à jour incrémentale, tandis que D en expose les avantages et contraintes. La conclusion (9.2.2.2) synthétise les enjeux de cette approche online.

A. Philosophie de l'Approche Online

Dans un **flux** continu, qu'il s'agisse de capteurs en temps réel, de données audio-vidéo, d'utilisateurs qui s'enregistrent successivement ou de documents arrivant par lots, la taille du SCN n'est plus figée. Les entités s'ajoutent ou se retirent, et leurs caractéristiques peuvent elles-mêmes fluctuer. Dès qu'une **nouvelle entité** $\mathcal{E}_{n+1}$ apparaît, on l'insère dans la matrice ω, initialisant $\omega_{(n+1),\,j}$ à de faibles valeurs (souvent 0 ou un ε). On procède ensuite au calcul ou à la mise à jour de la **synergie** :

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

Cette logique DSL, déjà décrite dans les chapitres précédents, s'applique en ligne, éventuellement sur un **voisinage** local pour ne pas recalculer $O(n^2)$ liaisons. Par ailleurs, en **mode online**, on tolère que certaines entités tombent en désuétude (capteurs obsolètes, données périmées) et que leurs pondérations $\{\omega_{i,j}\}$ se voient progressivement désactivées ou annulées.

B. Forme Mathématique d'une Mise à Jour Incrémentale

La logique DSL s'adapte à l'arrivée d'un **flux**, en se restreignant souvent à un *voisinage* $\mathrm{N_{new}} \subset \{1, \ldots, n\}$. On écrit :

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

Ce faisant, seules quelques liaisons $\{\omega_{(n+1),\,j}\}$ sont créées ou ajustées. Dans les variantes plus complexes, on peut ajouter un **terme** d'inhibition latérale ou de **bruit** (recuit simulé), reprenant la grammaire évoquée en chap. 7 :

$$\begin{aligned}\omega_{(n+1),\,j}(t+1) \\ = \;\omega_{(n+1),\,j}(t) \;+\; \eta\left[S\big(\mathcal{E}_{n+1},\,\mathcal{E}_j\big) \;-\; \tau\,\omega_{(n+1),\,j}(t) \;-\; \gamma \sum_{m \neq j} \omega_{(n+1),\,m}\,(t)\right] \\ +\; \xi_{(n+1),\,j}(t).\end{aligned}$$

On laisse ainsi une **dynamique** incrémentale s'exercer. Il est possible, à intervalles réguliers, de lancer un mini-cycle global pour réharmoniser la matrice ω dans son ensemble, si on craint une dérive ou un déséquilibre entre anciens et nouveaux nœuds.

C. Boucles de Contrôle : Continuité vs. Stabilisation

Un **SCN** en flux **ininterrompu** suppose que la matrice ω ne cesse d'être modifiée. On peut modéliser cela par un **processus** :

$$\omega(t+1) \;=\; F\big(\omega(t),\, \text{data}(t+1)\big),$$

où $\text{data}(t+1)$ indique l'ensemble des entités apparues à l'instant $t+1$. Dans la pratique, il se forme parfois des **périodes** de stabilisation relative, lorsque la cadence d'arrivée se fait faible et que le SCN se "repose" dans un régime quasi stationnaire. Mais ce repos peut être **bris** par l'arrivée d'un lot de nouvelles entités, provoquant une réorganisation locale ou globale. Par ailleurs, pour éviter l'oubli intégral de ce qui a été appris jadis, le **DSL** peut injecter de la vigilance ou un recuit périodique, incitant à revisiter les plus anciens liens et à les renforcer s'ils s'avèrent toujours pertinents (cf. chap. 7).

D. Avantages et Contraintes de l'Online

L'approche online offre d'abord la **réactivité** : le réseau s'adapte aussitôt que de nouvelles données surviennent, sans repartir de zéro. Elle assure également une **économie** de calcul par rapport à un recalcul complet, car on intègre les entités graduellement. En contrepartie, la **complexité** peut croître si le flux ne tarit pas et si aucune procédure de limitation de liens (k plus proches voisins, coupure des poids faibles) n'est mise en place. Enfin, le **SCN** ne converge pas nécessairement à un état unique, puisqu'il est sans cesse perturbé par des éléments neufs, appelant plutôt un régime de stabilisation parcellaire.

Conclusion

L'approche **online** (continu) fait de la matrice ω une **structure vivante**, en évolution incessante, capable de traiter en flux des entités successives. On évite ainsi le recalcul from scratch de $O(n^2)$ et on laisse la **logique** DSL se diffuser localement pour intégrer les nouveaux nœuds ou signaux, tout en préservant la mémoire acquise. Ce paradigme, décrit ici en [9.2.2.2], se relie directement à la notion plus générale d'**apprentissage continu** (chap. 9), lequel se déploie dans la durée et accompagne les scénarios dynamiques de la **robotique**, des **systèmes de recommandation** ou des **flux** multimédia.

9.2.2.3. Intérêt : s'adapter aux changements contextuels (nouveaux types d'entités, nouvelles corrélations)

Dans un **SCN** (Synergistic Connection Network) évoluant en **apprentissage continu**, il est indispensable de prendre en compte les **changements** de contexte, qu'il s'agisse de l'arrivée de **nouveaux types d'entités** ou de la formation de **nouvelles corrélations** entre entités déjà présentes. Contrairement à un apprentissage totalement figé (voir [9.2.2.1]), la possibilité de mettre à jour les pondérations $\{\omega_{i,j}\}$ lorsqu'apparaissent de nouvelles informations — qu'il s'agisse d'un nouveau module sensoriel, d'un inédit flux de données ou d'un changement de l'environnement — constitue un atout majeur pour la **flexibilité** et la **robustesse** du **DSL**. Les sections A, B et C ci-après discutent respectivement de l'accueil de **nouveaux types d'entités**, de la découverte de **nouvelles corrélations**, et de la manière dont le **SCN** conserve une **cohérence** adaptée au contexte. La section D [9.2.2.3] conclut sur l'importance pratique de cette adaptabilité.

A. Accueil de Nouveaux Types d'Entités

Les **nouvelles** entités $\mathcal{E}_{n+1}$ — qu'il s'agisse d'un capteur récemment ajouté ou d'un autre format d'information — peuvent survenir régulièrement dans des systèmes réels (robotique, veille de données, systèmes de recommandation). Sous une approche "batch" stricte, chaque insertion exigerait un "réapprentissage" complet, entraînant un coût démesuré. Dans un **SCN** qui accepte l'**apprentissage continu**, on se contente de créer de nouvelles lignes et colonnes $\omega_{(n+1),\,j}, \omega_{j,\,(n+1)}$ et d'initialiser ces pondérations à une valeur faible. Le calcul de la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ s'effectue alors, de préférence, sur un *voisinage* réduit (k plus proches voisins, par exemple) afin de ne pas engager $0(n^2)$ liaisons. Ce principe offre une **répartition** de la charge : plutôt que de relancer la dynamique intégrale du DSL, on limite la mise à jour au sous-ensemble de pondérations attachées à la nouvelle entité, tout en préservant les liens consolidés précédemment. Au niveau **mathématique**, l'ajout de la nouvelle entité $\mathcal{E}_{n+1}$ implique :

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

ce qui permet de **connecter** progressivement $\mathcal{E}_{n+1}$ au réseau sans déstabiliser la structure préexistante. Cette adaptation locale se révèle cruciale dans une mise en œuvre robotique où, par exemple, un **capteur thermique** est soudainement ajouté : le **SCN** se contente de calibrer un bloc de pondérations supplémentaires, plutôt que de reconstruire la matrice ω en totalité.

B. Découverte de Nouvelles Corrélations ou Synergies

Même dans un ensemble d'entités déjà présentes, les **corrélations** sous-jacentes peuvent varier avec le temps. De nouveaux **patterns** de **similarité** $S(\mathcal{E}_i, \mathcal{E}_j)$ peuvent émerger si l'environnement se transforme, si un événement externe modifie les liens statistiques (comme en finance, la corrélation entre actions pouvant monter ou descendre), ou si la distribution des données se déplace (changement de thématique dans un système de traitement du langage). Lorsque $S(i,j)$ change, un **SCN** orienté **online** ne se retrouve pas bloqué dans l'ancienne configuration. La règle DSL :

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

se met à jour en continu si l'on autorise $S(i,j,t)$ à dépendre explicitement de t. Les "**nouvelles corrélations**" montent en puissance au fur et à mesure que la pondération $\omega_{i,j}$ se renforce, tandis que les **anciens** liens perdent leur intensité lorsque leur synergie se dégrade. Sur le plan **mathématique**, tout se passe comme si le **SCN** se laissait porter par les fluctuations de S, ajustant graduellement ω sans interrompre la dynamique globale, préservant ainsi la **cohérence** et l'**historique** du réseau.

C. Maintien d'une Cohérence Adaptée au Contexte

Un **SCN** "vivant" oscille autour d'un **équilibre** qui n'est jamais complètement définitif. L'auto-organisation se poursuit au fil de l'arrivée de **nouvelles entités** (section A) ou de la découverte de **nouveaux liens** (section B). Cette **souplesse** confère au **DSL** (Deep Synergy Learning) une véritable capacité de **résilience** : si l'environnement connaît des modifications (changement de l'objectif robotique, mise à jour dans un système d'information, altération statistique dans les données), la **formule** DSL agit sur les pondérations $\omega_{i,j}$, alignant la structure sur ces nouveaux signaux. Le **SCN** en flux tire donc sa force de ce "équilibre en mouvement", plus apte à encaisser des perturbations qu'un modèle figé.

Conclusion

La faculté d'**adapter** un **SCN** aux changements contextuels (qu'il s'agisse de l'apparition de **nouveaux types** d'entités ou de la formation de **nouvelles corrélations**) représente un apport déterminant pour la **flexibilité** et la **robustesse** du **DSL** dans des conditions **réelles**. Lorsque le flux de données demeure ouvert, il est peu envisageable de relancer un apprentissage complet (coût prohibitif, perte d'historique). À l'inverse, l'ajustement progressif et incrémental des pondérations $\omega_{i,j}$ permet d'insérer de nouvelles entités dans un réseau déjà structuré sans désorganiser l'ensemble, d'autoriser la réorganisation des liaisons au gré des variations de $S(i,j,t)$ afin que les clusters et les liens prépondérants reflètent l'actualité des corrélations, et de conserver une cohérence malgré l'incertitude et l'évolutivité de l'environnement, grâce à l'auto-organisation itérative.

Ce parti pris est central au **Chapitre 9**, qui insiste sur l'**apprentissage continu** et la mise en œuvre d'un **SCN** capable de se **mouvoir** au même rythme que les données, en contraste avec l'approche offline (9.2.2.1) ou le recalcul partiel (9.2.2.2).

9.2.3. Équilibre entre Stabilisation et Plasticité

Dans un **SCN** (Synergistic Connection Network) fonctionnant en flux continu, le défi est de maintenir un **équilibre** entre la **stabilisation** des clusters formés et la **plasticité** nécessaire pour intégrer les changements (nouvelles entités, fluctuations contextuelles). Un réseau trop **rigide** ne s'adaptera pas aux évolutions ; un réseau trop **changeant** ne conservera aucune structure cohérente sur la durée. Ainsi, il faut ajuster la **vitesse** de mise à jour $\omega_{i,j}$ et la **persistance** des liens au fil des itérations, de manière à éviter l'oubli total des clusters tout en restant capable d'innover quand c'est utile.

9.2.3.1. Risque de sur-adaptation : si l'on modifie trop vite ω, on peut "oublier" les clusters déjà formés

Lorsque le **Synergistic Connection Network** (**SCN**) opère en **apprentissage continu**, la **mise à jour** de la matrice ω peut s'avérer trop brusque si le **taux d'apprentissage** η est excessivement grand ou si les nouvelles données modifient la **synergie** de manière trop radicale. Les deux sections A et B ci-dessous décrivent respectivement le **principe** du phénomène de sur-adaptation et l'**analyse formelle** du pas temporel, tandis que la section C [9.2.3.1] en tire les conclusions principales quant à l'oubli potentiel des clusters.

A. Principe de Sur-Adaptation et "Oubli" des Clusters

Dans un **SCN**, la **règle** DSL se formule généralement :

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

Si η (le **taux d'apprentissage**) est élevé, chaque nouveauté dans $S(i,j,t)$ affecte $\omega_{i,j}$ avec une ampleur importante, ce qui peut conduire à un **"oubli"** des liens antérieurement établis. En effet, une fois qu'un cluster $\{i,k,\dots\}$ est formé, on peut souhaiter le **préserver** tant que son intérêt ne disparaît pas totalement. Toutefois, si la règle de mise à jour autorise $\omega_{i,k}$ à chuter en quelques itérations face à un changement de synergie, le **cluster** se trouve rapidement démantelé, reproduisant un effet analogue au "**catastrophic forgetting**" connu dans d'autres branches de l'IA continue.

Sur le plan **mathématique**, un pas η trop élevé signifie :

$$\omega_{i,j}(t+1) \approx \omega_{i,j}(t) + \eta\, \Delta(t),$$

où $\Delta(t)$ peut être d'envergure dès que $S(i,j,t)$ se modifie un tant soit peu. L'expérience montre qu'il suffit alors de peu d'itérations pour "effacer" un cluster préalablement stabilisé, ce qui peut s'avérer nuisible si ces liens demeuraient pertinents en sous-contexte.

B. Analyse Formelle : le Pas Temporel

Il est souvent éclairant de décrire l'équation ci-dessus dans un **cadre** continu. Si l'on écrit :

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

on l'interprète comme la dérivée $d/dt\, \omega_{i,j}(t)$. La constante η joue donc le rôle d'un **pas** de temps. Si η est grand, $\omega_{i,j}(t)$ bouge rapidement, rendant la dynamique instable. À l'inverse, un η modeste impose un déplacement lent, intégrant l'historique de ω et évitant l'effondrement brutal de clusters.

Parfois, la mise à jour prend la forme d'un **filtre exponentiel** :

$$\omega_{i,j}(t+1) = (1-\alpha)\, \omega_{i,j}(t) + \alpha\, \omega_{\text{new}},$$

où ω_{new} correspond à la nouvelle valeur souhaitée. Si $\alpha \approx 1$, on bascule quasi instantanément vers ω_{new}. Si $\alpha \ll 1$, on opère une transition douce, sauvegardant une partie de la mémoire antérieure. Dans l'esprit du **DSL**, cet ajustement progressif évite la sur-adaptation.

C. Conclusion

Le **risque** de sur-adaptation se produit quand le **SCN** modifie ω trop vite, effaçant des clusters pourtant valables, un phénomène qui rappelle le "catastrophic forgetting". Pour maintenir un **équilibre** dans l'apprentissage continu (voir [9.2.3.2] pour la stabilisation), on veille à :

- **Limiter** la valeur de η pour modérer la réactivité,
- Introduire des mécanismes de **mémoire** longue (régularisation, filtre exponentiel, etc.),
- Surveiller la **cohérence** du réseau pour éviter que des clusters stables ne s'écroulent du jour au lendemain.

Ces ajustements assurent à la fois la plasticité nécessaire pour intégrer la nouveauté et la **stabilité** suffisante pour conserver l'essentiel des organisations antérieures. C'est dans cette tension que réside la robustesse d'un **SCN** en **apprentissage continu**, qui doit s'adapter sans pour autant sur-réagir à toute fluctuation passagère.

9.2.3.2. Nécessité d'un mécanisme contrôlant la vitesse d'apprentissage, parfois appelé "mémoire à long terme vs. court terme"

Lorsque le **Synergistic Connection Network** (**SCN**) opère en **apprentissage continu**, il devient capital de réguler la **vitesse** avec laquelle la matrice ω évolue. Un paramétrage inadapté du **taux d'apprentissage** η peut conduire à un "**oubli**" des clusters déjà formés (tel que décrit en [9.2.3.1]) ou, à l'inverse, à une inertie paralysante. Les sections A, B et C développent, respectivement, la notion de "vitesse d'apprentissage" et de "mémoire", l'impact sur la **dynamique** évolutive, et les outils mathématiques favorisant une gestion flexible de η. La section D [9.2.3.2] conclut en soulignant l'importance de ce mécanisme de contrôle.

A. Vitesse d'Apprentissage et "Mémoire" dans le SCN

Il est courant, dans la **formule** DSL, d'introduire :

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

Le paramètre η régule la **vitesse** à laquelle $\omega_{i,j}$ se rapproche d'une nouvelle "cible" dictée par $S(i,j,t)$. Une valeur **élevée** de η autorise une **réactivité** instantanée. Les changements de S affectent la structure ω très vite, ce qui peut être idéal dans un flux dynamique où l'on souhaite ne pas manquer les signaux récents. Toutefois, un tel réglage engendre la **fragilité**. De légères fluctuations ΔS suffisent à réécrire les liaisons, détruisant éventuellement les **clusters** qui s'étaient consolidés dans le passé.

À l'inverse, un η **faible** garantie la **stabilité**. Les pondérations $\omega_{i,j}$ intègrent lentement les nouveautés, préservant la **mémoire** des configurations antérieures. Mais ce parti pris peut s'avérer trop peu réactif si l'environnement (robotique, multimédia, logs) connaît des variations brusques. On voit donc émerger la tension entre un comportement "**court terme**" (absorber rapidement le nouveau) et "**long terme**" (conserver le noyau accumulé).

B. Impact sur la Dynamique Évolutive

Lorsque η est mal calibré, le **SCN** peut :

- **Osciller** ou **oublier** : si η est trop grand, chaque nouveau signal $S(i,j,t)$ bouleverse $\omega_{i,j}$, conduisant à un système hyper-réactif. Les liens se reforment et se détruisent en quelques itérations, empêchant la cristallisation de clusters solides.

- **Manquer d'adaptabilité** : si η est trop petit, le réseau stagne dans son état passé et tarde à refléter de nouveaux patterns. Sur le plan mathématique, $\omega_{i,j}(t+1) \approx \omega_{i,j}(t)$ pour de longues séquences, rendant le SCN incapable de "suivre" l'actualité des corrélations.
  Pour trouver un **équilibre**, on peut utiliser (chap. 7 ou [9.2.3.3]) des stratégies d'ajustement adaptatif. Le taux d'apprentissage change selon la phase de la dynamique, par exemple en s'accroissant en cas de perturbation massive et en diminuant lorsqu'un plateau de stabilité apparaît.

C. Outils Mathématiques pour la Gestion de la Vitesse

Il existe divers **mécanismes** pour orchestrer la répartition entre **mémoire à court terme** et **mémoire à long terme** :

- **Programmation par paliers**. On définit des moments $\{t_k\}$ où la valeur de η chute (ou augmente) d'un facteur α. Ainsi, on commence avec un η plus élevé pour saisir rapidement la structure grossière, puis on réduit η pour consolider les clusters.

- **Feed-back sur la variance**. Lorsque les changements $\Delta\omega_{i,j}$ demeurent importants d'un pas à l'autre, on abaisse η pour calmer la dynamique. Si, au contraire, $\Delta\omega$ reste trop minime, on peut l'augmenter pour injecter plus de plasticité.

- **Double contrôle**. On peut modéliser un $\omega_{i,j}^{(\text{fast})}$ et un $\omega_{i,j}^{(\text{slow})}$, chacun mis à jour avec un taux d'apprentissage différent. L'un gère la réactivité instantanée, l'autre la mémorisation de plus longue durée. Les deux se combinent in fine dans la pondération effective (chap. 9.2.3.3).

D. Conclusion

Il est essentiel pour un **SCN** en **apprentissage continu** de **contrôler** la vitesse à laquelle ω évolue. Ce contrôle renvoie à la mise en balance de la **mémoire à court terme**, qui répond vivement aux nouveautés, et de la **mémoire à long terme**, qui évite de sacrifier trop vite les clusters antérieurement construits. Les conséquences :

- Une réactivité modérée, pour ne pas tout réécrire au moindre changement,
- Une inertie non excessive, afin de ne pas ignorer les variations réelles de l'environnement,
- Un mécanisme adaptatif (paliers, feed-back sur la variance, double contrôle) permettant d'ajuster dynamiquement η au fil du temps.

Le **Chapitre 9** prolonge cette réflexion en introduisant des stratégies concrètes (chap. 9.2.3.3) destinées à gérer la plasticité/stabilité du réseau, de sorte que le DSL demeure un "réseau vivant" sans oublier ses acquisitions précédentes.

9.3. Mise à Jour Incrémentale et Localité

Dans de nombreux scénarios **temps réel**, il est peu réaliste de réévaluer la **matrice** $\{\omega_{i,j}\}$ au grand complet à chaque instant. Lorsque de **nouvelles informations** (entités, événements, mesures) arrivent, on souhaite plutôt **incrémenter** la structure du **SCN** (Synergistic Connection Network) sans tout recalculer. Cette logique d'**approche incrémentale** s'accompagne d'une notion de **localité**, où seules les liaisons $\omega_{i,j}$ réellement concernées par le flux de données sont mises à jour. (ex. un **voisinage** restreint), préservant ainsi une **complexité** raisonnable, tout en maintenant la **dynamique** d'auto-organisation du DSL.

9.3.1. Mise à Jour Incrémentale

Le principe général est de partir d'une **base** $\{\omega_{i,j}(t)\}$ déjà acquise à l'instant t, et de **réviser** localement ces pondérations en fonction des **nouvelles infos** qui parviennent au système. On cherche donc à ajuster $\omega_{i,j}(t+1)$ depuis $\omega_{i,j}(t)$ par l'ajout d'un **terme** $\Delta\omega_{i,j}$(nouvelles infos), sans repasser en revue toutes les entités.

9.3.1.1. Approche : $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta\omega_{i,j}$(nouvelles infos)

Dans une dynamique d'**apprentissage continu**, il est souvent nécessaire de mettre à jour le **Synergistic Connection Network** (**SCN**) de manière **incrémentale** plutôt que de recalculer intégralement toutes les liaisons. L'équation

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \Delta\omega_{i,j}(\text{nouvelles infos})$$

permet de souligner cette approche, on conserve $\omega_{i,j}(t)$, puis on y ajoute un **delta** qui rend compte des informations récemment apparues (une nouvelle entité $\mathcal{E}_{n+1}$, un flux sensoriel, un changement de contexte). Les sections A et B ci-dessous décrivent la **formulation** et l'**interprétation** du delta, illustrant cette approche avec un exemple d'insertion. La section C met en exergue les **avantages** et les **limites** liées à cet incrémental local. La conclusion (9.3.1.1) récapitule le principe et les prolongements attendus.

A. Formulation mathématique du "delta"

Dans un **flux** où apparaissent sans cesse de nouvelles entités $\{\mathcal{E}_{n+1}, \dots\}$, ou de nouveaux événements $\{e_1, e_2, \dots\}$, on définit :

$$\omega_{i,j}(t+1) \;=\; \omega_{i,j}(t) \;+\; \Delta\omega_{i,j}(\mathcal{E}_{\text{new}}, t),$$

lorsqu'on juge que la **mise à jour** de la pondération $\omega_{i,j}$ est requise. Si l'on applique la **logique** DSL habituelle, ce delta peut suivre un schéma :

$$\Delta\omega_{i,j} \;=\; \eta\big[S_{(t,\text{new})}(i,j) \;-\; \tau\,\omega_{i,j}(t)\big],$$

de manière à mimer la formule de renforcement/désactivation déjà présentée. L'idée est de **focaliser** la mise à jour sur un sous-ensemble d'indices (i,j). Si une liaison n'est pas impactée par la nouveauté, alors $\Delta\omega_{i,j} = 0$ et $\omega_{i,j}$ ne change pas. Sur le plan **mathématique**, on évite donc le recalcul global $O(n^2)$ pour se concentrer sur la zone concernée.

B. Interprétation

Cette approche incrémentale décrit une **réactualisation** progressive de la matrice ω. Au lieu d'un "batch" massif, on intègre petit à petit les **nouvelles** informations sans altérer l'essentiel de la structure. Dans le cadre du DSL, on reconnaît la forme :

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

mais **restreinte** à quelques paires (i,j). Cette logique peut se voir comme une forme de "gradient partiel", où seuls les poids proches de la nouvelle entité $\mathcal{E}_{\text{new}}$ sont modifiés.

En guise d'**exemple**, songeons à l'insertion d'une entité $\mathcal{E}_{n+1}$. On crée de nouveaux liens $\omega_{(n+1),\,j}$ initialisés à zéro. Puis, sur un certain "voisinage" $\mathrm{N}(\mathcal{E}_{n+1})$, on applique :

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

Le reste de la **matrice** $\{\omega_{i,j} \mid i,j \le n\}$ ne change pas, ou subit un ajustement mineur s'il existe un mécanisme global de "compétition" (inhibition latérale, chap. 7). Ainsi, la structure **globale** demeure, pendant qu'on intègre doucement $\mathcal{E}_{n+1}$.

C. Avantages et Considérations

L'équation

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta\omega_{i,j}(\text{nouvelles infos})$$

présente plusieurs **avantages**. Elle offre une **flexibilité** en permettant un traitement en quasi-temps réel, absorbant chaque flux d'information dès son apparition. Elle optimise l'**économie de calcul** en n'itérant pas sur toutes les paires (i,j), mais uniquement sur celles jugées **pertinentes** (voisinage, indices corrélés, etc.). Enfin, elle assure une **stabilité** en préservant la configuration antérieure tout en la **raffinant** localement, évitant ainsi de bouleverser la totalité de la **matrice** ω.

Néanmoins, il existe des **limites**. Si la nouvelle entité $\mathcal{E}_{n+1}$ se révèle centrale (fortement couplée à de nombreux nœuds), le champ d'intervention s'étend, ce qui peut imposer une mise à jour "en ondes". De plus, un contrôle fin du **taux d'apprentissage** η demeure requis pour éviter un "oubli" soudain de clusters solides (voir [9.2.3.1]) ou un temps de réaction trop lent en cas d'évolution drastique (voir [9.2.3.2]).

Conclusion

L'approche

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta\omega_{i,j}(\text{nouvelles infos})$$

résume le **principe** d'une mise à jour incrémentale dans le **SCN**. Au lieu d'un recalcul global (coûteux en $\mathrm{O}(n^2)$), on emploie un **delta** local pour insérer de nouvelles entités, adapter quelques poids, et préserver la structure existante. Cette logique **DSL** en flux est explorée plus avant dans [9.3.2] (stratégie de voisinage) et [9.3.3] (mise à jour progressive), autorisant un **réseau vivant** et constamment **ajusté** à la réalité changeante, sans détruire à chaque fois la configuration stable accumulée par le passé.

9.3.1.2. On ne recalcule pas toutes les synergies globales mais seulement dans le voisinage où le flux évolue (k plus proches voisins, etc.)

Lorsqu'un **Synergistic Connection Network** (**SCN**) doit gérer un **flux continu** de données ou de nouvelles entités, il n'est pas envisageable de **réévaluer** la **synergie** $S(i,j)$ pour *l'ensemble* des paires (i,j) à chaque instant. Une telle opération atteindrait une complexité $\mathrm{O}(n^2)$ lorsqu'il y a n entités, inadéquate pour des applications réactives. Cette section [9.3.1.2] décrit pourquoi et comment il est préférable de *localiser* le recalcul au **voisinage** de la zone affectée (p. ex. k plus proches voisins ou un ϵ-radius). Les paragraphes A, B et C discutent successivement du **rationnel**, des **conséquences** sur la mise à jour de $\omega_{i,j}$ et de la **cohérence** globale.

A. Rationnel et fondements mathématiques

Lorsque survient une **nouvelle entité** $\mathcal{E}_{n+1}$ ou qu'un nœud $\mathcal{E}_x$ se voit modifié (changement d'embedding, mise à jour des attributs), la solution naïve consisterait à recalculer $S(\mathcal{E}_x, \mathcal{E}_j)$ pour toutes les entités j. Cette stratégie conduit à un coût $\mathrm{O}(n)$ pour chaque entité impactée, qui, en s'additionnant sur plusieurs entités, peut dériver en $\mathrm{O}(n^2)$.
Au lieu de cela, l'idée est de restreindre le **recalcul** de la synergie à un *voisinage*, noté $\mathrm{N}(x) \subset \{1, \dots, n\}$, où $\mathcal{E}_x$ trouve ses **k plus proches voisins** (k-NN) ou ceux situés dans un rayon ϵ. Plusieurs techniques (structures d'indexation, KD-tree, ball-tree, etc.) accélèrent la détermination de ce voisinage. D'un point de vue **mathématique**, on n'évalue plus $S(\mathcal{E}_x, \mathcal{E}_j)$ pour *tout* j, mais seulement pour un $* \, sous-ensemble \, *$. Si la taille de ce sous-ensemble demeure bornée (k constant ou ϵ-radius), le coût se réduit à $\mathrm{O}(k)$ ou $\mathrm{O}(\log n)$ par insertion ou mise à jour locale.

B. Conséquences sur la mise à jour $\omega_{i,j}$

Dans l'**apprentissage continu**, on déploie généralement la **formule** de mise à jour DSL sur ce voisinage $\mathrm{N}(x)$. Prenons l'exemple d'une **nouvelle entité** $\mathcal{E}_{n+1}$. On crée les liaisons $\{\omega_{(n+1),j}\}$ en les initialisant à zéro (ou à un petit ε), puis on calcule :

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

Seuls les pondérations $\omega_{(n+1),j}$ où $j \in \mathrm{N}(n+1)$ évoluent, reflétant la **logique** DSL. Les autres $\omega_{(n+1),k}$ restent nuls (ou ne sont pas créés). Si un mécanisme de "compétition" (inhibition) ou de "voisin du voisin" intervient, on étend la mise à jour sur un second cercle, mais on s'abstient d'un **recalcul** global sur $\mathrm{O}(n^2)$. Il en résulte que le **graphe** $\{\omega_{i,j}\}$ demeure **sparse**, car chaque entité ne se lie réellement qu'à un sous-ensemble local.

C. Vérification de la cohérence globale

En limitant le recalcul à un voisinage, on renonce partiellement à la possibilité de découvrir des **corrélations lointaines** imprévues. Il peut alors être nécessaire de prévoir un "scan périodique" élargi pour détecter si, à la longue, $\mathcal{E}_{n+1}$ se révèle aussi proche de certains nœuds $\mathcal{E}_m$ hors de $\mathrm{N}(n+1)$. Mais dans la plupart des situations pratiques, la distribution des entités permet d'estimer que si deux nœuds sont distants, il est fort improbable qu'ils entrent subitement en forte synergie. Le **compromis** entre localisme et exhaustivité se résout souvent en faveur d'une stratégie k-NN ou ϵ-radius, conférant au SCN un calcul incrémental compatible avec un apprentissage continu en temps réel.

Si le SCN comprend un **macro-nœud** ou un niveau supérieur d'organisation (voir chap. 6), il peut injecter des signaux descendants pour réévaluer certains liens en dehors du voisinage direct

lorsqu'un phénomène global est détecté. On reste néanmoins dans un schéma **incrémental** et localement géré, plutôt qu'un recalcul batch.

Conclusion

La solution consistant à **recalculer** $S(i,j)$ uniquement sur le *voisinage* (k plus proches voisins, ϵ-radius) est un puissant levier d'**économie de calcul** et de **scalabilité** en flux continu :

- Le coût cesse d'être $O(n^2)$. Au moment d'insérer $\mathcal{E}_{n+1}$, on ne s'occupe que de $N(n+1)$.
- La **mise à jour** DSL maintient la structure existante et intègre la **nouvelle** entité ou les attributs modifiés dans un cercle local.
- Le **SCN** final est **sparse**, assurant une interprétation plus aisée et une auto-organisation plus rapide.

Cette approche se marie idéalement avec le cadre d'**apprentissage continu** développé au **Chapitre 9**, où l'on cherche à **absorber** les flux de données ou d'entités au fil de l'eau, sans refonder l'ensemble de la matrice ω.

9.3.1.3. Gains en temps de calcul

Un **Synergistic Connection Network** (**SCN**) appliqué à un **grand nombre** d'entités $\{\mathcal{E}_i\}_{i=1}^{n}$ peut faire face à un **coût** de calcul prohibitif si l'on maintient la matrice $\{\omega_{i,j}\}$ en $O(n^2)$. En contexte d'**apprentissage continu**, il est crucial de trouver des **dispositifs** qui réduisent sensiblement le temps de calcul associé à la mise à jour itérative des pondérations. Les trois sections A, B, et C explorent, respectivement, l'usage de **calculs distribués**, d'**approximations sélectives** (k-NN, ϵ-radius, etc.) et de **paramétrages** dynamiques. La section D [9.3.1.3] synthétise l'ensemble, soulignant l'impact sur les temps de calcul.

A. Distribution du Calcul et Multithreading

Dans un **SCN** volumineux, l'idée centrale consiste à **répartir** le travail de mise à jour en **blocs** ou **sous-SCN**, chacun traitant un sous-ensemble $\{\mathcal{E}_i\}$. Une manière élémentaire consiste à partager l'ensemble $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ en m blocs approximativement égaux, à appliquer la **formule** DSL de manière localement indépendante dans chacun de ces blocs (sous-SCN), puis à synchroniser occasionnellement les liens transversaux $\omega_{i,j}$ reliant des entités de blocs différents, selon une périodicité fixe ou adaptative.

Cette approche rompt la complexité $O(n^2)$ en blocs plus restreints, ce qui peut faire baisser les temps de calcul totaux grâce au **parallélisme**. Les **communications** inter-blocs sont moins fréquentes et peuvent s'effectuer en une phase de "merge" après un certain nombre d'itérations locales. Ce principe, analogue à une **architecture distribuée** (voir chap. 5.7.2), s'implémente aisément via des routines **multithreads** ou sur un cluster de machines, où chaque nœud gère un bloc. L'**avantage** est l'exploitation du **multicœur** ou du multiprocesseur, tout en évitant de manipuler simultanément $O(n^2)$ pondérations à chaque itération.

B. Approximations Sélectives : k-NN, ϵ-Radius, etc.

Une seconde famille de **dispositifs** vise à **restreindre** la liste des pondérations à mettre à jour, plutôt que de maintenir un graphe complet. On se limite, pour chaque entité $\mathcal{E}_i$, à ses "k plus

proches voisins" (k-NN) ou à ceux situés dans un rayon ϵ. Sur le plan **mathématique**, la matrice $\{\omega_{i,j}\}$ devient **clairsemée**, seule une poignée de poids $\omega_{i,j}$ demeurent actifs.

Une première approche repose sur le **k-NN**, où l'on définit un voisinage $\mathrm{N}_k(i)$ qui contient les k entités parmi $\{\mathcal{E}_j\}_{j\neq i}$ les plus proches de $\mathcal{E}_i$ selon une distance ou un critère de similarité. Les autres poids $\omega_{i,j}$ sont fixés à 0 (ou ϵ), et on ne les manipule pas. Cela **réduit** la complexité par entité à $\mathrm{O}(k)$, descendant ainsi de $\mathrm{O}(n)$ à $\mathrm{O}(k)$.

Une autre stratégie repose sur le ϵ-**Radius**, où l'on conserve uniquement les entités $\mathcal{E}_j$ situées dans un rayon ϵ autour de $\mathcal{E}_i$. On ne met à jour $\omega_{i,j}$ que si $\mathbf{x}_j$ est spatialement proche de $\mathbf{x}_i$. En pratique, on obtient, pour un jeu de données réparti, une **densité** de connections ω très moindre, limitant ainsi le travail de calcul.

En appliquant la **formule** DSL sur cette version sparse, on maintient l'**essentiel** des interactions (les entités fortement reliées) tout en s'épargnant l'effort de contrôler ou de recalculer les liaisons $\omega_{i,j}$ faiblement contributives. Empiriquement, cette troncature ne dégrade pas beaucoup la **qualité** des clusters, tout en abaissant sensiblement les coûts.

C. Ajustement Dynamique des Paramètres

Des **mécanismes** peuvent venir parachever la **réduction** du temps de calcul en ajustant en permanence certains paramètres.

Une première approche consiste à rendre η **et** τ **variables**. Il est usuel de démarrer avec un η plus élevé pour converger rapidement, puis de diminuer η afin de stabiliser les clusters. S'il existe un changement de contexte, on peut remonter η. Du point de vue **calcul**, on se permet ainsi de limiter la fréquence des passes de mise à jour intensive.

Une autre approche repose sur une **inhibition adaptable**. La force de l'inhibition latérale (voir chap. 7) peut être relevée pour filtrer plus agressivement les liens "moyens", forçant la **sparsité** de la matrice ω. Dans ce cas, on allège davantage la charge de calcul, puisqu'on se retrouve avec moins de pondérations non nulles à manipuler.

Ces principes visent à **accélérer** la phase d'apprentissage ou à éviter les mises à jour superfétatoires lorsque le SCN se rapproche d'un état stable ou lorsqu'un cluster est déjà bien formé.

D. Synthèse : Gains Mesurables en Temps (9.3.1.3)

En combinant la **distribution** du calcul via des approches multithread ou des sous-SCN, des **approximations** sélectives comme le k-NN ou le ϵ-radius pour limiter la densité de ω, ainsi que des **ajustements** dynamiques des paramètres η, τ, γ et autres, on observe une chute drastique du **coût** total. Plutôt que de traiter $\mathrm{O}(n^2)$ liaisons à chaque itération, on se limite à $\mathrm{O}(k \cdot n)$ ou $\mathrm{O}(n^2/m^2)$ selon l'option prise. Les expériences rapportées dans des scénarios classiques (réseaux de capteurs, flux multimédia, clusterisation de documents) révèlent des **facteurs** de gain pouvant aller de 5 à 50 ou plus, sans entamer de manière sensible la qualité de l'**auto-organisation**. Un SCN ainsi paramétré demeure **flexible**, capable d'apprendre en continu, tout en offrant la **scalabilité** requise pour aborder de larges volumes de données.

9.3.2. Stratégie de Voisinage Restreint

Dans un **SCN** (Synergistic Connection Network) de grande taille — disons n entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ — le coût computationnel de la mise à jour complète $\omega_{i,j}(t+1)$ pour toutes les paires (i,j) peut se révéler prohibitif ($O(n^2)$ par itération). Pour **alléger** cet encombrement, on recourt à une **stratégie de voisinage restreint**, consistant à ne **mettre à jour** que les liens $\omega_{i,j}$ jugés les plus *pertinents* ou les plus *proches* selon un certain critère (distance, similarité, type d'entité, etc.). Ainsi, on échappe au recalcul de la synergie pour toutes les paires, ce qui d'un point de vue algorithmique optimise sensiblement les performances.

9.3.2.1. Considérer que seuls les liens $\omega_{i,j}$ où $\mathcal{E}_j$ est proche (en distance ou type) sont mis à jour

Dans un **SCN** (Synergistic Connection Network) de grande taille, il est souvent recommandé de limiter la mise à jour des pondérations $\omega_{i,j}$ à un **voisinage** restreint pour chaque entité $\mathcal{E}_i$. Cela consiste à ne tenir compte que des liaisons $\omega_{i,j}$ où $\mathcal{E}_j$ se trouve dans un périmètre de proximité (au sens d'une distance ou d'une similarité). On obtient ainsi un **sous-ensemble** $\mathcal{V}(i) \subseteq \{1, \dots, n\}$, dont la taille est nettement inférieure à n. Les entités situées hors de ce sous-ensemble ne sont tout simplement pas incluses dans la mise à jour de $\omega_{i,j}$. L'approche, tout en améliorant l'efficacité, demeure cohérente avec la formule DSL.

Du point de vue **mathématique**, cette politique de voisinage se traduit par :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big] & \text{si } j \in \mathcal{V}(i), \\ \omega_{i,j}(t) & \text{sinon.} \end{cases}$$

Lorsque la distance ou la similarité $\mathrm{d}(\mathcal{E}_i, \mathcal{E}_j)$ est utilisée pour définir $\mathcal{V}(i)$, on ne retient par exemple que les k plus proches entités (k-NN) ou celles situées dans un rayon ϵ. On réduit ainsi drastiquement la complexité du calcul. Si $|\mathcal{V}(i)| = K$, chaque entité n'opère plus que $\mathrm{O}(K)$ mises à jour au lieu de $\mathrm{O}(n)$. Le coût global de la mise à jour par itération passe alors à $\mathrm{O}(nK)$, ce qui peut représenter un gain considérable si $K \ll n$.

Il faut toutefois noter que l'on risque de "manquer" la création de liaisons entre deux entités apparemment lointaines, mais potentiellement synergiques. Il devient alors utile de réévaluer, à intervalle régulier, le voisinage $\mathcal{V}(i)$ afin de laisser la place à de nouvelles connexions imprévues. Il peut aussi être envisageable de conserver un faible pourcentage de "liens exploratoires" aléatoires, dans l'éventualité où des entités lointaines, sur le papier, finissent par présenter un fort intérêt (la dynamique DSL pourrait ainsi ranimer ces liens).

Malgré ces restrictions, la qualité de la clusterisation reste bonne dans la plupart des cas. Les liaisons essentielles (entités effectivement proches) sont traitées de manière récurrente, et les liaisons négligeables ne pèsent plus sur la simulation. L'effet principal est de **clairsemer** la matrice $\{\omega_{i,j}\}$, tout en maintenant l'**esprit** du DSL, où le réseau s'auto-organise localement, en misant sur les liens qui comptent vraiment. Cette démarche s'avère d'autant plus cruciale à grande échelle ou en **temps réel**, où l'on ne peut supporter un coût $\mathrm{O}(n^2)$ en continu. L'usage d'un voisinage restreint constitue donc un compromis solide entre **efficacité** (faible coût de mise à jour) et **fidélité** (le SCN reste capable de capturer l'essentiel de la structure émergente).

9.3.2.2. Éviter un $O(n^2)$ complet à chaque itération

Lorsqu'un **Synergistic Connection Network** (**SCN**) se confronte à un nombre potentiellement large d'entités $\{\mathcal{E}_i\}_{i=1}^{n}$, un recalcul exhaustif des synergies $S(i,j)$ à chaque itération mènerait à un coût d'ordre $O(n^2)$. Cette échelle de complexité devient vite ingérable si l'on vise des mises à jour fréquentes ou un apprentissage continu dans un flux de données. Pour pallier ce problème, diverses stratégies d'allégement sont mises en œuvre, qui permettent de maintenir l'esprit du **DSL** (mise à jour adaptative) tout en contrôlant le temps de calcul.

A. Rappels du schéma ω et de son coût

L'équation canonique de mise à jour DSL s'écrit généralement

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

Appliquée naïvement à chaque paire (i,j), cette mise à jour a un coût $O(n^2)$ par itération puisque l'on considère toutes les combinaisons d'entités. Pour un grand n, cette exigence se révèle rapidement prohibitive. Dans la pratique, maintenir un SCN entièrement dense tout en réalisant un apprentissage continu n'est pas soutenable à grande échelle.

B. Grandes Approches pour Réduire le Calcul

Une première approche consiste à n'actualiser que les **voisins** les plus pertinents, limitant ainsi la densité du graphe. Il est alors possible de définir un ensemble $N(i) \subset \{1,\dots,n\}$ autour de chaque entité $\mathcal{E}_i$, soit par la méthode des k plus proches voisins (k-NN) selon une distance prédéfinie, soit par un rayon ϵ. Grâce à ce choix local, on ne met à jour $\omega_{i,j}$ que pour $j \in N(i)$, ce qui ramène le coût d'une itération à $O(n \cdot k)$ si la taille moyenne de $N(i)$ est de l'ordre k. Un autre levier s'appuie sur une **partition** préalable, on regroupe les entités en blocs ou macro-clusters, puis on effectue localement les mises à jour dans chaque bloc, ne recalculant que moins souvent les synergies entre blocs. Cette distribution du réseau permet de profiter d'une parallélisation naturelle si l'on dispose d'un matériel multiprocesseur.

C. Équilibre Qualité–Efficacité

En restreignant le calcul de la synergie, on risque d'omettre quelques liaisons lointaines potentiellement utiles. Toutefois, il s'avère que la plupart des liens à longue distance sont peu pertinents pour la formation de clusters stables. Un usage raisonné de stratégies comme k-NN ou ϵ-radius ne dégrade qu'à la marge la qualité globale de la structure émergente. Certains mécanismes prévoient une réévaluation périodique, afin de permettre l'apparition de liaisons "improbables" si elles deviennent pertinentes. L'**inhibition** ou les procédés de **recuit** restent compatibles avec cette approche parcimonieuse, puisqu'ils s'exercent sur la sous-partie du graphe dont les liens sont effectivement actifs.

D. Contexte d'Implémentation

Lorsque le **DSL** est appliqué à un réseau volumineux, toute solution qui s'approche de l'itération complète $O(n^2)$ finit par bloquer en pratique. Les solutions reposant sur la limitation du voisinage (k-NN, ϵ-radius), sur l'écrêtage automatique des pondérations trop faibles ou sur l'architecture distribuée deviennent alors incontournables. Ces aménagements permettent un apprentissage continu, tout en maintenant la dynamique adaptative chère au DSL. Le coût se voit considérablement réduit, souvent de plusieurs ordres de grandeur, tandis que la *diminution* d'exactitude (par rapport à la version entièrement dense) demeure acceptable dans la plupart

des scénarios. Il en résulte un **SCN** "vivant" et **scalable**, à même de suivre l'évolution d'un flux de données sans être encombré par le fameux $O(n^2)$.

9.3.2.3. Exemple : on ne recalcule la synergie que pour 50 plus proches entités

Dans de nombreux **scénarios** d'**apprentissage continu** ou de **mise à jour** incrémentale (voir [9.3.1.2] sur la réduction du $O(n^2)$), il est crucial de contenir la complexité lorsqu'une **nouvelle entité** $\mathcal{E}_{n+1}$ apparaît. L'évaluation de la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ pour tous les $j \in \{1, \dots, n\}$ induit un coût proportionnel à n. Si l'on répète l'opération à chaque arrivée dans un **flux**, le total peut devenir ingérable pour de grands n. L'idée, détaillée ci-après, consiste à se limiter aux k plus proches entités, par exemple $k = 50$. La section A présente le **contexte** et la démarche, la section B élabore la **formule** mathématique et la mise à jour, la section C propose une **discussion** sur le compromis efficacité–précision, et la section D [9.3.2.3] conclut en soulignant l'intérêt pratique de cette méthode.

A. Contexte et Démarche

L'approche s'inscrit dans l'esprit d'un **SCN** fonctionnant en flux (voir [9.3.1.1] pour la logique du delta incrémental). À l'instant t, une nouvelle entité $\mathcal{E}_{n+1}$ se joint au réseau. Dans un schéma naïf, on évaluerait la **distance** ou la **similarité** de $\mathcal{E}_{n+1}$ avec chaque entité $\mathcal{E}_j$ déjà présente, puis on calculerait la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$. Cela conduit à un coût $O(n)$ par insertion. Sur un ensemble grand, la répétition de cette opération à chaque mise à jour rendrait l'**apprentissage continu** irréalisable. Restreindre le calcul à un **voisinage** de taille fixe k est donc une solution pratique. Si on fixe $k = 50$, la recherche d'entités pertinentes se ramène à l'identification des 50 plus proches éléments, dans un sens de distance ou de similarité.

B. Formule Mathématique de la Mise à Jour Incrémentale

Supposons que chaque entité $\mathcal{E}_i$ soit décrite par un vecteur $\mathbf{x}_i \in \mathbb{R}^d$. On cherche à définir un ensemble $\mathrm{N}_{50}(n+1) \subset \{1, \dots, n\}$ de taille 50, constitué des entités jugées les plus proches de $\mathbf{x}_{n+1}$. On peut écrire :

$$\mathrm{N}_{50}(n+1) = \mathrm{TopK}\big(d(\mathbf{x}_{n+1}, \mathbf{x}_j),\ j = 1, \dots, n,\ K = 50\big),$$

où d est une distance (euclidienne, cosinus, etc.). Seuls les indices $j \in \mathrm{N}_{50}(n+1)$ seront utilisés pour calculer la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$.

Une fois ce voisinage déterminé, la **mise à jour** dans le **SCN** suit la formule du **DSL** (cf. chap. 7), appliquée uniquement à ces paires $\big((n+1), j\big)$. Si on part d'une initialisation nulle $\omega_{(n+1),\,j}(t) = 0$, on établit :

$$\omega_{(n+1),\,j}(t+1) = \omega_{(n+1),\,j}(t) + \eta\big[S(\mathcal{E}_{n+1}, \mathcal{E}_j) - \tau\, \omega_{(n+1),\,j}(t)\big], \quad j \in \mathrm{N}_{50}(n+1).$$

Les autres pondérations $\omega_{(n+1),\,k}(t+1)$ pour $k \notin \mathrm{N}_{50}(n+1)$ restent à zéro. On réduit ainsi le calcul de **synergie** de $O(n)$ à $O(50)$. La détermination de $\mathrm{N}_{50}(n+1)$ peut s'effectuer via des structures spatiales (k-d tree, vantage-point tree) ou des algorithmes d'approximate nearest neighbors, maintenant un coût $O(\log n)$ (ou $O(n^{\alpha})$) pour la recherche.

C. Discussion : Compromis Efficacité–Précision

La limitation à 50 plus proches entités est un **choix** fixant un compromis. Sur le plan **efficacité**, on obtient une insertion ou une mise à jour en $O(\log n + k) \approx O(k)$ pour la partie la plus lourde du calcul, ce qui est extrêmement avantageux si n dépasse plusieurs milliers ou millions. D'un autre côté, on néglige la possibilité qu'une entité $\mathcal{E}_{n+1}$ soit faiblement proche en vecteur mais présente une forte synergie conceptuelle avec une entité $\mathcal{E}_m$. Si l'on juge un tel cas improbable, ou si on laisse la dynamique DSL ultérieure réhabiliter ce lien via d'autres chemins (macro-nœuds, recuit, inhibition, etc.), on considère cette approximation comme suffisamment fiable. Dans la pratique, la démarche s'avère robuste, car la plupart des liens utiles relèvent de la proximité spatiale (ou d'une métrique sémantique), tandis que les entités trop lointaines en représentation n'apportent guère de synergie. On préserve ainsi l'**essentiel** des connexions sans saturer le calcul d'opérations peu contributives. Il est parfois envisagé de renouveler périodiquement la recherche de voisinage, au cas où $\mathcal{E}_{n+1}$ se déplacerait dans l'espace d'embedding ou participerait à de nouvelles interactions révélant d'autres affinités.

D. Conclusion

Ne calculer la **synergie** que pour 50 (ou un petit k) **plus proches** entités constitue une **exemple** emblématique de stratégie d'approximation locale dans un **SCN** en flux. Cette méthode permet :

- De ramener la mise à jour pour chaque nouvelle entité à un coût $O(k)$ au lieu de $O(n)$.
- De maintenir la **qualité** d'auto-organisation de la dynamique DSL, étant donné que la plupart des synergies intéressantes s'inscrivent dans un environnement proche.
- De rendre un **apprentissage continu** viable, là où un recalcul exhaustif deviendrait irréalisable dès que n croît.

La combinaison de ce principe avec des mécanismes comme l'**inhibition** et des structures d'indexation spatiale consolide encore la possibilité d'opérer un **SCN** à grande échelle, en respectant l'esprit du DSL — liaisons renforcées localement et adaptatives dans le temps, sans submerger le calcul par des $O(n^2)$ répétitifs.

9.3.3. Mécanisme de Diminution Progressive

Dans une **logique** d'apprentissage continu (chap. 9.3) et de **gestion** du SCN (Synergistic Connection Network) au fil des évolutions, il est souvent nécessaire de **décrémenter** ou de **faire oublier** les liaisons $\omega_{i,j}$ qui ne sont plus sollicitées. Ce phénomène de "**forgetting**" assure que le réseau demeure **dynamique**, les entités (ou clusters) anciennement liées, mais inactives, se voient progressivement **détachées** au lieu de persister indéfiniment. Ainsi, on évite l'**encombrement** par des connexions obsolètes.

9.3.3.1. Si une entité n'a pas interagi depuis longtemps, $\omega_{i,j}$ diminue progressivement (concept de "forgetting")

Dans un **Synergistic Connection Network** (**SCN**) où la **dynamique** se prolonge dans le temps (voir [9.3.1.1] pour la logique d'insertion en flux et [9.3.2.3] pour la restriction k-NN), il est utile de disposer d'un **mécanisme** de réduction progressive des liaisons $\omega_{i,j}$ restées inactives.

La **mémoire** du réseau doit certes conserver les **clusters** établis, mais ne pas s'encombrer de liens obsolètes. Cette section [9.3.3.1] montre pourquoi et comment on introduit un terme de "**forgetting**", qui érode $\omega_{i,j}$ lorsqu'aucun **renforcement** n'a été observé depuis un certain temps. Les paragraphes A et B expliquent le principe et les justifications mathématiques, tandis que C souligne l'importance du **taux** λ. La conclusion (9.3.3.1) récapitule l'intérêt de cette décroissance contrôlée.

A. Principe de Diminution Progressive

Il arrive qu'une entité $\mathcal{E}_i$ et une autre $\mathcal{E}_j$ n'entretiennent plus de synergie nouvelle pendant un laps de temps prolongé. On veut alors **réduire** la pondération $\omega_{i,j}$, évitant que le réseau ne sature de liens délaissés depuis trop longtemps. Sur le plan **mathématique**, on peut formaliser la règle par :

$$\omega_{i,j}(t+1) \leftarrow \begin{cases} \omega_{i,j}(t), & \text{si un renforcement } (\Delta\omega_{i,j} > 0) \text{ s'est produit,} \\ (1-\lambda)\,\omega_{i,j}(t), & \text{sinon,} \end{cases}$$

où $\lambda > 0$ est un paramètre contrôlant la vitesse de "**forgetting**". Concrètement, si aucun **renforcement** n'est survenu pour $\omega_{i,j}$ depuis un certain délai δ, la liaison décroît par un facteur $(1-\lambda)$. On peut ainsi implémenter un compteur $\text{count}_{i,j}$, où chaque itération sans mise à jour incrémente $\text{count}_{i,j}$; une fois $\text{count}_{i,j} \geq \delta$, on applique le "**forget**".

B. Justification Mathématique

Sur le plan **algébrique**, la présence d'un terme d'érosion $-\lambda\,\omega_{i,j}(t)$ conditionnel revient à introduire un "effet d'oubli" dans la dynamique du **DSL**. Sans ce mécanisme, la matrice ω peut s'alourdir de liens moyennement élevés qui ne reçoivent plus d'**activation** ni de **synergie** nouvelles. À mesure que de nouvelles entités arrivent (voir [9.3.1.2]) ou que les clusters se réorganisent, certains liens se retrouvent non sollicités, mais resteraient figés à des valeurs élevées, **perturbant** la lisibilité et la structure. En imposant :

$$\omega_{i,j}(t+1) = (1-\lambda)\,\omega_{i,j}(t) \quad \text{(lorsque inactif)},$$

on se donne la possibilité de ramener progressivement $\omega_{i,j}$ vers zéro. Les liens réellement "vivants", nourris par des synergies $\Delta\omega_{i,j} > 0$, ne subissent pas cet effacement puisqu'ils sont réactivés régulièrement. Cette **équation** de décroissance contribue à maintenir un **réseau vivant**, seuls subsistent les liens efficaces ou rafraîchis, tandis que les connexions passées sans usage tombent en désuétude, évitant la surcharge.

C. Choix du Taux λ

Le **paramètre** $\lambda \in (0,1)$ détermine la rapidité du "**forgetting**". Si λ est faible ($\lambda \approx 0.01$), les liens inactifs restent significatifs pendant plus longtemps. Si λ est grand ($\lambda \approx 0.1$ ou 0.2), la pondération chute vite à chaque étape de "non-renforcement". Le choix dépend du **niveau** de plasticité désiré. Un **faible** λ favorise la mémoire longue, potentiellement utile si l'environnement se stabilise à nouveau. Un λ élevé accentue la réactivité, mais peut inciter le **SCN** à "oublier" trop brusquement des clusters encore pertinents. Au-delà de ce paramètre, la synchronisation peut aussi tenir compte d'autres indicateurs, par exemple la variance de ω, la densité de liens, ou l'interaction avec l'inhibition (voir chap. 7.4) pour ajuster le "**forgetting**" dans le temps.

Conclusion

Lorsque l'on conçoit un **SCN** en flux, il est vital de prévoir un **mécanisme** de "**forgetting**" afin de gérer les liens inactifs ou obsolètes. La **diminution progressive** $\omega_{i,j}(t+1) \leftarrow (1-\lambda)\,\omega_{i,j}(t)$ (appliquée au bout d'un délai δ d'inactivité) évite la saturation d'une matrice ω massive. Sur le plan **mathématique**, on introduit ainsi un terme conditionnel d'érosion qui agit dès qu'aucun renforcement ne s'est produit. Les bénéfices sont multiples. L'allégement du réseau permet un meilleur repérage des **clusters** actifs et accélère les mises à jour, puisque des liens tombent de la dynamique. En revanche, le choix de λ et du délai δ exige un **calibrage** attentif, pour maintenir un juste équilibre entre mémoire historique et plasticité aux évolutions du **flux**.

9.3.3.2. Contrôle : $\omega_{i,j}(t+1) \leftarrow (1-\lambda)\,\omega_{i,j}(t)$ si pas de renforcement récent

Lorsqu'un **Synergistic Connection Network** (**SCN**) se déploie en **apprentissage continu**, il peut être nécessaire d'**effacer** ou de **diminuer** au fil du temps les liaisons qui ne reçoivent plus de signaux de renforcement (c'est-à-dire aucune synergie ou aucune co-occurrence nouvelle). Cette stratégie dite de "**forgetting**" prévient l'accumulation de liens moyennement forts devenus obsolètes. Plus précisément, la **règle** de contrôle :

$$\omega_{i,j}(t+1) \leftarrow (1-\lambda)\,\omega_{i,j}(t), \quad \text{si aucune synergie ou renforcement récent pour } (i,j),$$

avec $\lambda \in [0,1]$, fournit une **décroissance** contrôlée. La condition "pas de renforcement récent" signifie que la mise à jour **DSL** standard n'a pas produit de hausse ($\Delta\omega_{i,j} > 0$) depuis un certain nombre d'itérations ou qu'aucun événement $S(i,j)$ n'a été relevé sur ce lien depuis un délai δ. Les sections A, B, C et D ci-après explicitent respectivement la **justification mathématique**, l'**interprétation**, des **exemples** et la **conclusion** (9.3.3.2), en se référant aux principes développés précédemment (voir [9.3.3.1] sur le "forgetting" progressif).

A. Justification Mathématique

Dans la **dynamique** du DSL, on associe souvent à chaque lien $\omega_{i,j}(t)$ une **mise à jour** de la forme :

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

Cette équation (1) renforce $\omega_{i,j}(t)$ si la **synergie** $S(i,j)$ compense $\tau\,\omega_{i,j}(t)$. Toutefois, si aucune **interaction** ne vient nourrir $\omega_{i,j}$ sur plusieurs itérations, on souhaite un "**reset**" partiel. Le moyen le plus direct est la **décroissance** :

$$\omega_{i,j}(t+1) = (1-\lambda)\,\omega_{i,j}(t), \quad \text{si pas de renforcement depuis un certain temps.}$$

On peut rassembler (1) et (2) dans un **schéma** en deux étapes. On calcule d'abord la mise à jour DSL $\omega_{i,j}^{(\mathrm{DSL})}(t+1)$, puis on applique :

$$\omega_{i,j}(t+1) = \begin{cases} \omega_{i,j}^{(\mathrm{DSL})}(t+1), & \text{si } \Delta\omega_{i,j} > 0, \text{ (renforcement)}, \\ (1-\lambda)\,\omega_{i,j}^{(\mathrm{DSL})}(t+1), & \text{sinon.} \end{cases}$$

Mathématiquement, cela revient à imposer un "**frottement**" ou une "**pénalisation**" pour les liens inactifs. L'application de (2) ou (3) se fait typiquement si $\omega_{i,j}(t+1)$ n'a pas reçu de

hausse ($\Delta\omega_{i,j} > 0$) depuis δ itérations (ou un "timer" d'inactivité). Voir [9.3.3.1] pour la logique plus générale de l'oubli progressif.

B. Interprétation et Implications

Cette **diminution** $\omega_{i,j}(t+1) \leftarrow (1-\lambda)\,\omega_{i,j}(t)$ traduit la volonté de ne conserver dans la matrice $\{\omega_{i,j}\}$ que les liens réellement "**vivants**", c'est-à-dire réactivés régulièrement par le flux (voir [9.3.1.2] pour la logique d'insertion incrémentale). Les liaisons qui ne sont plus stimulées se "délitent" de manière exponentielle. Le paramètre $\lambda \in [0,1]$ est alors décisif :

- Si $\lambda \approx 0.01$, la liaison $\omega_{i,j}$ ne baisse que de 1 % par itération inerte. Il lui faudrait ~100 itérations de non-usage pour être divisée par e ≈ 2.718. On préserve ainsi une mémoire longue, ce qui peut être souhaitable si les clusters passés ont une chance de re-devenir utiles.

- Si $\lambda \approx 0.2$, la **régression** de la liaison se fait plus rapide, 20 % de chute par itération en l'absence de stimulus. C'est idéal pour un environnement très **changeant** où la "longue traîne" de liens anciens deviendrait encombrante.

Impact sur la **convergence** : ce "**forgetting**" conditionnel renforce la **sélectivité** du SCN. Les clusters évoluent plus nettement, les liaisons inactives disparaissent, ce qui favorise un graphe clairsemé et des **clusters** moins brouillons. Toutefois, si λ est trop grand, on court le risque d'**éradiquer** trop vite des liens encore potentiellement valables (mais non activés sur quelques itérations). Le meilleur choix demeure contextuel.

C. Application et Exemples

Exemple 1 : $\omega_{i,j} \approx 0.8$ **à l'instant** t**,** $\lambda = 0.05$

Imaginons que deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$ entretenaient un **lien** fort $\omega_{i,j} = 0.8$. Depuis quelque temps, plus aucun renforcement ne le nourrit ($\Delta\omega_{i,j} > 0$ n'a pas eu lieu). Au bout de δ itérations sans mise à jour positive, on applique $\omega_{i,j}(t+1) = (1-0.05)\,\omega_{i,j}(t) = 0.95\,\omega_{i,j}(t)$. Si cette décroissance est réitérée 10 fois de suite, la liaison descend de 0.8 à $0.8 \times (0.95)^{10} \approx 0.8 \times 0.598 \approx 0.478$. Au-delà de 20 ou 30 itérations inactives, elle peut quasiment s'effacer (valeurs proches de 0). Cela déleste le **SCN** d'un poids devenu inutile. Si un nouvel événement fait rejaillir la synergie, la formule DSL (1) s'en chargera, rehaussant $\omega_{i,j}$.

Exemple 2 : Réseau conversationnel

Dans un **SCN** lexical (voir [9.3.2.3] pour la restriction k-NN), si deux tokens $\mathcal{E}_i, \mathcal{E}_j$ ne co-apparaissent plus dans le **flux** de textes, $\Delta\omega_{i,j} \leq 0$ sur plusieurs itérations. On décide alors de décrémenter $\omega_{i,j}$ par $(1-\lambda)$ chaque fois qu'on constate "aucun renforcement". Ainsi, l'association lexicalement désuète s'estompe au profit de liaisons plus actives.

Exemple 3 : Système de recommandation

Un lien $\omega_{i,j}$ reliant deux items ou deux utilisateurs, qui ne reçoivent plus d'achats/notations communs, se voit érodé par $(1-\lambda)\,\omega_{i,j}(t)$. Le paramètre λ se calibre pour conserver un "souvenir" raisonnable (faible λ), ou pour insister sur la fraîcheur (fort λ). Au besoin, un "timer" δ peut compléter le mécanisme. Si $\text{count}_{i,j} \geq \delta$ itérations inactives, on applique la baisse.

D. Conclusion

Le **contrôle** :

$$\omega_{i,j}(t+1) \leftarrow (1-\lambda)\,\omega_{i,j}(t) \quad \text{(si pas de renforcement récent)}$$

matérialise un "**forgiveness**" ou un "**frottement**" qui pousse vers zéro les liaisons $\omega_{i,j}$ non alimentées par la règle DSL (1). On obtient un **SCN** plus économe, où la mémoire des liens se conserve seulement si des signaux la justifient. Cette dualité "**renforcement**" s'il y a synergie, "**décroissance**" s'il n'y en a plus, renforce la **plasticité** tout en évitant l'embourbement dans des liaisons figées. La mention d'un paramètre $\lambda \in [0,1]$ est cruciale. Un λ léger (≈ 0.01) assure un oubli lent, un λ fort (≈ 0.2) autorise une grande réactivité. Les **exemples** (capteurs, lexical, recommandation) illustrent cette mécanique au cœur de l'**apprentissage continu**, permettant à la matrice ω de ne **pas** stocker éternellement des poids inactifs et de reconfigurer librement la structure du **SCN** face aux changements du **flux**.

9.3.3.3. Prévention de l'encombrement mémoire (clusters obsolètes)

Lorsqu'un **Synergistic Connection Network** (**SCN**) est sollicité dans un **flux continu** (cf. [9.3.3.1] pour le principe d'oubli progressif et [9.3.3.2] pour la règle de décroissance si pas de renforcement récent), certains *clusters* perdent leur pertinence au fil du temps. Sans mécanisme pour gérer ces "**clusters obsolètes**", le système court le risque de **surcharger** sa mémoire, ralentir ses calculs et brouiller la dynamique globale. Il devient ainsi crucial de disposer de méthodes pour (i) **identifier** ces clusters désuets et (ii) **alléger** la structure en supprimant ou fusionnant ces groupes vieillissants. Les sections A, B, C et D ci-après détaillent ce défi. La section A pose le contexte, B développe les critères mathématiques, C énonce les stratégies de nettoyage, et D conclut (9.3.3.3) sur les bénéfices pour un SCN en flux.

A. Contexte : Clusters en Apprentissage Continu

Dans un **SCN** incrémental (voir chap. 9.2), l'insertion régulière de nouvelles entités $\{\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \ldots\}$ tend à réorienter la **synergie** globale. Des *clusters* se forment pour rassembler les entités actives du moment, mais certains clusters plus anciens peuvent ne plus recevoir d'entités entrantes ni de signaux de renforcement (voir [9.3.3.1] et [9.3.3.2] sur la logique d'oubli). Par exemple, dans un **système** de recommandation, un ensemble de produits qui n'est plus acheté finit par constituer un "**cluster fantôme**". S'il n'est pas désactivé, ce cluster obsolète continue d'exister dans la **matrice** $\{\omega_{i,j}\}$, ajoutant du bruit et consommant de la **mémoire**. Dans un contexte robotique ou conversationnel, le même phénomène se produit lorsqu'un sous-thème ne se voit plus alimenté. La synergie interne stagne, mais si l'on ne l'écarte pas, la structure globale du SCN s'encombre de liens inactifs.

B. Critères Mathématiques de Vieillissement ou d'Obsolescence

Plusieurs **critères** permettent de décider qu'un cluster $\mathcal{C}_\alpha$ est devenu obsolète.

Faible synergie moyenne. On peut évaluer la moyenne ou la somme des liaisons dans $\mathcal{C}_\alpha$. Par exemple,

$$\Omega(\mathcal{C}_\alpha) = \sum_{i,j \in \mathcal{C}_\alpha} \omega_{i,j}, \quad \overline{\omega}_\alpha = \frac{1}{|\mathcal{C}_\alpha|^2} \sum_{i,j \in \mathcal{C}_\alpha} \omega_{i,j}.$$

Si $\Omega(\mathcal{C}_\alpha)$ ou $\overline{\omega}_\alpha$ reste sous un seuil ϵ pendant un certain **laps** ou diminue sans se renouveler, c'est le signe que le cluster a perdu sa vigueur. On peut alors enclencher une **dissolution** (voir section C).

Faible flux entrant. Dans un **DSL** en flux, on note la synergie *entrante* depuis les nouvelles entités vers $\mathcal{C}_\alpha$. Formulé mathématiquement, si $\Delta_{\text{in}}(\mathcal{C}_\alpha, t) \approx 0$ sur plusieurs périodes, plus aucune entité récente ne vient se connecter ou renforcer $\mathcal{C}_\alpha$. Cela traduit le fait que ce groupe n'attire plus aucune co-occurrence, donc qu'il est "orphelin" dans la dynamique.

Inactivité temporelle totale. On peut surveiller la variation interne du cluster. Si

$$\max_{i,j \in \mathcal{C}_\alpha} |\omega_{i,j}(t+1) - \omega_{i,j}(t)| \; < \; \delta_0,$$

pour un certain seuil $\delta_0 > 0$ et sur une durée prolongée, on en déduit que le cluster ne **bouge** plus. Couplé à un module d'**oubli** (voir [9.3.3.2]), ce critère entérine l'absence de renforcement et justifie la suppression ou fusion.

C. Stratégies pour Libérer la Mémoire et Gérer les Clusters Obsolètes

Lorsqu'un cluster $\mathcal{C}_\alpha$ est jugé obsolète, on dispose de divers **mécanismes** :

Dissolution ou suppression. On met purement et simplement $\omega_{i,j} \leftarrow 0$ pour tous $i, j \in \mathcal{C}_\alpha$, puis on "libère" ces entités du graphe (elles peuvent passer en "standby" ou être redirigées vers un état inactif). Dans certains cas, on enregistre encore la possibilité qu'une entité $\mathcal{E}_i$ se réactive (ou un flux la ressuscite), mais le cluster en tant que tel n'existe plus.

Fusion ou réallocation. Si quelques nœuds $\mathcal{E}_i \in \mathcal{C}_\alpha$ manifestent encore un potentiel de synergie avec un autre cluster $\mathcal{C}_\beta$, on procède à un **transfert** progressif. On identifie les $\omega_{i,k}$ (avec $k \in \mathcal{C}_\beta$) qui restent modérément élevés, et on *migre* $\mathcal{E}_i$ dans $\mathcal{C}_\beta$. Ainsi, on évite de détruire tout le cluster $\mathcal{C}_\alpha$ si certaines entités conservent un usage. On le fait en synergie avec la règle de "décroissance" (voir [9.3.3.2]) pour vider peu à peu $\mathcal{C}_\alpha$.

Nettoyage progressif. D'un point de vue algorithmique, on peut effectuer un "scan" périodique. On détecte les clusters dont la synergie interne n'évolue plus et on applique un paramètre λ' plus élevé pour "effacer" graduellement toutes leurs $\omega_{i,j}$. Cela peut se modéliser comme la formule (voir [9.3.3.2]) :

$$\omega_{i,j}(t+1) \leftarrow (1-\lambda')\,\omega_{i,j}(t) \quad (i,j \in \mathcal{C}_\alpha,\ \lambda' > \lambda).$$

Ainsi, on accélère la sortie de scène d'un cluster complet.

D. Bénéfices et Exemples Numériques

En se débarrassant de clusters obsolètes, on **diminue** la densité de la matrice $\{\omega_{i,j}\}$. Le coût par itération (souvent $O(n^2)$ en naive, voir [9.3.2.2]) se réduit parce que l'on ne maintient plus la mise à jour des poids internes au cluster mort.
Dès lors qu'un cluster n'apporte plus de synergy, l'inclure dans la dynamique DSL ne fait que troubler la formation de nouveaux clusters. Le fait de le "supprimer" ou de le "fusionner" clarifie la topologie, permettant au SCN de se concentrer sur les groupes réellement soutenus par le flux.

Supposons qu'un cluster $\mathcal{C}_\alpha$ regroupe 100 entités, et que la moyenne interne $\overline{\omega}_\alpha \approx 0.8$ à l'instant t. Après plusieurs itérations de non-usage, disons $\Delta t = 50$, la **règle** $(1-\lambda)\,\omega_{i,j}(t)$

(voir [9.3.3.2]) fait décroître la plupart des liaisons, si $\lambda = 0.05$. En 50 itérations, $(1 - 0.05)^{50} \approx (0.95)^{50} \approx 0.076$. Ainsi, la synergie interne retombe près de 0.06-0.07. On constate alors $\Omega(\mathcal{C}_\alpha) \approx 0$, ou $\Delta_{\text{in}}(\mathcal{C}_\alpha) \approx 0$. On déclare $\mathcal{C}_\alpha$ "**obsolète**" et on l'exclut de la suite de la dynamique ou on le fusionne s'il y a quelques entités qui restent modérément reliées ailleurs.

Conclusion

Le **SCN** en **apprentissage continu** doit gérer la **fin de vie** des clusters devenus inutiles pour éviter un encombrement mémoire qui alourdirait la dynamique. À travers des **critères** (faible synergie moyenne, faible flux entrant, inactivité temporelle) et des **stratégies** (suppression, fusion, nettoyage progressif), on assure que les entités obsolètes ne continuent pas d'occuper un espace démesuré dans la matrice $\{\omega\}$. Des **exemples** numériques (ex. un cluster de dimension 100 perdant sa synergie en 50 itérations à $\lambda = 0.05$) illustrent l'amortissement rapide des poids et l'exclusion d'un cluster ancien. Cette approche concourt à la **stabilité** globale et à la **cohérence** d'un SCN qui évolue au gré du flux, ne laissant pas traîner des structures mortes dans la mémoire.

9.4. Insertion et Suppression d'Entités en Flux

Dans de nombreux scénarios (systèmes de recommandation, robotique multi-sensorielle, surveillance continue, etc.), le **SCN** (Synergistic Connection Network) ne se limite pas à un ensemble fixe d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$. Au contraire, de **nouvelles entités** peuvent apparaître (et d'autres disparaître) au fil du **temps** ou du **flux** de données. L'aptitude du **DSL** (Deep Synergy Learning) à gérer ces **insertions** et **suppressions** de manière incrémentale est cruciale pour demeurer un **réseau** vivant, capable d'évoluer et de se reconfigurer sans "réapprentissage" global.

Cette section (9.4) est dédiée à la **gestion** de ces changements structurels : **insertion** (9.4.1) et **retrait** (9.4.2) d'entités, complétés par un aperçu de **scénarios** multi-arrivées (9.4.3). Nous commençons par détailler l'ajout d'une nouvelle entité dans un **SCN** déjà formé, en nous focalisant particulièrement sur la façon d'initialiser et de mettre à jour les **pondérations** ω associées.

9.4.1. Insertion d'Une Nouvelle Entité

Le sous-réseau DSL doit intégrer la **nouvelle entité** $\mathcal{E}_{\text{new}}$ en lui attribuant (et en recevant) des **liens** $\omega_{\text{new},j}$ (ou $\omega_{j,\text{new}}$) vis-à-vis des entités existantes $j = 1, \dots, n$. Comme l'on souhaite éviter un **coût** de recalcul global $O(n^2)$, il est fréquent de recourir à une initialisation locale ou approximative (ex. k plus proches voisins).

9.4.1.1. addEntity($\mathcal{E}_{\text{new}}$) : initialiser $\omega_{\text{new},j}$ via un calcul local (k plus proches voisins, sim)

Lorsqu'un **Synergistic Connection Network** (**SCN**) opère en **apprentissage continu**, l'arrivée d'une **nouvelle** entité $\mathcal{E}_{\text{new}}$ implique de l'**intégrer** sans réentraînement global. Pour y parvenir, on appelle une fonction addEntity($\mathcal{E}_{\text{new}}$) qui initialise les liens $\omega_{\text{new},j}$. Il faut éviter de démarrer avec $\omega_{\text{new},j} = 0$ pour *tous* les j, car la nouvelle entité ne développerait aucun lien si la mise à jour DSL n'était pas globale. La stratégie consiste donc à **deviner** un ensemble de liaisons initiales "raisonnables", généralement en recourant à un **calcul local** : on identifie, pour $\mathcal{E}_{\text{new}}$, un **voisinage** de taille k (k plus proches voisins selon la distance ou la similarité), et on initialise les pondérations $\omega_{\text{new},j}$ pour ces j.

Les sous-sections A et B décrivent successivement la **formulation** de la méthode (calcul local en k-NN) et les **conséquences** sur la dynamique, avant la conclusion (9.4.1.1). Les **formules** ci-après détaillent l'initialisation et l'implantation de ce principe.

A. Principe d'Insertion et Formulation Mathématique

Lors de l'invocation addEntity($\mathcal{E}_{\text{new}}$), on crée les **liaisons** $\{\omega_{\text{new},j}\}_{j=1}^{n}$. Si on ne fait rien (tous à 0), $\mathcal{E}_{\text{new}}$ n'aura pas d'attaches directes. On préfère donc la **méthode** k plus proches voisins :

- **Calcul** d'une **mesure** $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ ou d'une **distance** $d(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ pour $j = 1, \dots, n$. Cela induit un coût $O(n)$ si l'on doit examiner toutes les entités.
- **Tri** ou **sélection** des k entités les plus "pertinentes" ou "similaires", notées $N_k(\text{new})$. Avec un algorithme "**select**" on peut se contenter de $O(n)$, tandis qu'un tri complet

coûte $O(n\log n)$. On peut aussi recourir à une structure **ANN** (Approximate Nearest Neighbors) pour réduire le temps de recherche lorsque n est grand.

- **Initialisation** des liens : pour $j \in N_k(\text{new})$, on fixe

$$\omega_{\text{new},j}^{(0)} = \phi\left(S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)\right),$$

où ϕ est une **fonction** déterminant la pondération initiale (ex. $\phi(x) = \alpha\, x$ ou $\phi(x) = \min(\alpha x, \omega_{\max})$). En dehors de $N_k(\text{new})$, on met $\omega_{\text{new},j} = 0$.

- **Suite** de la dynamique : dès l'itération suivante, on applique la mise à jour DSL standard :

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

ainsi que d'autres mécanismes (inhibition, recuit) pouvant affiner la valeur initiale.

Ce **voisinage** de taille k assure que $\mathcal{E}_{\text{new}}$ ait quelques liens de départ, évitant une "**désert**" dans la matrice ω. L'exploration DSL ultérieure peut alors conforter ou diminuer ces premiers liens.

B. Conséquences sur la Dynamique et la Topologie

L'**initialisation** des liaisons $\omega_{\text{new},j}$ a un **impact** direct sur la dynamique du réseau.

La **réaction en chaîne** se produit dès que la nouvelle entité est reliée à k autres. La mise à jour DSL interagit alors avec ces liens, entraînant une possible reconfiguration des clusters. Si $\mathcal{E}_{\text{new}}$ est très similaire à un cluster existant, elle s'y associe naturellement et peut influencer la répartition des pondérations au sein de ce cluster, provoquant fusion ou renforcement.

Le **gain d'efficacité** est notable. On évite un recalcul global "**from scratch**" en privilégiant un **calcul local** de complexité $O(n) + O(n)$ ou $O(n\log n)$ pour sélectionner les k plus proches voisins. À l'itération suivante, seules les liaisons $\omega_{\text{new},j}$ (pour $j \in N_k$) et éventuellement $\omega_{j,\text{new}}$ dans le cas d'un réseau symétrique sont intégrées dans la boucle DSL.

Des **risques** existent si $\mathcal{E}_{\text{new}}$ est fortement similaire à plusieurs clusters. On peut alors voir émerger un "super-nœud" connectant plusieurs ensembles, ce qui peut entraîner une fusion excessive. Toutefois, cela reflète la **philosophie** d'un SCN, qui s'auto-organise librement. Pour limiter une sur-connexion immédiate, un mécanisme d'**inhibition** peut être intégré.

Conclusion

La fonction addEntity($\mathcal{E}_{\text{new}}$) s'**implémente** idéalement par un **calcul local**, consistant à identifier les k entités les plus proches selon un critère de similarité ou de distance, puis **initialiser** $\omega_{\text{new},j}$ et $\omega_{j,\text{new}}$ (en cas de SCN non orienté, on force la symétrie). Les formules :

$$\omega_{\text{new},j}^{(0)} = \phi\left(S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)\right), \quad j \in N_k(\text{new}),$$

et $\omega_{\text{new},j}^{(0)} = 0$ pour $j \notin N_k$, fournissent une base. La dynamique DSL affine ensuite les liens. Ainsi, la **nouvelle** entité s'insère dans le **SCN** de manière **incrémentale**, évitant le coût $O(n^2)$ d'une réévaluation globale, tout en maintenant une **cohérence**, la vectorisation ou la mesure de similarité assure que $\mathcal{E}_{\text{new}}$ démarre dans un cluster proche des entités qui lui ressemblent.

9.4.1.2. Ajuster $\omega_{j,\text{new}}$ symétriquement ou non (si on est en version orientée)

Lors de l'**insertion** d'une entité $\mathcal{E}_{\text{new}}$ dans un **SCN** (Synergistic Connection Network), on doit spécifier les liaisons $\omega_{j,\text{new}}$ (et éventuellement $\omega_{\text{new},j}$) reliant la nouvelle entité aux nœuds déjà présents. Le **choix** entre **symétrie** ($\omega_{j,\text{new}} = \omega_{\text{new},j}$) et **distinctivité** ($\omega_{j,\text{new}} \neq \omega_{\text{new},j}$) dépend de la nature même du **réseau**, selon qu'il soit non orienté ou orienté. Les paragraphes A et B ci-après distinguent ces deux cas, avant que la section C discute des **conséquences** sur la dynamique, et que la conclusion (9.4.1.2) récapitule les points clés.

A. Réseau Non Orienté : Ajustement Symétrique

Si le **SCN** se veut **non orienté**, la pondération d'un lien entre $\mathcal{E}_j$ et $\mathcal{E}_{\text{new}}$ doit s'interpréter de la même façon dans les deux sens. On impose donc :

$$\omega_{j,\text{new}}(t) \;=\; \omega_{\text{new},j}(t), \quad \forall\, j.$$

Cette **condition** permet de ne manipuler qu'une **unique** variable $\omega_{(j,\text{new})}$, évitant le double stockage. Lorsqu'on insère la nouvelle entité $\mathcal{E}_{\text{new}}$, on initialise la liaison de manière **symétrique**. Par exemple, si (voir [9.4.1.1] sur le calcul local) on a sélectionné un **voisinage** $\text{N}_k(\text{new})$ et calculé la similarité $S\big(\mathcal{E}_j, \mathcal{E}_{\text{new}}\big)$, on fixe :

$$\omega_{j,\text{new}}(0) \;=\; \omega_{\text{new},j}(0) \;=\; \phi\left(S\big(\mathcal{E}_j, \mathcal{E}_{\text{new}}\big)\right), \quad j \in \text{N}_k(\text{new}),$$

et on laisse $\omega_{j,\text{new}} = 0$ en dehors de ce voisinage (voir [9.4.1.1] pour la logique de k-NN). Par la suite, la **mise à jour** DSL (chap. 4) agit en rehaussant ou en diminuant la pondération, mais on conserve toujours :

$$\omega_{j,\text{new}}(t+1) = \omega_{\text{new},j}(t+1).$$

D'un point de vue **mathématique**, la mise à jour DSL est la même pour $\omega_{j,\text{new}}$ et $\omega_{\text{new},j}$. On renforce la liaison si la **synergie** $S(j,\ \text{new})$ l'exige, on l'atténue sinon, et on assure $\omega_{j,\text{new}}(t) = \omega_{\text{new},j}(t)$ à chaque étape. Cela souligne une **relation réciproque** entre $\mathcal{E}_j$ et $\mathcal{E}_{\text{new}}$.

B. Réseau Orienté : Ajustement Distinct $\omega_{j,new} \neq \omega_{new,j}$

Si, au contraire, le **SCN** est **orienté**, la liaison "$\mathcal{E}_j \to \mathcal{E}_{\text{new}}$" (pondération $\omega_{j,\text{new}}$) ne coïncide pas forcément avec "$\mathcal{E}_{\text{new}} \to \mathcal{E}_j$" ($\omega_{\text{new},j}$). On se retrouve alors avec **deux** variables indépendantes :

$$\omega_{j,\text{new}}(t), \quad \omega_{\text{new},j}(t).$$

C'est pertinent lorsqu'on modélise une influence **dirigée**, un flux causal ou toute **relation** intrinsèquement asymétrique. Sur le plan **mathématique**, dès qu'on insère $\mathcal{E}_{\text{new}}$, on peut :

1. **Calculer** une synergie "sortante" $S^{+}\big(\mathcal{E}_j, \mathcal{E}_{\text{new}}\big)$ pour $\omega_{j,\text{new}}$.
2. **Calculer** une synergie "entrante" $S^{-}\big(\mathcal{E}_{\text{new}}, \mathcal{E}_j\big)$ pour $\omega_{\text{new},j}$.

Ensuite, on **initialise** :

$$\omega_{j,\text{new}}(0) = \phi^{+}\left(S^{+}\big(\mathcal{E}_j, \mathcal{E}_{\text{new}}\big)\right), \quad \omega_{\text{new},j}(0) = \phi^{-}\left(S^{-}\big(\mathcal{E}_{\text{new}}, \mathcal{E}_j\big)\right).$$

La mise à jour DSL se scinde alors en deux formules :

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

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

Ce **double** paramétrage offre une **flexibilité** supplémentaire, au prix d'une plus grande **complexité** car la matrice n'est plus symétrique, et la somme des liens "$\mathcal{E}_j \to \mathcal{E}_{\text{new}}$" peut différer radicalement de "$\mathcal{E}_{\text{new}} \to \mathcal{E}_j$".

C. Conséquences sur la Dynamique du SCN

Simplicité. Dans un SCN **non orienté**, on réduit le nombre de paramètres à gérer de moitié : $\omega_{j,\text{new}} = \omega_{\text{new},j}$. Les **clusters** formés traduisent un regroupement réciproque (p. ex. similarité). L'**insertion** de la nouvelle entité se fait plus aisément, car on n'a qu'à définir un seul vecteur de liens.

Expressivité. Dans un SCN **orienté**, on gagne la faculté de modéliser des relations asymétriques (ex. "$\mathcal{E}_j$ influence $\mathcal{E}_{\text{new}}$" plus que l'inverse"). Ce mode orienté reflète une causalité ou un flux directionnel. Toutefois, il **double** la taille du stockage $\omega_{j,\text{new}}$ et $\omega_{\text{new},j}$ et complique la **mise à jour**, nécessitant soit le calcul de deux synergies (sortante et entrante), soit un unique $\tilde{S}$ scindé en deux composantes. L'équation DSL agit ensuite de manière distincte.

Coopération avec Mécanismes d'Inhibition. Quel que soit le mode (symétrique ou dirigé), l'arrivée d'une nouvelle entité $\mathcal{E}_{\text{new}}$ peut provoquer un "rassemblement" de clusters autour d'elle. Dans le cas orienté, on peut voir apparaître un "hub" sortant ou entrant. La **compétition** (chap. 7) ou l'**inhibition** latérale demeurent utiles pour éviter un surcroît de connexions.

Conclusion

Lorsque l'on **ajoute** une entité $\mathcal{E}_{\text{new}}$ au **SCN**, on doit décider si la relation $\omega_{j,\text{new}} = \omega_{\text{new},j}$ s'applique (cas non orienté) ou si on conserve deux variables distinctes (cas orienté). Le **choix** dépend de la nature de la **synergie**, selon qu'elle soit essentiellement réciproque (similarité, co-occurrence) ou plutôt directionnelle (influence, flux). Dans un réseau non orienté, la symétrie des poids est généralement préservée, on impose :

$$\omega_{j,\text{new}}(t) = \omega_{\text{new},j}(t),$$

et on n'a qu'un vecteur de liens à initialiser et mettre à jour. En réseau orienté, on autorise :

$$\omega_{j,\text{new}}(t) \neq \omega_{\text{new},j}(t),$$

avec deux **fonctions** de synergie $S^{+}(\cdot)$ et $S^{-}(\cdot)$. On gagne alors un potentiel descriptif plus riche, au prix d'une plus grande complexité de mise à jour. Dans tous les cas, la **dynamique** DSL (avec ou sans inhibition, recuit) évoluera en conséquence, ajustant $\omega_{j,\text{new}}$ et $\omega_{\text{new},j}$ (s'ils sont distincts) ou $\omega_{(j,\text{new})}$ (s'ils sont symétriques) selon la **synergie** mesurée et les principes d'**apprentissage continu**.

9.4.1.3. Risques : si $\mathcal{E}_{\text{new}}$ est très proche de plusieurs clusters, merges ou reconfigurations possibles

Il est fréquent, en **apprentissage continu**, qu'une **nouvelle entité** $\mathcal{E}_{\text{new}}$ s'insère dans un **SCN** (Synergistic Connection Network) et affiche une forte **synergie** avec plusieurs clusters déjà existants. Cette situation peut déclencher des **merges** imprévus ou provoquer de profondes

reconfigurations dans la structure globale, car la dynamique **DSL** (chap. 4) tente alors d'incorporer $\mathcal{E}_{\text{new}}$ de manière cohérente. Les principes d'initialisation de la nouvelle entité (voir [9.4.1.1]) et le choix d'une structure symétrique ou orientée (voir [9.4.1.2]) préparent ce moment, mais ne suffisent pas toujours à prévenir l'**unification** d'ensembles jusqu'ici séparés.

A. Proximité Multi-Cluster et Fondements Mathématiques

Supposons qu'on insère $\mathcal{E}_{\text{new}}$ à l'itération t et que, selon la règle DSL standard, chaque lien $\omega_{\text{new},i}$ évolue suivant la formule

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

Lorsque la **similarité** ou la **distance** entre $\mathcal{E}_{\text{new}}$ et un groupe $\mathcal{C}_1$ est très élevée, on observe une croissance notable de $\{\omega_{\text{new},j}\}_{j\in\mathcal{C}_1}$. Si, simultanément, un autre cluster $\mathcal{C}_2$ se voit également renforcé par $\mathcal{E}_{\text{new}}$, on obtient deux pôles d'ancrage, la nouvelle entité "**multiconnecte**" plusieurs ensembles. Si $\mathcal{C}_1$ et $\mathcal{C}_2$ sont jusqu'à présent distincts, la synergie inter-groupes peut augmenter via $\mathcal{E}_{\text{new}}$. Cela déclenche une **baisse** de la distance apparente entre $\mathcal{C}_1$ et $\mathcal{C}_2$, risquant d'entraîner leur **fusion**.

B. Dynamique de Reconfiguration et Merges Inopinés

Une entité $\mathcal{E}_{\text{new}}$ jouant le rôle de "pont" peut pousser le **SCN** à réaliser un **merge** entre deux clusters. Dans la pratique, on voit se constituer un **super-cluster** $\mathcal{C}_{\text{merge}} = \mathcal{C}_1 \cup \{\mathcal{E}_{\text{new}}\} \cup \mathcal{C}_2$. Sur le plan **mathématique**, la croissance des liaisons $\omega_{\text{new},j}$ (pour $j \in \mathcal{C}_1 \cup \mathcal{C}_2$) perturbe l'équilibre antérieur. Comme décrit dans la mise à jour standard,

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

les entités $\mathcal{E}_k$ connectées à $\mathcal{C}_1$ ou $\mathcal{C}_2$ peuvent elles-mêmes adapter $\omega_{k,j}$ en conséquence, déclenchant une **cascade** de modifications. On aboutit parfois à un **grand cluster** dont la cohérence interne s'avère discutable ; un re-splitting ultérieur peut alors se produire (chap. 10).

L'**instabilité** passagère découle du fait que la nouvelle entité "amène" un ensemble de liaisons élevées dans plusieurs directions. S'il n'existe pas de mécanismes de contrôle (inhibition latérale, feedback descendant, recuit, etc.), le réseau peut se retrouver inopinément fusionné, conduisant à un ensemble énorme et peu discriminant.

C. Risques de Convergence Sous-Optimale

Une telle **fusion** peut s'avérer souhaitable lorsque $\mathcal{C}_1$ et $\mathcal{C}_2$ auraient dû être unis depuis longtemps (regroupant des entités trop proches). Mais elle peut aussi être **trompeuse** si $\mathcal{E}_{\text{new}}$ se situe à mi-chemin entre deux ensembles fondamentalement distincts. La dynamique DSL pourrait en résulter :

$$\sum_{(i\in\mathcal{C}_1, j\in\mathcal{C}_2)} \omega_{i,j}(t+1)$$

qui grimpe du fait des liens "**bridge**" créés par $\mathcal{E}_{\text{new}}$. Si ce regain de liaison ne reflète pas une véritable compatibilité globale, le SCN risque alors de rassembler abusivement des entités hétérogènes dans un même "super-cluster", conduisant à une **convergence** sous-optimale ou à un re-splitting. Une période de flottement peut survenir, où la structure se re-répartit (chap. 9.3.3.x pour la gestion de clusters obsolètes ou fusions), imposant un coût de réadaptation plus élevé.

D. Conclusion

Lorsque $\mathcal{E}_{\text{new}}$ affiche une **forte synergie** envers plusieurs clusters, la dynamique **DSL** (chap. 4 et 7) peut effectuer des merges ou de grandes reconfigurations. Ce phénomène souligne la puissance **auto-organisatrice** du SCN, où un nœud "multi-proche" amène à repenser la partition existante et à unifier des ensembles. Néanmoins, la **fusion excessive** ou hâtive peut conduire à des effets parasites (grands clusters peu homogènes, oscillations). Les **mécanismes** de rétrocontrôle (inhibition, feedback descendant) demeurent cruciaux pour canaliser ces merges et garantir que la nouvelle entité s'intègre de façon équilibrée. Un **scénario** d'apprentissage continu exige donc une attention particulière. Si un unique nœud se révèle "pont" entre plusieurs sous-groupes, l'évolution de la matrice ω peut perturber la convergence globale.

9.4.2. Retrait d'Entité

Dans un **SCN** (Synergistic Connection Network) évoluant en flux (voir section 9.4.1 sur l'**ajout** d'entités), il peut arriver qu'une entité $\mathcal{E}_k$ devienne **obsolète** ou non pertinente (données périmées, capteur défaillant, concept jugé inutile). Le **retrait** de cette entité doit alors s'opérer de manière **cohérente** et **minimiser** les perturbations sur le reste du réseau. L'algorithme removeEntity$(\mathcal{E}_k)$ s'occupe de supprimer les liens $\{\omega_{k,j}\}$ (pour tout j) et de mettre à jour la structure du SCN en conséquence.

9.4.2.1. removeEntity$(\mathcal{E}_k)$ **: suppression des liens** $\omega_{k,j}$

Lorsque l'on décide de **retirer** une entité $\mathcal{E}_k$ d'un **SCN** (Synergistic Connection Network) en **apprentissage continu**, il convient de gérer la **suppression** de toutes ses liaisons $\{\omega_{k,j}\}$ pour garantir la cohérence de la matrice $\{\omega_{i,j}\}$. Cette opération se justifie lorsque $\mathcal{E}_k$ devient obsolète, inappropriée ou simplement hors d'usage (voir aussi [9.4.2.2] et [9.4.2.3] pour la poursuite du nettoyage ou la réallocation de clusters). Les parties A et B ci-après décrivent, dans un style proche d'un document de recherche, la mécanique générale de la suppression, ainsi que les conséquences possibles sur la topologie. Les **formules** affichées traduisent la remise à zéro des liens et, éventuellement, la désactivation progressive, tandis que des références à la dynamique DSL (chap. 4) ou aux mécanismes d'inhibition (chap. 7.4) situent le contexte.

A. Mécanique Générale de la Suppression

Dès lors qu'on juge nécessaire d'enlever $\mathcal{E}_k$ du **SCN**, on doit annuler les liaisons $\omega_{k,j}$ et $\omega_{j,k}$. Sur le plan **mathématique**, cela se traduit souvent par la règle :

$$\forall\, j \neq k, \quad \omega_{k,j}(t+1) \;\leftarrow\; 0, \quad \omega_{j,k}(t+1) \;\leftarrow\; 0.$$

Cette mise à zéro peut être immédiate, ou l'on peut se contenter de la stocker dans une structure marquant la ligne k et la colonne k comme inactives. La conséquence est que, pour chaque itération ultérieure, la mise à jour DSL ne tiendra plus compte des pondérations associées à l'indice k. Il est parfois utile de prévoir une phase de **désactivation progressive**, plutôt que d'appliquer une nullification brutale $\omega_{k,j}(t+1) = 0$, on laisse :

$$\omega_{k,j}(t+1) = (1-\alpha)\,\omega_{k,j}(t), \quad \forall j,$$

avec $\alpha \in (0,1)$. Cette décroissance est analogue au "forgiving" ou "forgetting" déjà mentionné (voir [9.3.3.2] pour la règle $\omega \leftarrow (1-\lambda)\,\omega$). On aboutit à une transition douce qui évite les chocs brutaux dans la dynamique, laissant le réseau réajuster ses autres liaisons $\{\omega_{i,j} \mid i \neq k,\ j \neq k\}$ au fil des itérations.

Sur le plan **algorithmique**, la mémoire associée à $\mathcal{E}_k$ doit éventuellement être déchargée ou archivée si l'entité portait un état interne (chap. 4.2.4). On retrouve alors une matrice $\{\omega\}$ de taille $(n-1) \times (n-1)$ ou un bloc inactif si l'on souhaite minimiser les réallocations de structure de données.

B. Conséquence sur la Topologie (Références aux Sections Suivantes)

La disparition des liaisons $\{\omega_{k,j}\}$ peut affecter la **topologie** du réseau, soit de façon négligeable si $\mathcal{E}_k$ était un nœud marginal, soit de manière drastique si $\mathcal{E}_k$ servait de **pont** entre différents clusters. La matrice ω peut ainsi voir s'envoler un sous-ensemble de connexions qui scindait un cluster unifié en plusieurs. Dans un **graphe** fortement connecté, la suppression d'un nœud pivot engendre parfois une coupure de la composante connexe en deux blocs distincts, alors que dans un graphe plus épars, l'effet reste localisé.

Sur le plan **mathématique**, la mise à jour DSL ne considérera plus aucune somme ou feedback passant par l'indice k. Si un algorithme d'inhibition (voir chap. 7.4) recourait à des sommes $\sum_{m \neq j}\ \omega_{k,m}$ ou autre, on doit retirer ces termes. Dans un système orienté, on supprime $\omega_{k,j}$ et $\omega_{j,k}$ sans que cela implique de symétrie particulière. Ce sujet se poursuivra dans [9.4.2.2] et [9.4.2.3], qui décrivent comment la topologie globale évolue et comment les clusters se réarrangent, ou se fusionnent s'il manque un nœud reliant les entités.

Conclusion

La fonction removeEntity$(\mathcal{E}_k)$ se réduit, dans un **SCN**, à annuler les liaisons $\omega_{k,j}$ et $\omega_{j,k}$. Les formules (1) et (2) signalent la possibilité d'une suppression **instantanée** ou **progressive**. Les conséquences dépendent de la place qu'occupait $\mathcal{E}_k$ dans la topologie. Si c'était un nœud secondaire, la perte est discrète ; si c'était un pivot, le réseau peut se scinder ou devoir se réorganiser en conséquence. Les **sections** [9.4.2.2] et [9.4.2.3] approfondiront la question de la réallocation et du rééquilibrage des liaisons au sein des clusters après la suppression d'une entité jugée non pertinente.

9.4.2.2. Impact sur la topologie (clusters peuvent se scinder, etc.)

Il est fréquent, dans un **Synergistic Connection Network** (**SCN**) évoluant sous un **apprentissage continu**, que la **topologie** ne se limite pas à un état figé. Au fur et à mesure que la dynamique du **DSL** (Deep Synergy Learning) modifie les pondérations $\omega_{i,j}$, on observe des regroupements ou des scissions de **clusters**, en particulier lorsqu'un flux de données ou des modifications de paramètres (inhibition, saturation, etc.) se produisent. Les principes décrits ici (9.4.2.2) s'appuient sur la logique d'ajout ou de retrait d'entités (sections précédentes) et sur la capacité du SCN à réallouer ses liaisons.

A. Rappel : Logique d'Auto-Organisation et Clusters

Dans un **DSL**, on appelle *cluster* un **sous-ensemble** $\{\mathcal{E}_k\}$ dont les **pondérations** $\omega_{k,k'}$ s'avèrent élevées, signifiant une **cohérence** ou une **synergie** interne. Le SCN se **stabilise** localement lorsque ces liaisons internes restent supérieures à un certain seuil, ou lorsqu'aucun mécanisme

d'inhibition ou de décroissance ne parvient à les briser. Cependant, l'arrivée ou la sortie d'entités, tout comme la simple variation de la fonction $S(i,j)$, peuvent entamer la **cohésion** interne d'un cluster ou, à l'inverse, relier plusieurs clusters distincts. Sur un plan mathématique, la mise à jour

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

intervient à chaque itération, si bien qu'un écart de $\Delta\omega_{i,j} \leq 0$ (décroissance) peut scinder progressivement un cluster, tandis qu'un $\Delta\omega_{i,j} > 0$ (renforcement) peut unifier deux sous-ensembles.

B. Causes et Mécanismes de Scission

Un **cluster** $\mathcal{C}$ se scinde typiquement lorsque ses membres n'entretiennent plus de **synergie** suffisante les uns envers les autres. Si l'on note

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

et que $S(i,j,t)$ chute durablement, le terme $-\tau\,\omega_{i,j}(t)$ prédomine, amenuisant $\omega_{i,j}$. Plusieurs paires internes à $\mathcal{C}$ peuvent simultanément s'éroder. On aboutit à une **réduction** de la somme

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

entraînant, à terme, une **scission** de $\mathcal{C}$ en deux sous-clusters. Il se peut aussi qu'un mécanisme d'inhibition global (chap. 7.4) ou un **feedback** top-down (chap. 6) renforce cette **baisse**, imposant un paramètre γ plus fort pour soustraire $\gamma \sum_{k\neq i} \omega_{k,i}$ dans la mise à jour. Un cluster perd alors sa cohésion s'il ne reçoit plus de renforcement positif.

C. Fusions et Émergence de Nouveaux Clusters

Si la scission reflète une **rupture** de cohérence, on peut observer l'effet inverse lorsqu'un **flux** de données ou l'apport d'une **nouvelle** entité (voir [9.4.1.3]) connecte deux clusters jusque-là séparés. Une entité $\mathcal{E}_{\text{new}}$ très proche à la fois d'un cluster $\mathcal{C}_1$ et d'un autre $\mathcal{C}_2$ peut servir de "**pont**", accroissant la **synergie** inter-groupe et amenant la dynamique à **fusionner** $\mathcal{C}_1$ et $\mathcal{C}_2$. Dans un tel cas, la **matrice** $\{\omega\}$ voit apparaître plusieurs liens transversaux $(i \in \mathcal{C}_1, j \in \mathcal{C}_2)$ renforcés, provoquant la mise en commun de ces entités en un "super-cluster". On parle alors d'**unification** de clusters.

D. Conséquences sur la Lecture du SCN

La **topologie** d'un **SCN** n'est donc pas stable. L'apprentissage continu, le recours à l'inhibition, ou l'insertion/suppression d'entités en flux imposent une suite d'**états** $\boldsymbol{\Omega}(t)$. D'un point de vue mathématique, l'ensemble des composantes connexes (clusters) change au fil du temps. Certaines entités se "désolidarisent" de leur groupe, d'autres rejoignent un ensemble plus vaste. Les schémas de scission/fusion augmentent la **plasticité** du DSL. Le réseau ne se fige pas dans une partition ancienne, il reste capable de **refléter** les nouveautés. Cependant, d'un point de vue **interprétation** et **analyse**, ce caractère éminemment dynamique peut complexifier la **lecture** du SCN. Il peut être utile de tracer l'**historique** des clusters, en observant un groupement $\mathcal{C}$ se diviser en $\mathcal{C}_1 \cup \mathcal{C}_2$ ou s'agréger avec d'autres, est un signal sur l'évolution du réseau. Les chapitres ultérieurs discutent plus avant la gestion de ces réagencements (feedback macro, heuristiques, paramétrages d'inhibition).

Conclusion (9.4.2.2). L'impact des modifications incrémentales sur la **topologie** d'un **SCN** peut conduire à la **scission** de clusters (lorsque la synergie interne décline, ou des liens sont inhibés) ou à la **fusion** (quand une entité pont crée un rapprochement entre groupes). Dans un DSL en flux, ces réajustements sont un signe de la plasticité du réseau, qui cherche constamment un équilibre face aux variations de $S(i,j)$ et aux mécanismes d'inhibition. Le résultat final est une **auto-organisation** plus flexible, mais exigeant une vigilance constante quant à la cohérence et la lisibilité des regroupements d'entités dans la matrice $\{\omega_{i,j}\}$.

9.4.2.3. Maintenance de l'index ou des structures d'adjacence

L'**évolution** d'un **Synergistic Connection Network** (*SCN*) en **apprentissage continu** exige un **entretien** permanent des **structures de données** internes. L'arrivée d'une **nouvelle entité** $\mathcal{E}_{n+1}$ (voir [9.4.1.1]) ou la suppression de $\mathcal{E}_k$ (voir [9.4.2.1]) modifient la taille et la composition du réseau, tandis que l'adaptation continue des pondérations $\omega_{i,j}$ (voir [9.4.2.2] et chapitre 4) peut faire apparaître ou disparaître des liaisons au fil du temps. Pour assurer un accès rapide et cohérent, il est nécessaire de **mettre à jour** l'**index** (k-d tree, vantage-point tree, structures de hachage, etc.) et la **représentation d'adjacence** (matrice, liste de voisins). Les sous-sections A et B clarifient ces concepts, puis la section C expose une **formulation** mathématique illustrant la "projection" qui consolide la liste d'adjacence ou la structure sparses. Enfin, la conclusion (9.4.2.3) récapitule l'importance de cette maintenance pour un **SCN** en flux.

A. Notion d'indexation et représentation d'adjacence

Dans un **SCN** de taille nn, il peut être prohibitif de scanner $\mathrm{O}(n)$ entités à chaque fois que l'on veut trouver les **voisins** pertinents de $\mathcal{E}_i$. Un **index** spatial (k-d tree, ball tree, vantage-point tree) ou un **index** sémantique (p. ex. table de hachage sur les paires (i,j)) permet de **retrouver** en $\mathrm{O}(\log n)$ ou $\mathrm{O}(n^{\alpha})$ les entités proches. Ces structures doivent être **incrémentales**. Lorsqu'on insère $\mathcal{E}_{n+1}$, on modifie localement l'index ; quand on supprime $\mathcal{E}_p$, on la retire ou on fusionne des feuilles, etc.

La **structure d'adjacence** complète la logique d'index. Si l'on opte pour une **matrice dense** $\boldsymbol{\Omega} \in \mathbb{R}^{n\times n}$, on stocke $\omega_{i,j}$ explicitement pour chaque paire (i,j). Insérer ou retirer une entité implique d'agrandir ou de réduire la matrice, pouvant entraîner un coût en $\mathrm{O}(n^2)$. Dans une **liste** d'adjacence, on ne conserve que les liaisons $\omega_{i,j}$ au-dessus d'un certain **seuil** θ. Chaque entité ii possède alors un **voisinage** $\mathrm{Adj}(i)$, c'est-à-dire une liste des jj pour lesquels $\omega_{i,j} > \theta$. Cette technique est avantageuse si le SCN reste **sparse** (inhibition latérale, règles d'oubli).

B. Évolution du SCN en temps réel : conséquences sur l'index et l'adjacence

Un **SCN** en flux ou en adaptation constante, comme décrit au chapitre 9, se voit modifié par :

- *Insertion* addEntity$(\mathcal{E}_{n+1})$: on insère $\mathcal{E}_{n+1}$ dans l'index spatial, on crée la nouvelle ligne et colonne dans la matrice, ou on initialise $\mathrm{Adj}(n+1)$ dans la liste d'adjacence pour les liens dépassant θ\theta.
- *Suppression* removeEntity$(\mathcal{E}_p)$: on élimine $\mathcal{E}_p$ de l'index, on met la ligne-colonne p à 0 (ou on la supprime) dans la matrice dense, ou on retire systématiquement les références (p,j) et (j,p) dans les listes.

- *Variation des liens* $\omega_{i,j}(t)$: la dynamique **DSL** impose $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$, qui peut faire grimper ou chuter $\omega_{i,j}$. Si on a défini un *seuil* θ, il se peut qu'une liaison $\omega_{i,j}$ devienne supérieure à θ alors qu'elle ne l'était pas, imposant de **l'ajouter** dans $\mathrm{Adj}(i)$. Ou, au contraire, elle peut tomber au-dessous de θ, imposant de **la retirer**.

Ces modifications incrémentales appellent une **mise à jour** continue de l'index et des structures d'adjacence, pour en maintenir la cohérence.

C. Formulation mathématique : projection par seuil sur la liste d'adjacence

Dans la version "liste d'adjacence avec seuil θ", on explicite souvent la **projection** Π_θ. Après avoir calculé la mise à jour DSL,

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

on applique :

$$\omega_{i,j}(t+1) \;\leftarrow\; \Pi_\theta\left(\omega_{i,j}^{\text{inter}}(t+1)\right),$$

où

$$\Pi_\theta(x) = \begin{cases} 0, & x < \theta, \\ x, & x \geq \theta. \end{cases}$$

Cette bascule (0 ou x) assure qu'on ne stocke $\omega_{i,j}$ que si elle dépasse θ. Dans la **liste** d'adjacence, cela s'interprète par : si $\omega_{i,j}(t+1) \geq \theta$, on **ajoute** $\left(j, \omega_{i,j}(t+1)\right)$ à $\mathrm{Adj}(i)$, sinon on la **retire**. Le coût d'un tel "post-traitement" est proportionnel au **degré** moyen $\overline{d}$ de chaque entité, ou à la liste d'événements ($\omega_{i,j}$ franchissant θ dans un sens ou l'autre). D'un point de vue algorithmique, on contrôle la dynamique :

$$\omega_{i,j}(t+1) = \max\big\{0,\ \omega_{i,j}^{\text{inter}}(t+1)\big\} \quad \text{si on prend } \theta = 0,$$

ou on impose un $\theta > 0$ plus élevé pour renforcer la **sparsité**. Ce mécanisme de "projection par seuil" s'intègre naturellement à la *matrice sparses* ou aux *listes d'adjacence*, tout en reliant l'**index** pour trouver de nouveaux candidats liens si $\omega_{i,j}$ vient de passer θ.

Conclusion

Dans un **SCN** évolutif, la **maintenance** de l'**index** (k-d tree, vantage-point tree, hachage…) et des **structures d'adjacence** (matrice, liste de voisins…) est indispensable pour que la **dynamique** DSL (ajout, suppression, ajustement de $\omega_{i,j}$) reste **cohérente**. Les **formules** de projection par seuil, par exemple

$$\omega_{i,j}(t+1) \;\leftarrow\; \begin{cases} 0, & \omega_{i,j}^{\text{inter}}(t+1) < \theta, \\ \omega_{i,j}^{\text{inter}}(t+1), & \omega_{i,j}^{\text{inter}}(t+1) \geq \theta, \end{cases}$$

garantissent une structure claire. Un lien inactif est éliminé, un lien dépassant θ est inséré, tandis que l'**index** se met à jour pour repérer la position de chaque entité. Sans cette maintenance, l'**apprentissage continu** s'effondrerait sous des incohérences (liaisons fantômes, index divergents), ou sous une complexité excessive $O(n^2)$. Avec ces règles, on préserve la **plasticité**

du SCN, capable de gérer en temps réel l'insertion/suppression de nœuds et la réallocation de liens, tout en contrôlant le **coût** et la **cohérence** interne.

9.4.3. Scénarios Multi-Arrivées

Lorsque le **SCN** (Synergistic Connection Network) se trouve confronté à l'arrivée massive ou simultanée d'un **lot** d'entités (par opposition à l'arrivée d'entités une par une), on parle souvent de **batch streaming**. Cela requiert un **traitement** légèrement différent pour intégrer ces nouvelles données dans la structure du réseau, car la mise à jour locale (cas "un par un") ne suffit plus à absorber soudainement un ensemble potentiellement vaste d'entités. Dans cette section (9.4.3), on distingue **trois** points clés (9.4.3.1–9.4.3.3), avec un focus ici sur **9.4.3.1**.

9.4.3.1. Arrivée simultanée d'un lot de nouvelles entités (batch streaming)

Dans un **Synergistic Connection Network** (SCN) en **apprentissage continu**, il arrive qu'un ensemble complet de nouvelles entités $\{\mathcal{E}_{n+1}, \dots, \mathcal{E}_{n+K}\}$ parvienne en un **unique** bloc, plutôt qu'une entité isolée à la fois. Le terme *batch streaming* reflète cette situation, où le système reçoit périodiquement des lots (documents, frames vidéo, relevés sensoriels) et doit en mettre à jour simultanément la matrice $\{\omega_{i,j}\}$. Les sections A, B, C et D ci-après décrivent la logique d'insertion en bloc, la modélisation mathématique de la mise à jour, des exemples et la conclusion (9.4.3.1). Les formules insistent sur la façon de limiter le recalcul global et de préserver la cohérence DSL sans un traitement $O(K \times n)$ trop lourd.

A. Contexte et Motivation

Un SCN traité en *flux* reçoit, dans certains scénarios, un **lot** simultané d'entités. Cela se produit lorsqu'un module amont n'émet pas un flux continu, mais plutôt des **paquets** de taille K. On se demande comment **insérer** les entités $\{\mathcal{E}_{n+1}, \dots, \mathcal{E}_{n+K}\}$ dans la matrice $\{\omega_{i,j}\}$ en évitant de recalculer un $O(K \times n)$ complet. Les principes d'insertion individuelle (voir [9.4.1.1] sur addEntity) se répliquent, mais on peut regrouper les opérations communes ou diffuser l'inhibition (chap. 7) en fin de cycle.

B. Stratégies de Mise à Jour

Au lieu de gérer chaque entité $\mathcal{E}_{n+k}$ séparément, on peut adopter différentes stratégies d'insertion.

Une **approche naïve** consiste à réitérer addEntity pour chacune des K entités, ce qui implique d'effectuer K fois la recherche de voisins (k-NN) et l'initialisation des poids $\omega_{(n+k),\,j}$. Cette méthode peut coûter $O(K \times n)$ ou $O(K \times n \log n)$ si un index spatial et un tri partiel sont utilisés. Lorsque K est grand, cette approche devient rapidement coûteuse.

Une alternative plus efficace est de **traiter en bloc**, en construisant pour chaque $\mathcal{E}_{n+k}$ un voisinage $N_{n+k} \subset \{1, \dots, n\}$. Cela revient à calculer la **synergie** $S(\mathcal{E}_{n+k}, \mathcal{E}_j)$ sur un sous-ensemble réduit de j (par exemple, en sélectionnant les k plus proches voisins en distance) pour tous $k \in \{1, \dots, K\}$. Puis on applique la mise à jour DSL :

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

On crée ensuite des liens $\omega_{j,(n+k)}$ si le réseau est symétrique, avant d'étendre l'inhibition ou la saturation (chap. 7) *après* avoir inséré les K entités. Cette pratique réduit potentiellement le nombre d'opérations, puisqu'on recourt à un unique "macro-cycle" d'inhibition pour tout le **batch**.

C. Modélisation Mathématique : $\Delta_{DSL}^{(batch)}$

Si l'on regroupe toutes les nouvelles entités dans un cycle unique, on peut formaliser la mise à jour globale comme :

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

où $\Delta_{\mathrm{DSL}}^{(\mathrm{batch})}(i,j)$ incorpore la somme des "influences" générées par le lot $\{\mathcal{E}_{n+1}, \dots, \mathcal{E}_{n+K}\}$. Plus concrètement, pour chaque $\mathcal{E}_{n+k}$, on actualise localement $\omega_{(n+k),i}$. Puis on applique l'inhibition latérale $\gamma \sum_{k \neq i} \omega_{\dots}$ ou tout autre module d'optimisation (chap. 7). Un nombre restreint d'itérations (par exemple 5) peut suffire pour intégrer le lot, plutôt que de lancer un recalcul complet pour chacun des K nœuds.

Cette approche se compare à une *descente de gradient par mini-batch* en apprentissage profond, où l'on cumule l'effet de K exemples avant d'ajuster les paramètres. Ici, on cumule l'effet des K entités avant de diffuser l'inhibition ou la régulation globale.

D. Exemples et Gains Pratiques

Exemple 1 : Système de **documents** textuels, recevant chaque matin un bloc de 100 articles. On **scanne** ces 100 documents pour leur associer un voisinage (k-NN en embedding), puis on applique la règle (1) pour créer $\omega_{(n+k),j}$. On clôture par un unique "tour" d'inhibition latérale, réduisant le temps global par rapport à 100 tours consécutifs.

Exemple 2 : Flux de **frames** vidéo, où l'on accumule 30 frames (une seconde de vidéo) en une micro-batch. L'ajout simultané de ces 30 entités dans le SCN minimise le recalcul d'inhibition ou de saturations, d'où une meilleure scalabilité.

Dans chaque cas, l'idée est d'**absorber** un mini-lot de taille K, l'insérer localement (k-NN, distances) en $O(K \times n)$ ou $O(K \times n \log n)$, puis diffuser les corrections par inhibition dans un nombre restreint d'itérations, assurant un seul "macro-cycle".

Conclusion.

L'arrivée **simultanée** d'un lot de nouvelles entités — **batch streaming** — s'intègre au **SCN** en appliquant la logique *addEntity* non plus entité par entité, mais pour l'ensemble $\{\mathcal{E}_{n+1}, \dots, \mathcal{E}_{n+K}\}$. On identifie le voisinage de chacune (k plus proches, etc.), on calcule la synergie, on initialise ω selon (1), puis on exécute une **phase** d'inhibition et de stabilisation pour tout le lot. Cette approche économise des ressources par rapport à l'addition séquentielle strictement, tout en maintenant la **philosophie** du DSL. La mise à jour reste locale (voisinage) puis globale (inhibition, recuit). Il se dégage un **compromis** pertinent entre la rigueur d'un traitement entité par entité et la lourdeur d'un recalcul exhaustif pour chaque nouvelle entité.

9.4.3.2. Intégration Progressive, Re-Agrégation de Clusters si Besoin

Lorsqu'un **Synergistic Connection Network** (**SCN**) s'inscrit dans un **DSL** (Deep Synergy Learning) en **flux continu**, de nouvelles entités peuvent régulièrement arriver, modifiant la

structure $\{\omega_{i,j}\}$. Il n'est pas souhaitable de réinitialiser toute la matrice à chaque entité entrante, car le SCN s'auto-organise déjà sur la base d'une configuration antérieure. Il faut donc insérer ces entités de manière **incrémentale**, tout en autorisant, si nécessaire, la **re-agrégation** (fission ou fusion) de clusters antérieurs. Les principes abordés dans les sections A, B, C et D (9.4.3.2) indiquent comment cette intégration progressive se concrétise et comment le SCN peut réorganiser des clusters en conséquence.

A. Intégration Progressive des Nouvelles Entités

Lorsque survient une **nouvelle** entité $\mathcal{E}_{\text{new}}$, on procède souvent par des étapes **locales** plutôt que de tout recalculer. On choisit un voisinage $\mathrm{N}(\text{new}) \subseteq \{1, \dots, n\}$ (par ex. k plus proches entités en distance ou un ϵ-rayon) et on amorce les liaisons $\omega_{\text{new},\,j}$:

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

ce qui évite un $\mathrm{O}(n^2)$ intégral. Cette mise à jour peut être réitérée sur quelques itérations, les entités $\mathcal{E}_j$ extérieures s'ajustant également, mais l'essentiel demeure, $\mathcal{E}_{\text{new}}$ ne "pollue" que ses voisins directs, le reste du SCN connaissant une perturbation moins lourde.

Ce schéma encourage une **coexistence** entre la configuration antérieure, déjà stabilisée, et l'apport local de la nouvelle entité. On n'abandonne pas les clusters formés ; on les étend ou on influe localement. Sur le plan **algorithmique**, cela s'apparente à ajouter une "sonde" $\mathcal{E}_{\text{new}}$ dans le tissu existant, sans démolir la structure.

B. Re-Agrégation de Clusters

L'arrivée d'une entité $\mathcal{E}_{\text{new}}$ peut amener des bouleversements. Si la synergie qu'elle partage est forte avec deux clusters $\mathcal{C}_\alpha$ et $\mathcal{C}_\beta$, elle peut déclencher une **fusion**. À l'inverse, si $\mathcal{E}_{\text{new}}$ génère un sous-ensemble plus cohérent, un cluster existant peut se **scinder**.

Lorsqu'on regarde la cohérence d'un cluster $\mathcal{C}$, on calcule souvent

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

et on surveille comment $\Omega(\mathcal{C})$ varie face à la présence d'une nouvelle entité reliant $\mathcal{C}$ à d'autres blocs. Sur le plan **mathématique**, si un sous-groupe $\mathcal{C}_1 \subset \mathcal{C}$ s'avère plus homogène que le reste, on peut observer un $\Delta\omega$ négatif sur des liens inter-sous-groupes, aboutissant à une scission. À l'inverse, si la présence de $\mathcal{E}_{\text{new}}$ intensifie la synergie entre deux clusters, une fusion s'opère. Dans tous les cas, on retrouve la logique *fission/fusion* caractéristique d'un SCN adaptatif (voir chap. 9.4.3.3 sur la plasticité).

C. Approches Mathématiques

On peut considérer un "post-traitement" après l'intégration de $\mathcal{E}_{\text{new}}$, rebalayant les liaisons. On obtient alors :

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

Si on compare la somme inter-bloc $\sum_{i \in \mathcal{C}_\alpha,\, j \in \mathcal{C}_\beta} \omega_{i,j}$ avant et après, on peut décider d'une fusion s'il y a un gain notable. Pour la scission, on étudie la partition interne d'un cluster $\mathcal{C}_\alpha$. Si $\Omega(\mathcal{C}_\alpha)$ se réorganise autour de deux sous-blocs fortement soudés et entre eux plus faiblement liés, on

déclenche une scission. Sur le plan **algorithmique**, on peut le faire tous les N itérations, ou de manière continue dès qu'un signal de réorganisation apparaît.

D. Conclusion

L'**intégration progressive** des entités nouvellement arrivées et la **re-agrégation** ultérieure de clusters distinguent un **SCN** adaptatif d'une simple structure figée. Sur le plan **mathématique**, on inscrit la nouvelle entité $\mathcal{E}_{\text{new}}$ dans un **voisinage** restreint, évitant un recalcul complet, puis on laisse la dynamique (éventuellement couplée à un post-traitement) décider de fusions, scissions ou réallocations de sous-groupes. Le résultat est un **réseau** qui ne se "détruit" jamais totalement, préservant la **mémoire** de la configuration antérieure tout en **incorporant** les nouveautés de façon incrémentale. La plasticité ainsi acquise permet à un DSL en flux (chap. 9) d'accommoder des changements constants, tant en synergie qu'en composition, sans retomber sur un apprentissage batch complet.

9.4.3.3. Exemples : flux de documents textuels, de frames vidéo

Les **données** arrivantes sous forme de **flux** constituent un cas d'usage central pour un **Synergistic Connection Network** (**SCN**) en **apprentissage continu**. Les deux exemples classiques sont un **flux de documents** textuels et un **flux de frames** vidéo. À chaque nouvel item (article, tweet, image), le **DSL** (Deep Synergy Learning) intègre l'entité, calcule sa **synergie** avec quelques entités préexistantes, et met à jour la matrice $\{\omega_{i,j}\}$ de manière incrémentale, évitant ainsi tout réapprentissage complet. Les sous-sections A et B décrivent respectivement le flux textuel et le flux vidéo, illustrant de manière **mathématique** la mise en œuvre des équations de synergy et de pondérations ω. La section C conclut (9.4.3.3) sur les avantages et défis.

A. Flux de Documents Textuels

Un **SCN** alimenté par un flux de **documents** ($\mathcal{D}_1$, $\mathcal{D}_2$, ...) peut, à chaque **arrivée** d'un nouveau document $\mathcal{D}_{n+1}$, l'**insérer** comme entité $\mathcal{E}_{n+1}$. Sur le plan **mathématique**, on associe au document un **vecteur** ou **embedding** $\mathbf{x}_{n+1} \in \mathbb{R}^d$. La **synergie** entre deux documents $\mathcal{D}_{n+1}$ et $\mathcal{D}_j$ se calcule souvent via une **similarité cosinus** :

$$S(\mathcal{D}_{n+1}, \mathcal{D}_j) = \frac{\mathbf{x}_{n+1} \cdot \mathbf{x}_j}{\| \mathbf{x}_{n+1} \| \, \| \mathbf{x}_j \|}.$$

Pour limiter la complexité, on se focalise sur le voisinage $\mathrm{N}(n+1) \subseteq \{1, \dots, n\}$ (k plus proches documents). On applique alors la mise à jour **DSL** :

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

Les autres liaisons restent à 0 ou à une très faible valeur initiale. Une fois cette insertion faite, on peut pratiquer l'**inhibition** (chap. 7.4) :

$$\omega_{(n+1),\,j}(t+1) \;\leftarrow\; \max\left(0,\, \omega_{(n+1),\,j}(t+1) - \gamma \sum_{k \neq j} \omega_{(n+1),\,k}\,(t+1)\right),$$

qui évite la prolifération de liens moyens. Ainsi, le **SCN** construit des **clusters** de documents, reflétant la proximité sémantique ou thématique, sans tout réentraîner à chaque document ajouté.

B. Flux de Frames Vidéo

Le **flux** vidéo ($\mathcal{F}_1$, $\mathcal{F}_2$, ...) se traite de façon similaire. Chaque frame $\mathcal{F}_t$ devient une **entité** (nouveau nœud) munie d'un **descripteur** $\mathbf{v}_t \in \mathbb{R}^d$ extrait d'un modèle visuel (CNN ou autre). La **synergie** entre deux frames peut être définie via une **distance** ou une **similarité** ; par exemple :

$$S(\mathcal{F}_t, \mathcal{F}_s) = \exp(-\alpha \parallel \mathbf{v}_t - \mathbf{v}_s \parallel^2).$$

Pour un nouveau frame $\mathcal{F}_{m+1}$, on calcule :

$$\omega_{(m+1),j}(t+1) = \omega_{(m+1),j}(t) + \eta \left[S\left(\mathcal{F}_{m+1}, \mathcal{F}_j\right) - \tau\, \omega_{(m+1),j}(t)\right], \quad j \in \mathrm{N}(m+1).$$

Dans certains scénarios, on pondère la synergie par un facteur temporel $\exp(-\beta|\, m+1-j|)$ pour accentuer la proximité entre frames successifs. De même, on peut appliquer une **inhibition** (analog. (3)) ou une **saturation** $\omega_{(m+1),j}(t+1) \leftarrow \min\left(\omega_{\max}, \omega_{(m+1),j}(t+1)\right)$. Ce processus incrémental génère des **clusters** de frames, correspondant à des segments vidéo homogènes (p. ex. un mouvement continu, une scène stable).

C. Avantages et Défis

Un **SCN** traitant un **flux** de documents ou de frames présente des avantages et des défis.

En termes d'**avantages**, il permet de **stocker** la mémoire des items antérieurs (articles, frames) tout en intégrant progressivement les nouveaux, en limitant le recalcul à un voisinage local $\mathrm{N}(n+1)$. La formation de **clusters** (thèmes textuels ou segments vidéo) évolue en continu, permettant au SCN de fusionner ou de scinder des sous-groupes en fonction de la synergie, sans nécessiter un re-entraînement complet.

Cependant, des **défis** apparaissent lorsque le flux devient massif et que n croît rapidement, risquant d'engendrer un surcoût de l'ordre de $\mathrm{O}(n)$ par item. Il devient alors essentiel d'utiliser un **index** (chap. 9.4.2.3) pour accélérer la recherche du voisinage $\mathrm{N}(n+1)$. Par ailleurs, l'application de **règles** d'oubli ou de suppression d'anciennes entités (chap. 9.4.2.1) est indispensable pour contenir la taille du réseau et éviter une explosion de complexité.

Si un flux textuel envoie 50 articles par jour, on intègre chaque bloc (voir [9.4.3.1] "batch streaming") en calculant la similarité cosinus (1) ou la distance (4), puis on insère ces 50 entités avec la mise à jour (2) ou (5) ; on applique ensuite un cycle d'inhibition (3). Si un article se révèle coller à un sous-groupe existant, la synergie interne de ce groupe croît ; si plusieurs articles créent un nouveau thème, le SCN forme un cluster distinct.

Conclusion.

Les **flux** de documents textuels et de frames vidéo servent d'**exemples** éloquents pour illustrer la **démarche** DSL en **temps réel** : chaque item entré s'inscrit dans le **SCN** via la mise à jour locale (2), (3), (5), permettant la formation incrémentale de clusters (thèmes de documents, séquences vidéo cohérentes). L'**adaptation** s'effectue sans retomber sur un recalcul $\mathrm{O}(n^2)$ à chaque insertion, grâce au voisinage $\mathrm{N}(n+1)$. Un index (k-d tree, vantage-point) limite la complexité, et des mécanismes (inhibition, saturation) canalisent l'extension du graphe. Ainsi, on gère efficacement de larges **flux** tout en préservant la **plasticité** dynamique du SCN.

9.5. Stabilité de l'Auto-Organisation en Flux Continu

Dans les scénarios où le **SCN** (Synergistic Connection Network) se met à jour de façon **incrémentale** en présence d'un **flux** de données continu (chap. 9.4), se pose la question de la **stabilité**. Le réseau peut-il parvenir à un état (ou un ensemble d'états) relativement stable, alors même que de nouvelles entités ou de nouvelles mesures arrivent en permanence ? Comment définir la **"stabilité"** quand, par essence, la dynamique ne se **termine** jamais ? La section 9.5 aborde ces problématiques, en distinguant les phénomènes de **stabilité locale** (9.5.1), d'**oscillations permanentes** (9.5.2) et des approches de **fenêtre glissante** (9.5.3) pour réguler la mémoire du réseau.

9.5.1. Notion de Stabilité Locale

Le fait qu'un **SCN** incrémental ne se "fige" pas entièrement (dans un flux sans fin) ne signifie pas qu'aucune forme de **stabilité** ne puisse émerger. On peut mesurer la stabilité **localement**. Sur une **fenêtre temporelle** donnée (ex. 100 itérations), on analyse si les liaisons $\omega_{i,j}$ varient peu, signifiant qu'un **cluster** se maintient au moins transitoirement.

9.5.1.1. Même si le SCN ne se fige jamais entièrement, on peut analyser la "stabilité" sur une fenêtre temporelle (ex. 100 itérations)

Lorsqu'un **Synergistic Connection Network** (**SCN**) évolue en **flux** sans jamais se figer totalement, il demeure souvent utile de disposer d'un **critère** ou d'une **mesure** pour déceler, dans un intervalle de temps donné, si les *liaisons* $\omega_{i,j}$ et les *clusters* qui en résultent restent à peu près constants. La notion de **fenêtre** temporelle ΔT (par exemple, 100 itérations) et de **variance** permet de qualifier une "stabilité locale" malgré l'absence de stabilité définitive. Les sections A, B, C ci-après (9.5.1.1) développent l'idée. La section A rappelle le principe de fenêtre fixe ou glissante, la section B présente la **variance** comme critère clé, et la section C décrit la stabilité des clusters dans ce cadre, avant de conclure sur l'interprétation mathématique.

A. Fenêtre Fixe ou Glissante

Sur un **SCN** en **apprentissage continu**, la mise à jour DSL s'applique :

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

ce qui peut perdurer indéfiniment (chapitres précédents). Même sans fin, on peut choisir un **intervalle** ΔT (par ex. 100 itérations) et regarder, pour un temps d'origine t, l'évolution $\omega_{i,j}(t+\tau)$ pour $\tau = 0, \dots, \Delta T - 1$. Si l'on constate peu de fluctuations pendant cette *fenêtre*, on dira que le lien $\omega_{i,j}$ se "stabilise" localement, et qu'en conséquence, tout un *cluster* peut être qualifié de stable dans $[t,\ t+\Delta T]$. Certaines études privilégient une **fenêtre glissante**, où à chaque itération t, on observe l'intervalle $[t,\ t+\Delta T]$ et on fait évoluer cette fenêtre de manière incrémentale, en la reculant ou en l'avançant.

B. Critère de Variance

On définit souvent une **variance** locale pour $\omega_{i,j}$. Sur l'intervalle $[t,\ t+\Delta T]$, on définit :

$$\bar{\omega}_{i,j} = \frac{1}{\Delta T} \sum_{\tau=0}^{\Delta T-1} \omega_{i,j}(t+\tau)$$

et

$$\sigma_{i,j}^2(\Delta T;\ t) = \frac{1}{\Delta T} \sum_{\tau=0}^{\Delta T-1} (\omega_{i,j}(t+\tau) - \bar{\omega}_{i,j})^2.$$

Si $\sigma_{i,j}^2$ est **faible**, cela signifie que $\omega_{i,j}$ n'a que très peu varié dans l'intervalle. Il existe un **seuil** ϵ^2 tel que si $\sigma_{i,j}^2(\Delta T;\ t) < \epsilon^2$, on dit que le lien (i,j) est "quasi stable" ou "faiblement fluctuant" dans ce créneau. On peut également définir une fonction d'**énergie** ou de **variance** globale rassemblant toutes les liaisons, mais, de manière plus simple, on se concentre parfois sur les paires $\omega_{i,j}$ surpassant un seuil de pondération pour ne pas en tenir compte si $\omega_{i,j} \approx 0$.

C. Clusters Stables

Au-delà de la stabilité de chaque lien, on peut se pencher sur la **cohésion d'un cluster** $\mathcal{C} \subseteq \{1, \dots, n\}$. On définit :

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

où la somme est effectuée sur tous les couples (i,j) appartenant au cluster (ou sur la moyenne $\frac{1}{|\mathcal{C}|^2}$). Si cette fonction $C(\mathcal{C}, \cdot)$ ne fluctue guère sur l'intervalle $[t,\ t+\Delta T]$, le cluster $\mathcal{C}$ est réputé "**stable**" dans cet intervalle. On obtient une variance $\sigma_{\mathcal{C}}^2(\Delta T;\ t)$:

$$\sigma_{\mathcal{C}}^2(\Delta T;\ t) = \frac{1}{\Delta T} \sum_{\tau=0}^{\Delta T-1} (C(\mathcal{C},\ t+\tau) - \overline{C}_{\mathcal{C}})^2,$$

avec

$$\overline{C}_{\mathcal{C}} = \frac{1}{\Delta T} \sum_{\tau=0}^{\Delta T-1} C\ (\mathcal{C},\ t+\tau).$$

Si cette variance $\sigma_{\mathcal{C}}^2$ demeure **en dessous** d'un certain ϵ^2, on conclut que le cluster $\mathcal{C}$ est *relativement figé* dans la période $[t,\ t+\Delta T]$. Il peut redevenir instable plus tard, si de nouveaux flux ou entités perturbent ses liaisons.

Exemple Numérique

Au temps $t = 10\,000$, supposons qu'un cluster $\mathcal{C}$ constitué des entités {1,2,3} semble peu fluctuer. On calcule $\sigma_{i,j}^2(\Delta T;\ 10\,000)$ pour $\Delta T = 100$. Si $\max\sigma_{i,j}^2 < 10^{-4}$, on en déduit que la cohésion $\omega_{(1,2)}$, $\omega_{(2,3)}$, $\omega_{(1,3)}$ demeure quasi inchangée dans $[10\,000,\ 10\,100]$. On qualifie {1,2,3} de cluster *stable localement*, tout en gardant à l'esprit qu'à l'itération $11\,000$, un flux peut *réactiver* la dynamique et briser ce cluster.

D. Interprétation : Transitoire vs. Durable

Puisqu'un SCN en flux continue de recevoir des données, la stabilité n'y est pas définitive. On ne parle que d'une **stabilité locale** ou transitoire. Sur le plan **mathématique**, la mise à jour

$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$ peut cesser de fluctuer dans une zone du réseau si plus aucune synergie nouvelle ne la perturbe. D'autres zones peuvent demeurer très actives et fluctuantes. Si l'on injecte du bruit ou si l'environnement évolue de façon répétée, la zone stable pourra être déstabilisée ultérieurement. Mais dans la fenêtre ΔT, la variance s'avère modeste et on en déduit que ce *cluster* ou ces *liaisons* se comportent comme quasi figées.

Une vision plus avancée (avec injection stochastique) assimile la mise à jour DSL à un **processus** de type markovien (chap. 2.3.4 potentiel). La "stabilité" locale signifie que le processus demeure dans un sous-ensemble d'états correspondant à l'arrangement en question, avec une probabilité élevée. Ceci illustre la **coexistence** de "**périodes**" ou "**régimes**" stables, entrecoupées de phases de réorganisation marquées.

Conclusion.

Même si un **SCN** en flux ne **cesse** de recevoir des entités (ou d'être mis à jour), on peut, en pratique, analyser la **stabilité** sur une **fenêtre** ΔT (par ex. 100 itérations). On définit la variance de $\omega_{i,j}$ (cf. équations de la section B) ou de la cohésion d'un cluster (section C). Si cette variance est inférieure à un **seuil** ϵ^2, on dit que la zone du réseau (ou le cluster concerné) est "stable" localement dans $[t,\ t + \Delta T]$. Cela fournit une **photographie** quasi stationnaire, aidant à repérer, au sein d'un SCN globalement instable, certaines **périodes** ou **sous-réseaux** figés sur un laps de temps déterminé. L'**analyse** multifenêtre (petite ΔT vs. grande ΔT) précise ainsi la durée de vie de ces îlots de stabilité locale.

9.5.1.2. Critères : la variance de $\omega_{i,j}$ devient faible sur un sous-ensemble, signifiant un cluster stable

L'émergence d'un **cluster** stable dans un **Synergistic Connection Network** (SCN) en **apprentissage continu** peut se diagnostiquer en examinant la **variance** interne des liens $\omega_{i,j}$. Lorsqu'un sous-ensemble $\mathcal{C} \subseteq \{1, \dots, n\}$ présente des liaisons internes $\{\omega_{i,j} \mid i,j \in \mathcal{C}\}$ dont la variance devient négligeable, on en déduit que la **dynamique** DSL a **stabilisé** ces connexions à des valeurs (quasi) constantes. Les paragraphes A, B, C et D (9.5.1.2) décrivent successivement le principe, la mise en œuvre mathématique, la logique d'interprétation, puis concluent.

A. Motivation : Pourquoi la variance ?

Un **SCN** (chap. 9.4) évolue sous la règle DSL (ou ses variantes) :

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

Un "cluster" $\mathcal{C}$ s'interprète comme un **sous-ensemble** d'entités fortement reliées (somme élevée des poids ω), mais aussi **homogènes**, où les liens internes $\omega_{i,j}$ varient peu au fil du temps. Pour qualifier quantitativement cette faible fluctuation, on calcule la **variance** interne au cluster. Un lien $\omega_{i,j}$ "stable" signifie qu'il oscille peu sur la fenêtre temporelle considérée, suggérant que l'**apprentissage** a convergé localement (pas de renforcement ou d'oubli notable en cours).

B. Mesurer la Variance dans un Cluster

Soit $\mathcal{C}$ un sous-ensemble d'entités et ΔT un horizon d'observation (ex. 100 itérations). On relève $\{\omega_{i,j}(t), \dots, \omega_{i,j}(t+\Delta T)\}$ pour $(i,j) \in \mathcal{C}^2$. Pour chaque paire (i,j), on définit :

$$\bar{\omega}_{i,j} = \frac{1}{\Delta T} \sum_{\tau=0}^{\Delta T-1} \omega_{i,j}\,(t+\tau),$$

et

$$\sigma_{i,j}^2(\Delta T;\, t) = \frac{1}{\Delta T} \sum_{\tau=0}^{\Delta T-1} (\,\omega_{i,j}(t+\tau) - \bar{\omega}_{i,j})^2.$$

Si la *max* ou la *moyenne* de $\sigma_{i,j}^2$ sur $(i,j) \in \mathcal{C}$ est **inférieure** à un seuil ϵ^2, on juge la variance interne faible. Cela reflète une **faible dispersion** des liaisons internes, signe que $\mathcal{C}$ est **cohérent** et **peu changeant**. On peut aussi condenser cette analyse en calculant la variance d'une **grandeur globale**, comme la somme

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

Une variance $\mathrm{Var}\big(\Omega(\mathcal{C},\cdot)\big)$ faible sur la fenêtre $[t,\, t+\Delta T]$ indique que le cluster conserve sa cohésion au cours de ΔT.

C. Interprétation : Cluster stable

Quand la variance interne tombe sous ϵ^2 et n'évolue plus, on conclut que $\mathcal{C}$ a **stoppé** sa réorganisation, au moins pour l'intervalle observé. Cela s'apparente à une **mini-convergence** locale, où les liaisons internes $\{\omega_{i,j} \mid i,j \in \mathcal{C}\}$ atteignent un "état fixe" relatif. Sur le plan **dynamique**, il n'y a plus de renforcement ni d'oubli substantiel en cours. Il est alors possible de :

- Déclarer ce cluster "validé" (dans un cadre d'application pratique),
- Réduire la fréquence de mise à jour DSL pour ce groupe, ou
- Exploiter l'existence de ce bloc stable pour d'autres finalités (visualisation, hiérarchie).

D. Conclusion (9.5.1.2)

La variance des liaisons $\omega_{i,j}$ constitue un **indicateur** essentiel pour détecter la stabilisation de **clusters** au sein d'un SCN. En mesurant sur une fenêtre temporelle ΔT :

- **Faible variance** $\sigma_{i,j}^2 < \epsilon^2 \Rightarrow$ la pondération $\omega_{i,j}$ ne fluctue pas,
- **Homogénéité** du sous-ensemble $\mathcal{C} \Rightarrow$ toutes les liaisons internes restent proches d'une valeur moyenne stable,

on repère les **clusters** qui ont cessé d'évoluer (au moins temporairement). Dans un **DSL** non figé (chap. 9.4.2), la recherche de variance faible aide à repérer les "périodes" de stabilité locale dans un réseau en flux continu. Aucun cluster n'est éternellement figé si l'environnement évolue, mais l'approche variance demeure un outil précieux pour qualifier les "moments" ou "zones" du SCN atteignant un **quasi-état stationnaire**.

9.5.2. Oscillations ou Fluctuations sans Fin

Dans un contexte d'**évolution temps réel** (chapitre 9) où de nouvelles données, entités ou modifications arrivent de façon continue, il est possible que le **Synergistic Connection Network (SCN)** ne parvienne jamais à atteindre un **état** stationnaire. Au lieu de cela, on observe des **fluctuations** ou **oscillations** perpétuelles dans les pondérations $\{\omega_{i,j}\}$. Cette section (9.5.2) s'intéresse précisément à ce **risque**, en soulignant que si le **flux** de données est trop variable ou trop "bruyant", le SCN peut rester en mouvement constant, sans jamais "se poser".

9.5.2.1. Si le flux de données est trop variable, le SCN peut rester dans un état de fluctuation permanente

Même si un **Synergistic Connection Network** (SCN) en **apprentissage continu** est conçu pour s'adapter à des modifications progressives, il peut arriver que le flux de données ou de signaux soit **excessivement** changeant, empêchant toute stabilisation locale. Les paragraphes ci-après décrivent, sur le plan **mathématique**, comment la règle de mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$ peut se retrouver piégée dans des oscillations ou fluctuations persistantes, et quelles en sont les conséquences pour l'organisation du SCN.

A. Principe d'Adaptation Infinie

Un **SCN** classique suppose qu'après un certain temps de mise à jour :

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

les liaisons $\omega_{i,j}$ convergent vers un arrangement stable ou quasi stable, dès lors que le flux $\{S_t(i,j)\}$ devient relativement cohérent ou stationnaire. Cependant, si $S_t(i,j)$ demeure **fortement** variable à chaque itération t (données contradictoires, aléas extrêmes, changements de contexte non prévisibles), la mise à jour (1) s'effectue en vain, chaque adaptation locale est aussitôt contrée par une nouvelle perturbation.

Sur le plan **mathématique**, on aboutit à un **processus** $\omega(t)$ ne se fixant pas, voire oscillant à grande amplitude, puisqu'aucun compromis n'est atteint pour l'ensemble des liaisons $\omega_{i,j}$. On perd la notion de clusters stables, remplacés par un **mouvement** permanent dans l'espace des paramètres.

B. Oscillations dans $\omega_{i,j}(t)$

Lorsque l'**excitateur** $S_t(i,j)$ varie trop, par exemple alternant des valeurs positives et négatives, la règle (1) peut conduire $\omega_{i,j}(t)$ à **monter** quand $S_t(i,j)$ est élevé, puis **descendre** quand $S_{t+1}(i,j)$ est faible ou négatif, etc. En l'absence d'un amortissement suffisant (grand τ, ou mécanismes d'inhibition capables de forcer un tri net entre liens forts et faibles), l'état $\omega_{i,j}(t)$ ne cesse de rebondir.

Exemple mathématique : signaux contradictoires

On considère un scénario "périodique" :

$$S_t(i,j) = \begin{cases} +1, & \text{si } t \text{ est pair,} \\ -1, & \text{si } t \text{ est impair.} \end{cases}$$

On applique $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S_t(i,j) - \tau\,\omega_{i,j}(t)\big]$. Si τ et η ne sont pas ajustés, $\omega_{i,j}$ risque de "monter" pour les itérations paires (où $S_t(i,j) = +1$) puis "descendre" pour les itérations impaires (où $S_t(i,j) = -1$), entraînant une oscillation. D'autres exemples incluent un **bruit blanc** $S_t(i,j)$ de moyenne nulle, qui peut maintenir $\omega_{i,j} \approx 0$ mais en fluctuation incessante.

C. Conséquences sur la Dynamique

Lorsque le flux est **trop variable**, le SCN n'**atteint** plus de configuration stable de $\{\omega\}$. Les **clusters** censés se former se défont aussitôt, conduisant à une **oscillation** ou un pseudo-chaos :

- **Pas de clusters durables** : la logique de stabilisation est continuellement contrecarrée,
- **Incohérence** en pratique : le système ne parvient pas à proposer des regroupements pertinents,
- **Stress computationnel** : l'actualisation de ω se poursuit à l'infini, sans zones de repos.

Mesures de la fluctuation globale

On peut définir un **déplacement** global entre deux instants :

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

Si $\Delta(t)$ reste **important** (et ne décroit pas sur le long terme), on voit que $\omega(t)$ oscille ou se balade dans l'espace paramétrique sans stabilisation. Parfois, les **mécanismes** (inhibition adaptative, bruits amortis, forts τ) permettent de limiter les oscillations, mais un flux extrême peut surpasser ces amortissements.

D. Conclusion (9.5.2.1)

Si les **données** ou la **distribution** $\{S_t(i,j)\}$ changent trop rapidement et de manière contradictoire, un **SCN** demeure en **fluctuation** permanente. La règle DSL :

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

ne stabilise pas ω faute d'un contexte cohérent. On assiste à des **oscillations** ou un pseudo-chaos, conduisant à l'impossibilité de former des **clusters** durables. Selon la vision mathématique, la trajectoire $\{\omega(t)\}$ dans l'espace des pondérations ne converge pas, car le flux injecte continuellement des signaux **incompatibles**. Sans mécanismes d'amortissement extrêmes ou d'adaptation de τ, η en conséquence, le SCN ne parvient pas à cristalliser une configuration.

9.5.2.2. Utiliser les mécanismes (inhibition adaptative, recuit, etc.) pour éviter un chaos continu

Lorsqu'un **Synergistic Connection Network** (SCN) est soumis à un **flux** de données potentiellement très variable, il peut se retrouver en perpétuel mouvement (voir [9.5.2.1] sur l'état de fluctuation). Pour amortir ou endiguer ce **chaos** et permettre malgré tout la formation

de *clusters* stables, la logique **DSL** (Deep Synergy Learning) propose plusieurs **mécanismes** de régulation :

- **Inhibition adaptative**,
- **Recuit simulé** (injection de bruit contrôlé),
- **Saturations et seuils** (*clipping*).

Les sections A, B, C en décrivent la mécanique, puis la conclusion (9.5.2.2) insiste sur l'importance de les combiner pour un **SCN** en flux.

A. Inhibition Adaptative : réguler la compétition

Dans la **règle** DSL standard :

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

on ajoute souvent un **terme** d'inhibition latérale $\gamma \sum_{k\neq j} \omega_{i,k}\,(t)$. Cela se généralise en une **inhibition adaptative**, où la variable $\gamma(t)$ s'ajuste en fonction de l'état global du SCN. Elle augmente lorsqu'un trop grand foisonnement de liaisons moyennes est détecté et diminue si le SCN devient trop épars. L'équation devient :

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

Ce **terme** $-\gamma(t)\sum_{k\neq j}\ \omega_{i,k}(t)$ introduit une **compétition**. Si les poids $\omega_{i,k}$ deviennent trop nombreux, l'inhibition s'intensifie, réduisant les valeurs trop moyennes et empêchant un **chaos** d'oscillations excessif. Sur le plan **mathématique**, $\gamma(t)$ se règle par un "feedback" global. Dès que la densité de liens dépasse un certain seuil, on augmente γ, ce qui incite le réseau à forcer une sélection (cf. chap. 7.4).

B. Recuit Simulé : bruit contrôlé en flux

Le **recuit simulé** (chap. 7.3) ne se limite pas à la phase initiale d'entraînement, il peut être prolongé dans un cadre **incrémental**. Cela revient à introduire un **bruit** $\xi_{i,j}(t)$ modéré à chaque itération, corrélé à une "température" $T(t)$:

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

où $\sigma(t) \sim \sqrt{T(t)}$ mesure l'ampleur de la perturbation. Cela **aide** à éviter les cycles stériles ou la sur-réactivité à un flux variant :

- **Sans bruit**, si $S_t(i,j)$ fluctue fortement, $\omega_{i,j}(t)$ peut suivre des allers-retours.
- **Avec bruit** : la dynamique explore un espace plus large, cassant d'éventuels attracteurs chaotiques, et autorisant de petites corrections quand un lien oscille.

On veille à contrôler $\sigma(t)$ pour ne pas ruiner la formation de clusters stables (la température ne doit pas être trop haute).

C. Saturations / Seuils (Clipping)

Pour éviter que $\omega_{i,j}$ ne prenne des valeurs trop extrêmes et accentue les oscillations, on applique un **clipping** au-dessus de $\omega_{\max}$. De même, on impose parfois un **seuil minimal**. Si $\omega_{i,j}$ tombe sous $\theta_{\min}$, on le fixe à 0 ou quasi nul. Après la mise à jour :

$$\omega_{i,j}(t+1) \leftarrow \min\Big(\omega_{\max}, \max\big(0,\ \omega_{i,j}(t+1)\big)\Big)$$

ou toute variante incorporant $\theta_{\min} > 0$. Cette **opération** annexe se justifie **mathématiquement** car elle borne la grandeur $\omega_{i,j}$. S'il y a un coup de "rebond" dans la synergie $S_t(i,j)$, la pondération ne s'emportera pas trop loin au-dessus de $\omega_{\max}$ (limitant le chaos), et elle sera élaguée sous $\theta_{\min}$ si elle devient insignifiante (limitant les fluctuations dans un magma de petits liens).

Conclusion (9.5.2.2)

Lorsqu'un **flux** engendre de **fortes** fluctuations, mettant en danger la formation de **clusters** dans un SCN, on recourt aux **mécanismes** de stabilisation suivants :

- **Inhibition adaptative** : la concurrence s'ajuste selon la densité de liens,
- **Recuit simulé continu** : injection d'un bruit modéré, cassant les attracteurs chaotiques et évitant un piégeage dans des oscillations trop brutales,
- **Saturations / Seuils** : on borne la croissance de ω par un $\omega_{\max}$ et on coupe ce qui est trop faible, stabilisant la dynamique.

Grâce à ces **termes** et **clippings**, la règle de mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$ ne part pas en **chaos**. On contraint la réactivité du SCN, assurant un **rythme** d'adaptation plus mesuré, essentiel pour la **cohérence** des clusters. Bien calibré, ce dispositif équilibre la *plasticité* (capacité à s'ajuster à de nouveaux flux) et la *stabilité* (non-basculement permanent).

9.5.3. Approche de Fenêtre Glissante

Dans nombre de scénarios d'**apprentissage continu** (cf. chapitre 9), le **flux** de données peut s'étendre sur de longues périodes, avec un environnement évoluant progressivement ou de façon intermittente. Pour maintenir un **équilibre** entre la mémorisation des informations passées et l'adaptation aux nouvelles, une **stratégie** de **"fenêtre glissante"** peut être adoptée. On ne conserve qu'un **lot** de données correspondant à la portion **récente** (taille de fenêtre W) et on **oublie** progressivement celles qui sont trop anciennes.

9.5.3.1. On n'entretient la mémoire que sur les données "récentes" (taille W), en effaçant l'influence trop ancienne

Un **SCN** (Synergistic Connection Network) en **apprentissage continu** peut s'avérer submergé si la base de données accumulée depuis le début de la session devient trop large ou trop obsolète. L'idée d'une **fenêtre glissante** de taille W est de ne considérer que les données les plus **récentes**, en ignorant (ou oubliant) les flux antérieurs à $t - W$. Les paragraphes A, B, C (9.5.3.1) décrivent successivement le principe, les avantages et limites, et la mise en œuvre

mathématique. La conclusion finalise l'articulation de ce mode de gestion de la mémoire dans un **SCN** dynamique.

A. Principe de la Fenêtre Glissante

Dans un flux $\{\mathbf{x}(t)\}_{t\geq 1}$, on considère un **intervalle** $[t-W+1,\dots,t]$ de **longueur** W comme la "fenêtre" active. Pour tout $\tau < t-W+1$, on cesse de prendre en compte $\mathbf{x}(\tau)$. L'approche se lit comme suit :

- Le SCN **calcule** la synergie $S(i,j,t)$ à partir des données situées dans la fourchette $[t-W+1,\dots,t]$ seulement,
- Les pondérations $\omega_{i,j}$ ne retiennent l'influence que de ce **sous-ensemble** de données,
- À l'itération $t+1$, la fenêtre "glisse" et elle devient $[t-W+2,\dots,t+1]$. On ajoute la donnée $\mathbf{x}(t+1)$ et on retire $\mathbf{x}(t-W+1)$.

De la sorte, on pratique un "**effacement**" progressif de tout ce qui est trop ancien, cohérent avec un paradigme **online** ou un environnement changeant.

B. Avantages et Limites

L'un des principaux avantages de cette approche réside dans sa **capacité d'adaptation rapide**. Le SCN se **réoriente** facilement vers les nouvelles tendances en excluant de la fenêtre les contextes obsolètes, ce qui lui permet de rester pertinent face à un flux dynamique. De plus, la **complexité reste maîtrisée** puisqu'à chaque étape, seuls W points sont pris en compte, évitant ainsi l'explosion exponentielle de la mémoire et simplifiant le calcul de synergie $S(i,j,t)$.

Toutefois, cette méthode présente également des limites. L'**oubli du contexte lointain** constitue une contrainte majeure, car dès que t avance, tout ce qui précède $t-W$ disparaît. Si des **patterns** historiques réapparaissent après un long délai, le SCN ne peut plus les exploiter directement. Le **choix de la fenêtre** W devient alors un paramètre délicat à ajuster : une valeur trop faible entraîne une surréaction aux variations locales, tandis qu'une valeur trop élevée réduit la réactivité et stocke une mémoire inutilement étendue.

C. Mise en Œuvre Mathématique : Formules de Synergie et Mise à Jour

On peut définir

$$S_{\text{fenètre}}(i,j,t) = \frac{1}{W}\sum_{\tau=t-W+1}^{t} \kappa\left(\mathbf{x}(\tau),\, i,\, j\right),$$

où $\kappa(\mathbf{x}(\tau), i, j)$ représente la contribution de la donnée $\mathbf{x}(\tau)$ aux entités i, j. La somme est normalisée par W. Lorsque le temps avance de 1, on fait :

$$S_{\text{fenètre}}(i,j,t+1) = S_{\text{fenètre}}(i,j,t) + \frac{1}{W}\,\kappa(\mathbf{x}(t+1),\, i,\, j) - \frac{1}{W}\,\kappa(\mathbf{x}(t-W+1),\, i,\, j).$$

Ceci économise un recalcul complet.

Dans la **règle** DSL, au lieu de recourir à un $S(i,j)$ global, on se sert de $S_{\text{fenètre}}(i,j,t)$:

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

De ce fait, $\omega_{i,j}$ ne reflète que la *mémoire* des W dernières unités de temps, un "effet d'oubli" se manifestant pour $\tau < t - W + 1$.

Conclusion.

La **fenêtre glissante** de taille W sert de moyen pour que le **SCN** n'accumule pas indéfiniment l'historique, et ne se base que sur les données "récentes", abandonnant l'influence plus ancienne. Sur le plan mathématique, cela équivaut à tronquer la **somme** ou la **moyenne** dans la fonction $S_{\text{fenêtre}}$, impliquant une **descente d'énergie** (ou mise à jour DSL) restreinte au sous-ensemble $\{\mathbf{x}(t - W + 1), \dots, \mathbf{x}(t)\}$. On y gagne en **adaptation rapide** et en **complexité** (taille contrôlée de la fenêtre), au prix de perdre la mémoire d'anciens patterns s'ils réapparaissent plus tard. Ce compromis s'inscrit dans la logique du flux (chap. 9), où l'on préfère une plasticité à court/moyen terme plutôt qu'un stockage permanent.

9.5.3.2. Contrôle paramétrique : plus la fenêtre est grande, plus on mémorise l'historique, moins on s'adapte vite

Lorsque l'on conçoit un **Synergistic Connection Network** (SCN) pour un **flux** de données, il est courant de définir une **fenêtre** (ou un horizon) de taille W qui borne la mémoire. Seules les données comprises dans $[t - W + 1, \dots, t]$ influencent les calculs de synergie $S(i, j, t)$ et, partant, la mise à jour $\omega_{i,j}$. Cette section (9.5.3.2) examine comment le paramètre W conditionne la *vitesse* d'adaptation et la *quantité* d'historique retenu. Plus W est grand, plus l'**apprentissage** prend en compte un historique étendu, mais moins il réagit aux nouveautés ; moins W est grand, plus il est réactif mais moins il conserve la trace du passé.

A. Contexte Mathématique : Fenêtre Glissante

On considère que la **synergie** $\tilde{S}(i, j, t)$ est calculée sur la base des données récentes dans $[t - W + 1, \dots, t]$. Par exemple,

$$\tilde{S}(i, j, t) = \frac{1}{W} \sum_{k=0}^{W-1} \Phi\left(\mathbf{x}(t - k),\, i,\, j\right).$$

Cette somme moyenne sur les W derniers instants. Si W est **élevé**, $\tilde{S}(i, j, t)$ devient plus "inertiel", puisant dans un large historique ; si W est **faible**, $\tilde{S}$ reflète surtout l'actualité immédiate. Dans la mise à jour DSL, on injecte :

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

ce qui confère un "effet mémoire" de longueur W sur les pondérations $\{\omega_{i,j}\}$.

B. Équilibre entre Mémoire et Réactivité

Fenêtre Large $\Longrightarrow$ Histoire plus longue

Si W est **grand**, $\tilde{S}(i, j, t)$ incorpore beaucoup de données passées. La variation d'un instant récent n'impacte que faiblement la moyenne (1). Par conséquent, $\omega_{i,j}$ met plus de temps à réagir aux changements soudains. Cela permet un **lissage temporel**, réduisant ainsi les fluctuations erratiques et assurant une certaine stabilité dans l'évolution du réseau. Toutefois, cette inertie peut aussi ralentir la détection de nouvelles tendances ou de ruptures significatives, augmentant ainsi le risque de conserver un historique dépassé qui ne reflète plus la dynamique actuelle.

Fenêtre Petite ⟹ Grande Réactivité

Si W est **petit**, les données anciennes sont rapidement **oubliées**, ce qui rend $\tilde{S}(i,j,t)$ très réactif aux nouveaux événements. La mise à jour (2) devient ainsi plus rapide, favorisant une **adaptation immédiate** aux signaux récents et réduisant la latence. Toutefois, cette forte réactivité s'accompagne d'une **volatilité accrue** : un flux bruité peut provoquer des oscillations non désirées, et toute information antérieure à W est définitivement perdue, ce qui empêche la prise en compte de tendances plus longues.

C. Autres Formes de Paramétrage (Exponentiel vs. Fenêtre)

Plutôt qu'une **fenêtre** glissante "abruptement" coupée, on peut opter pour un **filtrage exponentiel** (ou un "forgetting factor" α) :

$$\tilde{S}(i,j,t) = (1-\alpha)\,\tilde{S}(i,j,t-1) + \alpha\,\Phi(\mathbf{x}(t),\,i,\,j).$$

Ici, $\alpha \in (0,1]$ joue un rôle similaire à $1/W$. Plus α est petit, plus la mémoire est longue ; plus α est grand, plus on oublie vite. Sur le plan **mathématique**, un filtre exponentiel est équivalent à une fenêtre "infinie" mais avec un poids décroissant au fil du temps, tandis qu'une fenêtre glissante "coupe" net les données plus anciennes que $t-W$. Les deux approches incarnent la même **trade-off** : *inertie* vs. *rapidité.*

D. Conséquences sur la Dynamique du SCN

La **convergence** et la **plasticité** sont directement influencées par la taille de W. Un **grand** W (ou un petit α) permet au SCN de mieux conserver l'historique de la distribution et d'éviter les oscillations excessives (cf. chap. 9.5.2). Toutefois, en cas de **changement brutal** du flux, le réseau met plus de temps à se reconfigurer. À l'inverse, un W **court** rend le SCN très réactif, répondant immédiatement aux variations récentes. Cela peut être avantageux dans un environnement à forte variabilité, mais risque d'entraîner des réorganisations incessantes si le flux est trop instable.

Sur le plan **algorithmique**, l'implémentation repose sur le stockage des W derniers items ou sur une gestion par **somme glissante**, mise à jour en $O(1)$ à chaque itération en ajoutant $\Phi\big(\mathbf{x}(t)\big)$ et en retirant $\Phi\big(\mathbf{x}(t-W)\big)$. Cette approche permet de limiter la charge de calcul sans compromettre l'efficacité du modèle.

Dans des **SCN multi-échelle** (cf. chap. 6), il est possible d'associer plusieurs horizons temporels. Un premier paramètre W_1 assure une réactivité immédiate au niveau micro, tandis qu'un second horizon $W_2 > W_1$ capture des tendances plus longues. Cette combinaison permet au réseau de concilier adaptation rapide et conservation d'une mémoire plus durable.

Conclusion

Plus la fenêtre est grande, plus on mémorise l'historique, moins on s'adapte vite. Cette maxime synthétise le **trade-off** au cœur d'un SCN disposant d'un **horizon** glissant. Sur le plan **mathématique**, élargir W accroît l'inertie de la synergie $\tilde{S}(i,j,t)$, tandis que réduire W accentue la sensibilité aux signaux récents. Selon la **nature** du flux (stable vs. volatile), on module W (ou un facteur d'oubli exponentiel α) pour trouver un compromis entre la **cohérence historique** et la **réactivité**. Cette configuration influe fortement sur la dynamique, la *convergence*, et la stabilité (ou l'instabilité) de l'apprentissage incrémental au sein du SCN.

9.5.3.3. Comparaison “fenêtre glissante” vs. “forgetting factor”

Dans l’**apprentissage continu** ou l’**adaptation en flux** pour un **SCN** (Synergistic Connection Network), il est souvent indispensable de ne pas prendre en compte l’intégralité de l’historique depuis le début, mais de diminuer l’influence des données anciennes. Deux méthodes couramment adoptées sont la **fenêtre glissante** (sliding window) et le **facteur d’oubli** (forgetting factor). Les parties A et B (9.5.3.3) présentent les principes de chaque méthode, puis la section C discute de leurs différences et implications pour un **DSL** temps réel, avant la conclusion.

A. Fenêtre Glissante

La **fenêtre glissante** de taille W consiste à ne garder que les données ou événements compris dans l’intervalle $[t - W + 1, \dots, t]$. À chaque nouveau pas de temps $t + 1$, on ajoute la donnée $\mathbf{x}(t + 1)$ et on retire la plus ancienne $\mathbf{x}(t - W + 1)$. Ainsi, on a toujours un “stock” de longueur fixe W de données considérées comme “actives”.

Formellement, si la synergie $\tilde{S}(i, j, t)$ est calculée sur la base d’une somme ou moyenne des observations passées $\{\mathbf{x}(\tau)\}$ pour $\tau \in [t - W + 1, \dots, t]$, on peut écrire :

$$\tilde{S}_{\text{fenètre}}(i, j, t) = \sum_{\tau=\max(0,t-W+1)}^{t} \kappa(\mathbf{x}(\tau),\, i,\, j),$$

où κ représente la contribution de chaque donnée $\mathbf{x}(\tau)$. Lors de la mise à jour DSL, on s’en sert par exemple dans :

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

L’un des principaux **avantages** de cette approche est le contrôle explicite de la **taille** W. Toute donnée antérieure à $t - W$ est totalement ignorée, empêchant l’explosion de la mémoire et simplifiant la gestion du **concept-drift**. Cette propriété assure que seules les informations récentes influencent l’évolution du réseau.

Cependant, cette méthode présente aussi un **inconvénient** majeur. Lorsqu’un point sort de la fenêtre, son influence passe brutalement de 1 à 0, ce qui peut engendrer des discontinuités et provoquer des **sauts** dans la dynamique du SCN si W est mal calibré.

B. Facteur d’Oubli (Forgetting Factor)

Une autre approche est de pondérer chaque observation selon le temps qui s’est écoulé, via un **facteur** $\lambda \in (0,1)$. Au lieu de couper à un horizon W, on diminue l’influence d’une donnée ancienne “progressivement” selon $\lambda^{t-\tau}$. Mathématiquement, on définit :

$$\tilde{S}_{\text{forget}}(i, j, t) = \sum_{\tau=1}^{t} \lambda^{t-\tau}\, \kappa(\mathbf{x}(\tau),\, i,\, j).$$

Une donnée ancienne $\mathbf{x}(\tau)$ pour $\tau \ll t$ reste prise en compte, mais son **poids** se réduit exponentiellement au fil des pas.

L'un des principaux **avantages** de cette approche est qu'elle assure une **transition lisse**, sans rupture brutale lorsque les données deviennent trop anciennes. Contrairement à une fenêtre fixe, elle évite les discontinuités et garantit une évolution progressive des pondérations.

Cependant, un **inconvénient** notable réside dans la persistance des signaux anciens. Même s'ils perdent en influence, ils continuent de contribuer aux calculs, ce qui peut ralentir la réaction du SCN face à un **changement** brutal. Si $\lambda \approx 1$, la mémoire est trop conservatrice, risquant d'induire une inertie excessive. De plus, aucun *hard reset* naturel n'existe, sauf si λ est modifié soudainement.

C. Comparaison et Usage en DSL Temps Réel

La **fenêtre glissante** garantit une mémoire de taille **constante** W. Tant qu'une donnée est dans la fenêtre, sa contribution est pleine (valeur 1) et elle disparaît immédiatement lorsqu'elle en sort (valeur 0). Cette approche est simple à interpréter et permet un contrôle direct du stock courant, mais elle introduit des ruptures abruptes.

Le **facteur d'oubli**, en revanche, permet une mémoire **potentiellement illimitée**, chaque donnée étant pondérée par un coefficient $\lambda^{t-\tau}$. Cette méthode offre une transition plus progressive et évite les discontinuités, mais elle risque de conserver des informations obsolètes si λ est trop élevé.

La différence entre les deux approches peut être illustrée en comparant leur impact sur le calcul de la **synergie** $\tilde{S}(i,j,t)$.

Dans le cas d'une **fenêtre fixe** de taille $W = 100$:

$$\tilde{S}_{\text{fenètre}}(i,j,t) = \sum_{\tau=t-99}^{t} \kappa\left(\mathbf{x}(\tau), i, j\right).$$

À chaque nouvelle itération, on ajoute $\kappa\big(\mathbf{x}(t+1)\big)$ et on retire $\kappa\big(\mathbf{x}(t-99)\big)$, ce qui provoque une transition nette des contributions.

Avec un **facteur d'oubli** $\lambda = 0.98$:

$$\tilde{S}_{\text{forget}}(i,j,t+1) = \lambda\, \tilde{S}_{\text{forget}}(i,j,t) + (1-\lambda)\, \kappa\big(\mathbf{x}(t+1)\big).$$

Ici, aucune donnée n'est supprimée brutalement, son influence décroît de manière exponentielle avec $\lambda^{t-\tau}$.

Dans un **SCN** en **apprentissage continu** (chap. 9), les **entités** $\{\mathcal{E}_i\}$ et leurs liaisons $\omega_{i,j}$ évoluent selon la règle de mise à jour :

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

Ici, $\tilde{S}_{\text{eff}}$ peut être soit $\tilde{S}_{\text{fenètre}}$ (avec une fenêtre de taille W), soit $\tilde{S}_{\text{forget}}$ (avec un facteur d'oubli λ).

Le **choix** entre une **fenêtre glissante** et un **facteur d'oubli** dépend de la **variabilité** et de la **taille** du flux. Une fenêtre fixe assure une réactivité immédiate et permet des "resets" complets, tandis qu'un facteur d'oubli favorise une adaptation plus douce et continue.

Conclusion (9.5.3.3)

La comparaison **“fenêtre glissante” vs. “facteur d’oubli”** illustre deux façons de **pondérer** différemment les données anciennes dans un contexte de flux continu. Les points essentiels sont :

- **Fenêtre Glissante** :
    - Force un *hard cutoff* des données plus vieilles que W,
    - Meilleure en cas de changements rapides, un “effacement” clair,
    - Mais peut provoquer des discontinuités au moment de la sortie d’un point de la fenêtre.
- **Forgetting Factor** :
    - Poids **exponentiellement** décroissant, aucune coupure abrupte,
    - Peut conserver la trace d’anciens signaux malgré un concept-drift, si $\lambda \approx 1$,
    - Plus *doux*, mais exige un réglage prudent de λ.

Le **DSL** temps réel peut employer l’un ou l’autre (ou un **hybride**) en fonction du **niveau** de mémoire à retenir (court vs. long terme) et du **contexte** (stabilité du flux, nécessité d’effacement brutal, etc.).

9.6. Apprentissage Continu et Catastrophic Forgetting

Dans de nombreux scénarios d'**apprentissage continu** (ou **online learning**), il est essentiel que le système (Deep Synergy Learning, DSL, ou tout autre) soit capable d'**accumuler** de l'expérience sur des séquences de tâches (ou de données) sans pour autant **anéantir** les compétences acquises précédemment. L'une des **difficultés** majeures rencontrées en apprentissage continu est le **catastrophic forgetting**, où l'ajout de nouvelles informations **efface** (partiellement ou totalement) la performance sur les **tâches** ou **données** antérieures.

Dans la section 9.6, nous abordons ce phénomène de "Catastrophic Forgetting" et explorons comment le **DSL** peut y faire face — ou en être victime — dans le cadre d'un **SCN** (Synergistic Connection Network) évolutif. Nous commençons (9.6.1) par définir ce concept, puis (9.6.2) nous verrons différentes **stratégies** pour éviter l'oubli brutal, avant de conclure (9.6.3) par des **exemples** pratiques.

9.6.1. Définition du "Catastrophic Forgetting"

9.6.1.1. Concept venant du NN : un réseau entraîné sur une tâche A, puis sur B, perd toute performance sur A

Le phénomène de l'**oublie catastrophique** (ou *catastrophic forgetting*) est né de l'étude des **réseaux neuronaux** successivement entraînés sur plusieurs **tâches** distinctes. Dans un cadre classique, on optimise d'abord les **poids** θ d'un réseau pour **minimiser** une fonction de coût $\mathcal{L}_A(\theta)$ associée à la tâche A, puis on réentraîne ces mêmes poids pour **minimiser** $\mathcal{L}_B(\theta)$ relative à la nouvelle tâche B. La mise à jour des poids, souvent guidée par une descente de gradient $\nabla_\theta \mathcal{L}_B(\theta)$, détruit alors la configuration qui minimisait $\mathcal{L}_A$. Cette interférence aboutit à une **chute brutale** de la **performance** sur la tâche A, d'où le qualificatif de *catastrophique*.

Dans le **Deep Synergy Learning (DSL)**, l'architecture ne suit pas exactement celle d'un réseau **feed-forward**, mais repose sur un **Synergistic Connection Network (SCN)**. Il s'agit d'un graphe pondéré $\{\omega_{i,j}\}$ dont les **pondérations** évoluent en fonction d'une **fonction de synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$. Toutefois, l'idée d'une **auto-organisation** progressive des connexions y est similaire. Lorsque le SCN est successivement exposé à de **nouvelles entités** ou à un **nouveau flux** de données (typiquement associé à une nouvelle tâche B), on modifie les liaisons $\omega_{i,j}$ pour **favoriser** la formation de **clusters** et de connexions répondant à ce flux. Si rien n'est prévu pour maintenir les anciennes structures, les liens plus anciens peuvent se **délier** ou s'affaiblir.

La logique de mise à jour peut s'écrire, dans une version simple, par la formule :

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

où η désigne un **taux d'apprentissage**, τ un **coefficient** de décroissance et $S(i,j)$ la **synergie**. Lorsqu'une **nouvelle tâche** B injecte un volume important de données ou d'entités inédites, ces **nouvelles synergies** tendent à dominer la mise à jour. Les **clusters** et les liaisons qui étaient stables pour l'ancienne tâche A voient leurs pondérations $\omega_{i,j}$ se **décroître** (car le réseau réalloue ses liens là où la **synergie** est la plus grande). En l'absence de **mécanisme** dédié pour préserver la structure acquise, la conséquence est un effondrement soudain du cluster de la tâche A, reflétant un véritable *oubli*.

Le caractère “catastrophique” s’observe lorsqu’une réorganisation rapide et forte efface presque entièrement l’**information** antérieure. Dans un **exemple minimal**, on peut imaginer un SCN où un **macro-cluster** $\mathcal{C}_A$ est associé à une classe ou à un ensemble d’entités A. Si, par la suite, on introduit une série de nouvelles entités “B” fortement corrélées entre elles, le **poids** de leurs liens s’amplifie très vite pour constituer un cluster B dominant. La synergie des entités A devient alors relative, conduisant à une **désactivation** ou à une **mise à zéro** des anciennes connexions, spécialement si l’on applique une règle de parsimonie (coupure sous un certain seuil $\omega_{\min}$). On se retrouve ainsi à “oublier” la configuration de départ, ce qui correspond à une **perte soudaine de performance** pour gérer ou distinguer les entités A.

Il existe une **analogie directe** avec le phénomène classique observé dans les **réseaux neuronaux**. Lorsqu’on passe d’une **tâche A** à une **tâche B**, la nouvelle optimisation détruit la solution précédemment adaptée à A. Dans le DSL, l’équivalent se produit sur la **matrice** ω dont les éléments changent d’équilibre pour s’aligner sur les **nouvelles** corrélations. Cette **spécificité** d’un SCN, fondée sur une *dynamique d’auto-organisation*, ne doit pas occulter la réalité d’un même problème de **concurrence des apprentissages**. La question essentielle est alors de savoir comment garantir que l’introduction de nouvelles entités ou de nouvelles tâches ne **détruise** pas l’organisation déjà en place, mais qu’elle la complète ou la stabilise. Les chapitres traitant du **lifelong learning** ou de la **stabilisation** des clusters dans le DSL explorent précisément cette problématique, en proposant diverses stratégies pour protéger les liens anciens (p. ex. par des mécanismes d’**inhibition compétitive** ou de **consolidation** des clusters historiques).

En définitive, le **catastrophic forgetting** se traduit, dans un **Synergistic Connection Network**, par une **dynamique** où l’arrivée de nouvelles données oriente les pondérations $\omega_{i,j}$ vers de nouveaux pôles de synergie. Sans **mécanisme** de préservation ou de régulation, les liens antérieurs s’affaiblissent jusqu’à l’oubli quasi total. C’est la raison pour laquelle un **DSL** appliqué sur plusieurs sources ou plusieurs classes de données doit intégrer des règles (stratégiques ou heuristiques) permettant d’éviter le phénomène de “catastrophe” et de maintenir une forme de **plasticité-stabilité** dans le réseau.

9.6.1.2. Dans le DSL, risque de supprimer ou affaiblir des clusters anciens si les nouvelles données dominent

Lorsqu’un **réseau DSL** (*Deep Synergy Learning*) est employé en mode **continu**, il se trouve exposé à un flux successif de **nouvelles données** $\{\mathcal{E}_{\text{new}}\}$. Cette arrivée permanente d’entités inédites peut, dans certains cas, **déstabiliser** ou **diluer** les **clusters** formés à partir d’entités plus anciennes. Au sein du **Synergistic Connection Network (SCN)**, la mise à jour des pondérations $\omega_{i,j}$ peut se réorienter très fortement en faveur de ces nouveautés, conduisant ainsi à un **affaiblissement** rapide des liens historiques et, parfois, à la **disparition** des clusters qu’ils structuraient.

La présente section (9.6.1.2) décrit **pourquoi** ce phénomène se produit, **comment** il se manifeste mathématiquement, et **quels** mécanismes peuvent être proposés pour atténuer cette forme de “submersion” du réseau par les entités les plus récentes.

A. Mécanisme d’Affaiblissement des Clusters Anciens

Lorsque de nouvelles données affluent, la **règle de mise à jour** des poids $\omega_{i,j}$ favorise leur intégration dans le réseau, ce qui implique souvent une forme de **désinvestissement** à l’égard des clusters plus anciens. Cette dynamique repose sur une équation générale de type :

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

où la **fonction de synergie** $S(i,j)$ indique à quel point les entités i et j sont mutuellement utiles ou fortement corrélées, et où $\Delta_{\text{noise ou inhibition}}(\dots)$ peut représenter divers termes de compétition ou de bruit.

Lorsque le réseau reçoit une **nouvelle** classe d'entités $\{\mathcal{E}_{\text{new}}\}$, ces dernières peuvent présenter entre elles une synergie **élevée**. Le DSL met alors à jour les pondérations en faveur de ces liaisons, que l'on peut voir comme une **extension** ou un **nouveau** cluster se formant autour des entités récentes.

Si, en parallèle, la synergie entre entités anciennes $\{\mathcal{E}_{\text{old}}\}$ ou entre anciennes et nouvelles s'avère **plus faible**, ou si un **mécanisme** d'inhibition latérale est à l'œuvre, les **liens** $\omega_{\text{old,old}}$ peuvent décliner. D'un point de vue mathématique, certains modèles imposent un **budget limité** de connections fortes pour chaque entité. Lorsqu'un nœud i renforce de nouvelles connexions $\omega_{i,\text{new}}$, les **liaisons** $\omega_{i,\text{old}}$ se réduisent :

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

ce **terme** $\gamma \sum_{k \neq j} \omega_{i,k}\,(t)$ jouant le rôle d'une **inhibition** qui rééquilibre la somme des poids autour d'une valeur maximale. Ainsi, les **clusters** formés à partir des anciennes entités s'étiolent, faute de soutiens suffisants.

B. Surdominance des Nouveaux Clusters

Dans un flux **incrémental**, si les entités récemment introduites possèdent une synergie **très forte** entre elles, elles forment un **macro-cluster** dominé par ces nouvelles données. Le **réseau** alloue alors la majeure partie des pondérations positives aux connexions internes à ce cluster. Vu sous l'angle biologique, on peut y voir un phénomène où les "événements récents" monopolisent la plasticité des synapses, entraînant la **perte** ou la **désactivation** d'associations plus anciennes.

En l'absence de renforcement continu ou de données "de rappel" pour les entités plus anciennes, la **décroissance** $\tau\,\omega_{i,j}(t)$ finit par gommer leurs liens historiques. De façon opérationnelle, un cluster "historique" se **dissout** dès lors qu'il ne reçoit plus aucune synergie active. Dans un tel scénario, le **SCN** se reconfigure autour de la **nouveauté**, aboutissant à une **instabilité** persistante de toutes les configurations jugées moins pertinentes à l'instant présent.

C. Méthodes pour Éviter la Suppression Indue

Plusieurs approches peuvent limiter la "disparition" rapide des clusters anciens et préserver la **mémoire** des configurations antérieures :

1. Consolidation ou Mémorisation Long Terme

Une première voie consiste à imposer un **plancher** pour les poids jugés essentiels. On peut, par exemple, fixer

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

de sorte que tant qu'un lien a dépassé un certain **seuil** $\omega_{\min}$ et s'est stabilisé au sein d'un cluster, il ne sera pas automatiquement ramené à zéro par une seule vague de nouvelles données.

2. Stratégies de "Verrouillage" de Clusters

Il est également possible, passé un certain **degré de stabilité** pour un cluster, de "verrouiller" partiellement ses connexions internes. Dans ce cas, on limite la **décroissance** pour les poids $\omega_{i,j}$ au sein de ce cluster. On parle souvent de *gradual freeze*, où seuls les liens externes au cluster restent totalement plastiques, tandis que ceux internes sont protégés.

3. Renforcement Multi-Temporal ou Multi-Échelle

Une autre méthode consiste à introduire plusieurs **niveaux** de représentation. Si le SCN opère à la fois sur un **niveau micro** (entités individuelles) et un **niveau macro** (super-nœuds représentant des groupes déjà validés), alors le cluster ancien peut survivre sous forme d'un super-nœud "vétéran" même si, localement, les entités micro changent de connections. Cela permet de **conserver** la mémoire de regroupements antérieurs, tout en assurant la plasticité sur le niveau plus fin.

Conclusion

Dans un **DSL** soumis à un flux continu de données, les **clusters** acquis antérieurement sont exposés au risque de **suppression** ou de **fort affaiblissement** dès lors que les nouvelles données, jugées plus synergiques, prennent la place dans la dynamique des poids $\omega_{i,j}$. Les liens anciens se voient alors réduits ou dissous, phénomène que l'on peut assimiler à une forme de "*catastrophic forgetting*" dans un cadre non supervisé.

Ce mouvement, parfois **inéluctable** sans mécanisme protecteur, résulte d'une **réallocation** des liaisons vers ce qui paraît le plus pertinent au moment présent. Pour éviter une telle **érosion** automatique, le DSL peut être complété par différentes **stratégies**, telles que l'instauration de seuils minimaux, la consolidation partielle ou encore l'adoption d'organisations **multi-échelle**. L'enjeu fondamental est de **maintenir** un certain équilibre entre la plasticité face aux nouveautés et la stabilité nécessaire pour **conserver** les connaissances — ou, ici, les clusters — accumulées.

9.6.2. Stratégies pour Éviter l'Oubli Brutal

Lorsque l'on pratique un **apprentissage continu** ou un **SCN** (Synergistic Connection Network) évoluant en flux (voir §9.6.1), un **risque** majeur réside dans l'**oubli** rapide — ou "catastrophique" — des connaissances antérieures. À chaque venue de nouvelles entités $\{\mathcal{E}_{\text{new}}\}$, le réseau peut réallouer ses ressources (ses liaisons $\omega_{i,j}$) si violemment qu'il en vient à **gommer** la structure acquise. Les **stratégies** ci-dessous (9.6.2.1 à 9.6.2.3) visent à **conserver** un minimum de mémoire sur la synergie passée, pour maintenir la stabilité des clusters déjà formés.

9.6.2.1. Régularisation : conserver un $\alpha\%$ de la synergie passée comme "ancre"

Un moyen de **freiner** l'oubli brutal, dans un **DSL** (Deep Synergy Learning) soumis à des flux successifs de nouvelles données, consiste à **introduire** une **régularisation** qui **contraint** la pondération $\omega_{i,j}(t+1)$ à ne pas trop s'éloigner de $\omega_{i,j}(t)$. Concrètement, on ajoute un **terme** $\alpha\,\omega_{i,j}(t)$ décrivant la **fraction** (ou le pourcentage) de l'ancienne valeur que l'on souhaite conserver à chaque itération.

Une version simplifiée de l'idée s'écrit :

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

où $\alpha \in (0,1)$ indique la proportion d'**ancrage** que l'on retient de l'ancienne pondération $\omega_{i,j}(t)$.

La partie $\omega_{i,j}(t)+\eta\big[S(i,j)-\tau\,\omega_{i,j}(t)\big]$ traduit la **règle** de plasticité caractéristique d'un **SCN**. La pondération est ajustée en fonction de la **synergie** $S(i,j)$, et de la **décroissance** $\tau\,\omega_{i,j}(t)$.

La pondération ainsi ajustée est **pondérée** par $(1-\alpha)$. Parallèlement, on **ajoute** $\alpha\,\omega_{i,j}(t)$, qui est la part d'**ancrage** gardée "telle quelle".

En pratique, on retrouve une idée comparable à l'ajout d'un **terme** $\beta \parallel \omega_{t+1}-\omega_t \parallel$ dans l'énergie ou la fonction coût, imposant une forme de **régularisation** temporelle.

En **développant** le terme, on obtient :

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

Après simplifications, on identifie la correction effective :

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

Ainsi, la modification $\Delta\omega_{i,j}$ se trouve **amortie** par le facteur $(1-\alpha)$. La partie $\alpha\,\omega_{i,j}(t)$ représente la portion de l'ancienne pondération préservée **intacte**.

La **régularisation** agit comme une **ancre**, dans le sens où les anciennes valeurs $\omega_{i,j}(t)$ ne peuvent plus être remises en cause trop brutalement. Cela contribue à **limiter** l'ampleur de l'oubli ou de la destruction des anciens clusters. Par analogie, on peut voir cela comme un **terme** de pénalisation de la distance entre ω_{t+1} et ω_t, garantissant qu'on ne modifie pas trop rapidement la structure pondérée du SCN.

Le choix du paramètre α influe directement sur la dynamique du SCN. Lorsqu'il est **proche de zéro**, l'ancrage est quasi inexistant et la règle DSL reste "pure" :

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

Dans ce cas, le réseau peut **oublier** des clusters anciens très rapidement.

À l'inverse, si α est **proche de 1**, la réécriture des pondérations devient quasi inexistante :

$$\omega_{i,j}(t+1) \approx \omega_{i,j}(t),$$

ce qui **fige** le SCN et empêche l'intégration de nouvelles informations.

Dans la plupart des cas, on privilégie des **valeurs intermédiaires**, typiquement dans l'intervalle $\alpha \in [0.01,0.2]$, afin de préserver **une fraction** des valeurs passées tout en laissant l'algorithme s'adapter aux **nouvelles données**.

L'un des principaux **avantages** de ce paramétrage est qu'il protège la **mémoire** existante, évitant des modifications brutales des liaisons anciennes. Il permet aussi de limiter l'ampleur de la **plasticité** à chaque itération, assurant une stabilité accrue du réseau.

Cependant, une **limite** importante apparaît lorsque α est trop grand : le **réseau** devient presque insensible aux nouvelles synergies et risque de **stagner**, ne se réorganisant plus correctement.

Il est donc crucial de régler α en fonction du **taux de renouvellement** du flux de données et de l'importance accordée à la **stabilité** par rapport à la **flexibilité**.

Exemple Numérique

Soit $\alpha = 0.1$, $\eta = 0.05$, $\tau = 0.2$, et une synergie $S(i,j) = 0.8$. Si la valeur actuelle est $\omega_{i,j}(t) = 0.3$, on calcule :

$$\omega_{i,j}(t+1) = 0.9 \times \big(0.3 + 0.05(0.8 - 0.2 \times 0.3)\big) + 0.1 \times 0.3.$$

Terme DSL :

$$0.3 + 0.05(0.8 - 0.06) = 0.3 + 0.05 \times 0.74 = 0.3 + 0.037 = 0.337.$$

Multiplier par $(1-\alpha) = 0.9$ donne 0.3033.

Ajout de l'ancre $\alpha\,\omega_{i,j}(t) = 0.1 \times 0.3 = 0.03$.

$$\omega_{i,j}(t+1) = 0.3033 + 0.03 = 0.3333.$$

Ainsi, plutôt que de monter directement autour de 0.337, on atterrit un peu **en dessous** grâce à la pondération $(1-\alpha)$. On **force** donc une évolution plus **douce** entre deux itérations.

Conclusion

La **régularisation** où l'on conserve un **pourcentage** $\alpha\%$ de la pondération passée constitue un **mécanisme** simple mais efficace pour **limiter** l'oubli catastrophique dans un **SCN**. Elle apporte un **équilibre** entre :

- **Plasticité** : on continue de tenir compte des nouveautés grâce au terme $(1-\alpha)\,\eta\,[S(i,j) - \tau\,\omega_{i,j}(t)]$.
- **Stabilité** : on retient **systématiquement** la fraction $\alpha\,\omega_{i,j}(t)$ de l'ancienne configuration, ce qui évite un réajustement trop violent des liens et maintient un **socle** de mémoire.

Le paramètre α devient alors **clé**. Une valeur élevée accentue la **stabilité**, tandis qu'une valeur plus faible permet au réseau de se **réorganiser** plus librement.

9.6.2.2. Mémoire structurelle : stockage de "clusters stables" pour les préserver de l'extinction

Dans un **SCN** (*Synergistic Connection Network*) sujet à une **évolution continue**, les **clusters** qui se forment à un instant donné risquent de s'éroder ou de disparaître dès lors que de nouvelles entités affluent et que la **synergie** $S(i,j)$ se réorganise. Pour **préserver** certains groupes particulièrement **cohérents** ou **utiles**, on peut mettre en place une **mémoire structurelle**, c'est-à-dire un **système de stockage** visant à **bloquer** ou **figer** partiellement les liens internes de certains clusters dits "stables". Cette section (9.6.2.2) décrit le **principe**, l'**approche mathématique** et les **avantages** / **limites** d'une telle stratégie.

A. Motivation et Contexte

Les **mises à jour** habituelles d'un DSL, en particulier lorsqu'on active des mécanismes de **parsimonie** ou d'**inhibition** entre liaisons, peuvent entraîner la **dissolution** de clusters antérieurement formés. Un nouveau **flux** de données ou une synergie S réévaluée peut "éclipser" un regroupement pourtant jugé **pertinent** dans le passé.

Lorsqu'un cluster $\mathcal{C} \subset \{\mathcal{E}_i\}$ se révèle important (pour une tâche, un contexte, ou une cohésion forte), il n'est pas toujours souhaitable qu'il se **dilue**. Un simple affaiblissement de $\omega_{i,j}$ peut compromettre l'exploitation de ce cluster à l'avenir.

Par exemple, dans un **SCN** appliqué à la classification, si un cluster correspond à une "catégorie" stable, on aimerait pouvoir **stabiliser** les liaisons internes, même si de nouveaux exemples (d'autres classes) arrivent.

L'idée d'une **mémoire structurelle** est donc de **stocker** la configuration interne d'un cluster (les entités qui le composent et leurs pondérations $\omega_{i,j}$) dès que celui-ci est **considéré** stable. On peut alors :

- **Geler** partiellement les pondérations internes pour qu'elles ne chutent pas sous l'effet des nouvelles synergies.
- **Réactiver** le cluster plus tard si une situation similaire se reproduit.

B. Approche Mathématique : "Verrouillage partiel" de pondérations

Pour empêcher la disparition brutale d'un cluster stable, on peut mettre en œuvre un **"verrouillage"** (lock) ou un **plancher minimal** de ω. Concrètement, lorsqu'un sous-groupe $\mathcal{C}$ est **enregistré**, on restreint la dynamique de décroissance habituelle des pondérations $\omega_{i,j}$ à l'intérieur de $\mathcal{C}$.

On **détecte** qu'un cluster $\mathcal{C}$ est stable (par exemple, quand :

$$\sum_{(i,j)\in\mathcal{C}} \omega_{i,j} \;>\; \Theta_{\text{stab}} \quad \text{ou} \quad \text{modularité}(\mathcal{C}) \;>\; \beta_{\text{limite}},$$

pour certains seuils Θ_{stab} ou β_{limite}). On mémorise alors la "photographie" de ce cluster, c'est-à-dire l'ensemble des entités $\mathcal{C}$ ainsi que leurs poids internes $\omega_{i,j}$.

Au cours des itérations suivantes, on **interdit** à $\omega_{i,j}$ de retomber **en dessous** d'un certain $\omega_{\min} > 0$ s'ils appartiennent à $\mathcal{C}$. Plus généralement, on peut leur octroyer une décroissance réduite. D'un point de vue algorithmique :

$$\omega_{i,j}(t+1) \;=\; \max\big(\omega_{\min},\, \omega_{i,j}(t) + \eta\,\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]\big) \quad \text{pour } (i,j) \in \mathcal{C}.$$

Cet "effet de seuil" ou "hystérésis" empêche les liaisons de **s'effondrer** complètement.

Si, plus tard, des données ou des indicateurs signalent que le cluster $\mathcal{C}$ redevient **utile**, on peut restaurer ou renforcer ses liens. Par exemple, on peut **injecter** à nouveau la configuration mémorisée $\mathbf{\Omega}_{\mathcal{C}}^{(\text{stable})}$, de façon à "réinitialiser" ces liaisons à un niveau élevé au lieu de les réapprendre à partir de zéro.

C. Prévention de l'Extinction : Critères et Sélection

Tout cluster ne mérite pas forcément d'être mémorisé. Il est crucial de définir un **critère** de sélection (seuil sur la somme des poids internes, sur une mesure de densité ou de pertinence) pour éviter d'encombrer la mémoire structurelle. Par exemple :

$$\text{Si } \sum_{(i,j)\in\mathcal{C}} \omega_{i,j} > \Theta_{\text{stab}}, \quad \text{alors “stockage”.}$$

Une **même** entité $\mathcal{E}_i$ peut apparaître dans **plusieurs** clusters stables mémorisés. On peut alors définir des **règles** de gestion de l'overlap. Par exemple, si $\mathcal{E}_i$ appartient à deux sous-ensembles $\mathcal{C}_1$ et $\mathcal{C}_2$, chacun enregistrant $\omega_{i,j}$ différemment, on peut fixer la valeur “en dur” la plus élevée ou faire une moyenne pondérée.

On peut **réévaluer** périodiquement le besoin de conserver un cluster dans la mémoire structurelle. Si l'environnement a changé et que le cluster est devenu incohérent, on peut l'**archiver** ou l'**invalider** pour libérer de la place.

D. Avantages et Limites

L'un des principaux **avantages** de cette approche réside dans sa **robustesse contre l'oubli**. En maintenant un **plancher** ou un verrou sur les liaisons internes d'un cluster utile, on évite une disparition accidentelle liée à un épisode transitoire de synergie faible ou à une compétition temporaire. Cette méthode favorise également une **réutilisation rapide**. Si le contexte redevient favorable au cluster (même type de données ou même tâche), la **mémoire structurelle** permet de **réactiver** immédiatement les connexions pertinentes sans nécessiter une nouvelle phase d'auto-organisation longue. Par ailleurs, cette approche joue un rôle similaire à la **consolidation des acquis** en neurosciences, où certaines synapses sont stabilisées après avoir prouvé leur utilité dans une situation donnée.

Cependant, certaines **limites** apparaissent. La **consommation mémoire** peut devenir problématique, car l'accumulation de plusieurs “snapshots” de clusters risque d'alourdir les ressources d'un réseau de grande taille, nécessitant alors des critères de sélection et une gestion contrôlée de leur durée de vie. De plus, un excès de **rigidification du réseau** peut nuire à son adaptabilité : si trop de liaisons sont **verrouillées**, le **SCN** perd en flexibilité et peine à intégrer de nouvelles entités ou à restructurer des clusters face aux évolutions du flux. Enfin, un risque d'**obsolescence potentielle** existe. Un cluster pertinent à un instant donné peut devenir **inadéquat** si l'environnement change radicalement. Dans ce cas, verrouiller ses pondérations peut **parasiter** le réseau, d'où la nécessité d'un mécanisme d'actualisation ou d'oubli contrôlé.

Conclusion

La **mémoire structurelle** — consistant à **stocker** des **clusters stables** et à empêcher ou ralentir leur extinction — est un moyen efficace de **préserver** des connaissances jugées stratégiques dans un **SCN** en évolution continue. Elle confère au réseau :

- Une **inertie** positive, maintenant des configurations utiles,
- Une **réactivité** en cas de réapparition d'un contexte favorable,
- Une protection contre l'**érosion** trop rapide liée à l'arrivée de nouvelles données ou à des mécanismes d'inhibition.

Toutefois, il s'agit d'un **compromis**. Une mémorisation excessive fige la plasticité, tandis qu'une mémorisation insuffisante favorise l'amnésie. Bien paramétrée, la **mémoire structurelle** contribue à un équilibre "plasticité-stabilité" plus robuste, permettant d'allier l'**apprentissage incrémental** et la **conservation** de clusters précieux.

9.6.2.3. Freezing partiel : on "gèle" certaines liaisons ω jugées critiques (clusters bien établis)

Dans un **DSL** (*Deep Synergy Learning*) soumis à un **apprentissage continu**, il peut s'avérer souhaitable de **stabiliser** une partie de la structure (les pondérations $\{\omega_{i,j}\}$) qui a déjà démontré sa cohérence et son importance. Cette stratégie, appelée **freezing partiel**, consiste à "verrouiller" certaines liaisons à des valeurs fixes (ou quasi fixes) afin de préserver les **clusters** qui se sont avérés pertinents. Pendant ce temps, le reste du **Synergistic Connection Network (SCN)** reste libre de se réorganiser en fonction des nouvelles données.

A. Motivation et principes mathématiques

Lorsque la dynamique DSL a fait émerger un **cluster** robuste (forte **cohésion interne** des poids $\omega_{i,j}$, synergies $S(i,j)$ élevées et stables), il est légitime de **protéger** cette partie du réseau. Par exemple, dans un SCN où l'on repère un sous-système $\mathcal{C}$ de dimensions importantes qui reste **homogène** et stable, on aimerait éviter qu'un nouvel afflux de données, un changement de synergie ou un recuit agressif ne détruise ce résultat.

Sur le plan **mathématique**, on peut :

$$\Omega(\mathcal{C}) \;=\; \sum_{i,j \in \mathcal{C}} \omega_{i,j},$$

et si $\Omega(\mathcal{C})$ (ou une autre métrique de cohésion) franchit un certain seuil θ_{freeze} et se maintient pendant plusieurs itérations, on déclare le cluster $\mathcal{C}$ "bien formé".

Le "gel" d'une liaison $\omega_{p,q}$ signifie qu'on **bloque** sa valeur à l'avenir :

$$\omega_{p,q}(t+1) \;=\; \omega_{p,q}(t),$$

ou à défaut, on peut imposer une **amplitude de variation** extrêmement réduite :

$$\omega_{p,q}(t+1) \;=\; \omega_{p,q}(t) \;+\; \epsilon, \quad \epsilon \ll 1.$$

Ainsi, la **règle** de mise à jour habituelle (basée sur $S(p,q)$ et $\omega_{p,q}$) ne s'applique plus, ou de façon très amortie, aux liaisons gelées. On peut voir cela comme l'annulation (ou la forte pénalisation) de la dérivée $\partial \mathcal{J} / \partial \omega_{p,q}$ si l'on raisonne dans un cadre d'énergie ou de fonction coût.

Ce verrouillage **empêche** le cluster de se déformer malgré les flux de nouvelles entités ou les adaptations dans d'autres zones du SCN. On gagne en **stabilité**, mais on perd en **souplesse**. Une entité $\mathcal{E}$ appartenant à ce cluster figé aura plus de difficultés à migrer vers un autre cluster si, plus tard, cela devient pertinent.

B. Stratégies de sélection des liens à geler

Une façon de procéder est de fixer un **seuil** ω_{freeze} :

- Si, pour un couple (i,j) donné, la pondération $\omega_{i,j}(t)$ dépasse ω_{freeze} **durablement** (par exemple, au-delà de ω_{freeze} pendant T_0 itérations consécutives), on la **déclare** gelée.

- Cette règle s'applique souvent aux liaisons **internes** à un cluster $\mathcal{C}$ fortement associé (i.e., la majorité de ses couples (i,j) dépassent ω_{freeze}).

On peut aussi exiger que $\omega_{i,j}$ soit **quasi stationnaire** :

$$|\omega_{i,j}(t') - \omega_{i,j}(t'+1)| \ < \ \epsilon, \quad \text{pour tout } t' \in [t_0, t_1],$$

indiquant que la liaison $\omega_{i,j}$ ne bouge presque plus. On **officialise** alors ce gel, au motif qu'il n'évolue déjà plus.

Plutôt que de figer intégralement $\omega_{p,q}(t+1) = \omega_{p,q}(t)$, on peut **réduire** fortement la vitesse d'apprentissage (baisser η de façon drastique) pour ces liaisons. Cela permet une **micro-adaptation** sans remettre en cause l'essentiel du cluster.

C. Impact sur la dynamique

Le gel partiel "**solidifie**" un cluster en empêchant la mise à jour de ses liaisons internes, ce qui peut :

- **Réduire** le nombre de variables actives dans la dynamique, donc **accélérer** les calculs.

- Faire émerger des **blocs** où l'ajustement local est figé, pendant que les frontières externes restent mobiles.

En mode **apprentissage continu**, un cluster gelé ne sera pas **détruit** par l'arrivée d'entités nouvelles ou un excès de bruit. On **conserve** ainsi un noyau de connaissances stables, assurant la **mémoire** de découvertes passées.

Si, ultérieurement, le contexte évolue fortement, un **cluster** anciennement gelé peut se révéler obsolète ou sous-optimal pour la nouvelle situation. Le gel empêche la "migration" des entités concernées. On peut donc prévoir une option de "dégel" (*unfreeze*) si un certain indicateur global l'exige (p. ex. si un macro-niveau de contrôle détecte un conflit ou un changement de distribution majeure).

D. Théorisation en termes d'énergie

On peut modéliser le gel comme un **terme** dans la fonction d'énergie (ou de coût) $\mathcal{J}$, qui rend très onéreux le fait de modifier $\omega_{p,q}$. Par exemple, si $\omega_{p,q}$ doit rester à ω^*, on introduit $\kappa\left(\omega_{p,q} - \omega^*\right)^2$ avec $\kappa \gg 1$. En descente de gradient, ce terme empêche $\omega_{p,q}$ de s'éloigner, à moins d'un bouleversement majeur.

De manière plus radicale, on peut **restreindre** l'espace de $\{\omega\}$ en omettant $\omega_{p,q}$ des variables libres, le fixant à ω^*. C'est un "sous-espace" de la dynamique, la liaison gelée n'étant plus optimisée.

Avantages et Limites

Dans ce cadre, plusieurs **avantages** émergent. La **sauvegarde de l'information** permet de préserver les clusters ayant fait leurs preuves, évitant ainsi un *catastrophic forgetting*. De plus, la **réduction du nombre de mises à jour** diminue le nombre de variables actives, ce qui peut

accélérer l'itération et alléger la charge de calcul. Enfin, la **robustesse** du SCN est renforcée, car les parties gelées deviennent moins sensibles aux perturbations mineures, garantissant une base stable sur laquelle le réseau peut s'appuyer.

Cependant, certaines **limites** apparaissent. Une **rigidité excessive** peut empêcher l'adaptation du réseau en cas de changement radical, rendant certains clusters gelés inadaptés et contrariant ainsi la réorganisation nécessaire. De plus, la nécessité de **définir des critères précis** devient incontournable : il faut décider **quand** geler un lien et **quand** le réactiver, ce qui impose des heuristiques claires basées sur des seuils de stabilité ou le suivi du contexte. Enfin, la **gestion de l'hétérogénéité** peut devenir problématique. Certains liens gelés risquent d'être en conflit avec de nouvelles synergies émergentes, nécessitant un mécanisme pour arbitrer ces situations, par exemple par un macro-niveau d'ordonnancement.

En définitive, le **freezing partiel** apparaît comme un **levier stratégique** pour équilibrer *plasticité* et *stabilité* dans un SCN évolutif. Il favorise la **continuité** de l'apprentissage tout en protégeant les clusters déjà formés contre une dégradation prématurée.

Conclusion

Le **freezing partiel** de liaisons ω considérées comme "critiques" (car appartenant à des **clusters bien établis**) vise à :

- **Protéger** ces acquis contre les aléas de la dynamique et l'afflux de nouvelles entités.
- **Réduire** la complexité et les ajustements permanents, en retirant certaines liaisons du champ de la plasticité.
- **Maintenir** la cohérence locale de sous-groupes déjà validés, tout en laissant le **reste** du réseau s'adapter à de nouveaux stimuli.

9.6.3. Exemples Concrets

Dans ce cadre de **flux évolutif** et de **rétroaction** (cf. §9.6.2), on peut illustrer la coexistence entre l'**ancien** et le **nouveau** à travers plusieurs scénarios où le **DSL** (Deep Synergy Learning) doit gérer simultanément la **rétention** (ne pas oublier totalement le passé) et la **plasticité** (accueillir les nouveaux schémas). Le premier exemple (9.6.3.1) se situe dans un **système textuel**, où la dynamique de mots-clés en flux continu peut contraindre le **SCN** (Synergistic Connection Network) à renouveler ses liens $\omega_{i,j}$ de manière adaptative, tout en essayant de ne pas effacer prématurément des mots-clés potentiellement encore utiles.

9.6.3.1. Système textuel : anciens mots-clés chassés par les nouveaux ?

Dans un **SCN** dédié à l'**analyse** de flux textuels, chaque **mot-clé** ou token important est représenté comme une **entité** $\mathcal{E}_i$. L'**idée** est de calculer une **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ tenant compte, par exemple, de la **co-occurrence** ou de la **distance sémantique** entre mots. À chaque itération t, la **mise à jour** des pondérations $\omega_{i,j}(t)$ suit une équation inspirée de la logique du Deep Synergy Learning, par exemple :

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

Dans un **environnement textuel**, il arrive fréquemment que de **nouveaux mots-clés** surgissent lorsqu'un événement ou une tendance émergente. Ces tokens inédits suscitent alors un **renforcement rapide** de leurs liaisons $\{\omega_{i,j}\}$ avec d'autres entités, en particulier si elles partagent le même *buzz* ou un degré de **similarité** élevé. La pondération $\omega_{i,j}(t)$ s'amplifie donc pour ces mots récents, d'autant plus si des mécanismes d'**inhibition** limitent la capacité de chaque entité à maintenir un trop grand nombre de liens.

A. Problématique : Effacement progressif des mots-clés "vieillissants".

Lorsqu'un flux continu de **nouveaux** mots-clés déferle, la distribution des pondérations peut se réorienter vers ces mots. Les **mots-clés historiques**, moins utilisés sur la période récente, voient alors leurs liaisons $\omega_{i,j}$ décroître par simple application du terme $\tau\,\omega_{i,j}(t)$. Au fil des itérations, cette décroissance peut s'amplifier si aucune utilisation ne vient **ranimer** l'intérêt pour ces anciens mots. On aboutit à un **risque** de "sur-oubli". Certains tokens antérieurement pertinents se retrouvent complètement effacés, alors qu'ils pourraient réapparaître (par exemple, un sujet d'actualité saisonnier).

Lorsque le flux de nouveauté est **très rapide**, la convergence vers les nouveaux mots tend à chasser, presque trop efficacement, la "mémoire" des anciens. Sur le plan **mathématique**, la somme $\sum_j \omega_{i,j}\,(t)$ pour un mot-clé "vieillissant" se réduit jusqu'à un niveau quasi nul, car la compétition pour liaisons actives est dominée par les tokens les plus récents.

B. Risques de sur-oubli dans le SCN textuel.

La **dynamique** DSL conduit à avantager ce qui est détecté comme **fortement synergique** dans le moment présent. Si les **nouveaux** mots-clés captent l'essentiel de la synergie, l'ancien se **délite** sans nécessairement conserver une pondération résiduelle. Le **risque** est que ce mécanisme d'oubli soit trop *drastique,* ce qui empêche de **réactiver** rapidement l'entité correspondante si, par exemple, le flux de textes redevient propice à ce mot-clé. Dans la littérature de l'apprentissage incrémental, on parle de perte de **"knowledge reuse"**, où l'oubli entraîne une forme de réapprentissage coûteux si le même concept revient ultérieurement.

C. Mécanismes de rétention et d'équilibre.

Pour **modérer** ce phénomène, il est envisageable d'**introduire** un **terme de mémoire** $\Delta_{\text{memory}}(i,j)$ dans l'équation de mise à jour :

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

L'idée est qu'un mot-clé $\mathcal{E}_i$ peut porter un certain **crédit d'historique** qui vient partiellement **compenser** la décroissance lorsqu'il est momentanément moins utilisé. Ainsi, si un concept a **prouvé** sa pertinence dans le passé, on ne le laisse pas s'éteindre trop rapidement. Un réglage fin de $\Delta_{\text{memory}}(i,j)$ permet de gérer un **compromis**. Une valeur trop grande fige le réseau, rendant le SCN peu sensible aux nouveaux mots, tandis qu'une valeur trop faible n'endigue pas l'oubli.

Ce **terme** de mémoire peut être défini selon plusieurs critères, comme le nombre total d'occurrences passées, la longévité du token dans le système ou le niveau moyen de co-occurrence qu'il entretenait avec d'autres entités. On peut aussi envisager une **inhibition modérée**, dans laquelle on limite la compétition au lieu de la renforcer pleinement, ce qui évite de forcer la disparition trop rapide des clusters historiques.

D. Exemple illustratif : actualités et "buzz".

Lors d'un événement sportif majeur ou d'une élection, un **lot** de nouveaux mots-clés prend soudainement une grande importance. Le SCN reconfigure alors ses liaisons ω pour refléter cette nouveauté. Une fois l'événement passé, le flux de documents contenant ces mots peut diminuer, invitant la pondération ω à se recentrer sur d'autres sujets. Un **sous-oubli** maîtrisé peut être sain, car on ne veut pas conserver éternellement des topics obsolètes. Mais si la **prochaine** édition de l'événement survient et que le SCN a totalement effacé les liaisons associées, il faudra tout **réapprendre** au lieu de bénéficier d'un **socle** qui faciliterait la réactivation de ce cluster.

Conclusion.

Dans un **système textuel** à flux continu, l'arrivée de **nouveaux** mots-clés a pour effet d'**augmenter** leur pondération ω et, souvent, de **chasser** les **anciens** mots, devenus moins fréquents. Cette dynamique DSL, si elle n'est pas **contrebalancée** par un mécanisme de rétention (ou de mémoire) adaptatif, peut conduire à un **sur-oubli**. Sur le plan **mathématique**, la chute trop rapide de $\omega_{i,j}$ pour les tokens historiques se formalise par la dominance de $\tau\,\omega_{i,j}(t)$ et le renforcement concurrent réservé aux entités nouvelles. Pour **atténuer** ce phénomène et préserver la capacité à retrouver rapidement d'anciens clusters, on introduit des **termes** de **mémoire** $\Delta_{\text{memory}}(i,j)$ ou on paramètre soigneusement la **compétition** pour atteindre un juste équilibre entre **plasticité** (à l'écoute des nouveautés) et **stabilité** (préservation de la connaissance passée).

9.6.3.2. Robotique : anciens comportements sensorimoteurs qui pourraient redevenir utiles plus tard

Dans un **système** robotique en **apprentissage continu**, il est fréquent que le **SCN** (Synergistic Connection Network) manipule des **entités** $\mathcal{E}_i$ représentant différents **capteurs** et **actionneurs**. À force d'**auto-organisation**, le réseau fait apparaître des **clusters** associant certaines observations sensorielles à des schémas moteurs cohérents. On obtient ainsi des **comportements** sensorimoteurs bien identifiés (par exemple, un module "approche et saisie" ou un module "évitement d'obstacle"). Lorsque ces comportements cessent d'être utiles, le **DSL** (Deep Synergy Learning) laisse leurs **liaisons** s'affaiblir ou s'inhiber. Cependant, grâce aux mécanismes d'apprentissage incrémental, ces anciens couplages demeurent en "dormance" et peuvent être **réactivés** ultérieurement si un **contexte** similaire se présente.

A. Concept : Dormance et Réactivation de Comportements

Dans un système robotique exploitant la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$, un **cluster** de capteurs et d'actionneurs forme un **module sensorimoteur** lorsque les **pondérations** $\omega_{i,j}$ sont élevées pour certaines paires (i,j). On peut ainsi parler d'un **comportement**, où le robot apprend qu'en activant un ensemble donné de moteurs lors de la détection d'un certain pattern de capteurs, il obtient un résultat pertinent, comme une posture de saisie ou un mouvement d'évitement. Si la mission change, ces liens ω se voient parfois soumis à une **inhibition** ou à un **terme** de décroissance, laissant le **cluster** inactif. Sur le plan **mathématique**, on dit que les liaisons internes se rapprochent d'une valeur réduite $\omega_{p,q}(t) \approx \epsilon$ sans être nécessairement nulles. Ce phénomène correspond à un état de **dormance**, où le comportement n'est plus sollicité sans être entièrement effacé.

Le point crucial est que ce **savoir inactif** n'est pas définitivement perdu. Si des **conditions** similaires réapparaissent (par exemple, le robot doit à nouveau saisir un objet dans un contexte proche de celui rencontré précédemment), la **fonction de synergie** $S(\mathcal{E}_p, \mathcal{E}_q)$ redevient forte pour ces mêmes liaisons $\omega_{p,q}$. La mise à jour

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

peut alors **réactiver** rapidement les pondérations associées au cluster sensorimoteur mis en sommeil. La valeur n'est pas réinitialisée à zéro. Un **noyau** de pondération subsiste, permettant de renforcer la liaison $\omega_{p,q}$ en quelques itérations seulement, sans devoir tout réapprendre depuis le début.

B. Mécanismes DSL Favorisant la "Renaissance" de Comportements

Dans la **logique** du DSL, plusieurs **mécanismes** peuvent encourager ce type de dormance et de réactivation.

Le premier repose sur la **parsimonie** contrôlée, où les liaisons dont la pondération passe sous un certain $\omega_{\min}$ ne sont pas coupées instantanément, mais seulement maintenues à un niveau très bas. Cela se traduit par :

$$\omega_{p,q}(t+1) \;=\; \max\!\big(\omega_{\min},\; \omega_{p,q}(t) + \eta\left[S(p,q) - \tau\,\omega_{p,q}(t)\right]\big).$$

Ainsi, même quand un **module** sensorimoteur n'est plus requis, ses liaisons internes ne tombent jamais exactement à zéro. Elles restent **latentes** et susceptibles d'être **réactivées**.

Un deuxième mécanisme, souvent lié à l'**inhibition compétitive**, fait que l'entité (capteur ou moteur) ne peut maintenir qu'un nombre limité de liaisons fortes à la fois. Les anciens liens ne sont plus renforcés et deviennent faibles, mais peuvent **rebondir** si une haute **synergie** se manifeste à nouveau. Lorsque l'**événement** sensoriel ou la tâche motrice associée réapparaît, la **cohérence** interne du cluster ressort, provoquant une augmentation rapide de $\omega_{p,q}$.

Un troisième aspect est la **boucle** auto-organisée. Dès qu'une partie des liaisons relevant d'un comportement se remet à augmenter, d'autres entités adjacentes verront leur synergie relancée. Cela crée un **effet domino**, où la redécouverte de l'activation sensorielle liée à l'ancien module ravive la configuration d'ensemble. En quelques itérations, l'ancien **comportement** redevient fortement établi.

C. Exemple : Bras Robotique et Locomotion Combinée

Un robot muni de roues et d'un bras préhensile peut apprendre un **comportement** "approche et saisie", lequel repose sur la coopération de capteurs de vision, d'un détecteur de proximité, d'un algorithme de reconnaissance de forme, et du contrôle de la pince ou du bras. Lorsque la tâche du robot évolue, qu'il s'agisse d'inspection ou de patrouille, le module "approche et saisie" devient inutile. Ses liaisons ω sont alors inhibées, descendent sous un seuil et passent en **dormance**.

Si, ultérieurement, le contexte retrouve la nécessité de saisir un objet, la **dynamique** DSL capte la hausse de synergie entre la vision, la pince, et le paramétrage moteur associé. Les liaisons mises en sommeil se **rallument**, $\omega_{p,q}$ quittant la zone $\omega_{\min}$ pour retrouver des valeurs élevées. Le robot récupère alors son comportement de saisie en un temps bien plus court que s'il devait tout réapprendre de zéro. Sur le plan **mathématique**, on profite d'une mémoire interne déjà

structurée (un quasi-attracteur), qui n'a besoin que d'un "signal" (une synergie accrue) pour redevenir opérationnel.

D. Considérations Mathématiques et Convergence

On peut interpréter ces **patterns** sensorimoteurs comme autant de **configurations** stables dans l'espace des liaisons $\{\omega_{i,j}\}$. Chacune d'elles constitue un quasi-attracteur dont on **n'exclut** pas la présence, même si elle est temporairement faiblement pondérée. Le fait qu'il existe un **plancher** $\omega_{\min}$ ou que la diminution de ω ne soit pas totale empêche la disparition complète des anciens comportements.

La **convergence** vers un comportement en dormance s'effectue dès que le **signal** $S(i,j)$ redevient positif et dépasse un certain seuil, permettant à la boucle itérative $\omega(t+1) = \omega(t) + \eta[S(i,j) - \tau\,\omega(t)]$ de regonfler rapidement les liaisons. Ce phénomène s'apparente à de la **mémoire associative**, où une stimulation partielle, comme la réapparition d'une situation, suffit à réactiver un *cluster* entier sans nécessiter un apprentissage ex nihilo.

Conclusion

Dans un **SCN** robotique évolutif, les **anciens comportements sensorimoteurs** ne s'effacent pas intégralement lorsque la tâche ou l'environnement change. Les liaisons qui les sous-tendent entrent en **dormance** (pénalisées, mais pas anéanties), ce qui autorise leur **réactivation** ultérieure. Cette capacité procure :

Avantages
Elle assure une **flexibilité** dans la récupération d'un comportement déjà maîtrisé, diminue le temps d'apprentissage lorsqu'un contexte se répète, et renforce la **robustesse** du système face à la variabilité des missions. Elle améliore la **plasticité** globale, puisque le **DSL** n'a pas à réencoder à partir de zéro des patterns pourtant déjà connus.

Limites
Elle peut toutefois maintenir des comportements ou sous-ensembles de liens inutiles pour de longues périodes, ce qui, si mal géré, risque de gaspiller de la ressource ou de la mémoire. Par ailleurs, un comportement très ancien mais inadapté pourrait être préservé alors qu'il serait plus optimal de le laisser disparaître si la situation ne doit jamais se reproduire. Un compromis demeure nécessaire pour moduler la **dormance**, le **gel** ou la **suppression**, selon la dynamique réelle de l'application robotique.

9.6.3.3. Observations : analyses quantitatives de la "rétention" vs. la "plasticité"

Dans un **SCN** conçu pour l'**apprentissage continu** (cf. sections **9.6.1** et **9.6.2**), la **dynamique** des liaisons $\omega_{i,j}$ cherche un **compromis** entre la **rétention** (conserver la structure acquise) et la **plasticité** (s'ajuster aux nouveautés). Cette section **9.6.3.3** propose des indicateurs et observations **quantitatives** pour évaluer comment le réseau gère cet **équilibre**. L'idée est de mesurer la **stabilité** des pondérations lorsqu'affluent de nouvelles entités ou surviennent des changements de contexte, tout en vérifiant la **capacité** du réseau à se réorganiser efficacement.

A. Définitions de la "rétention" et de la "plasticité"

Dans l'esprit d'un **SCN**, la mise à jour s'écrit souvent :

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

voire agrémentée de mécanismes d'**inhibition** ou de **régularisation**. Pour **quantifier** le niveau de **stabilisation** ou de **réorganisation**, on peut définir deux **mesures** essentielles.

Rétention $\mathcal{R}(t)$

La **rétention** représente la capacité du **réseau** à **conserver** les liaisons et les clusters existants malgré l'arrivée de nouvelles entités. Une manière de la formaliser consiste à comparer la **configuration** actuelle $\boldsymbol{\Omega}(t)$ des pondérations $\{\omega_{i,j}(t)\}$ à celle d'un **instant** antérieur $\boldsymbol{\Omega}(t-1)$. On peut par exemple utiliser la **similarité cosinus** entre ces deux "vecteurs" :

$$\mathcal{R}(t) = \frac{\sum_{i,j}\ \omega_{i,j}(t)\,\omega_{i,j}(t-1)}{\sqrt{\sum_{i,j}\left[\omega_{i,j}(t)\right]^2 \times \sum_{i,j}\left[\omega_{i,j}(t-1)\right]^2}}.$$

Une valeur de $\mathcal{R}(t)$ proche de 1 indique que la configuration de pondération demeure **quasi inchangée** d'une itération à l'autre, reflétant une **stabilité** élevée (haute rétention). Une valeur basse témoigne au contraire d'un **changement** substantiel dans l'organisation du réseau.

Plasticité $\mathcal{P}(t)$

La **plasticité** exprime la **faculté** du SCN à **réviser** ses liens et à incorporer des données nouvelles. On peut introduire un **taux de réécriture** des liaisons pour chaque itération. Si l'on note

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

on peut choisir un seuil $\varepsilon > 0$ et compter la **proportion** de liaisons dont la variation excède ε. On définit alors :

$$\mathcal{P}(t) = \frac{1}{n^2}\sum_{i,j}\ \mathbf{1}\left[\delta_{i,j}(t) > \varepsilon\right],$$

où $\mathbf{1}[\cdot]$ est l'indicateur de la condition $\delta_{i,j} > \varepsilon$. Ainsi, $\mathcal{P}(t)$ reflète la **part** (en pourcentage) des liens $\omega_{i,j}$ ayant subi un changement notable. Une **plasticité** élevée ($\mathcal{P}(t)$ importante) indique que le SCN est en pleine réorganisation, tandis qu'une plasticité basse traduit une relative inertie de la structure.

B. Observation et Interprétation

Il est intéressant de tracer au cours du **temps** les courbes $\mathcal{R}(t)$ et $\mathcal{P}(t)$. Généralement, on constate l'existence d'un **dilemme** :

Lorsque la **rétention** est élevée, cela signifie que le réseau ne modifie que très faiblement ses pondérations. Par conséquent, la **plasticité** se trouve souvent minorée, et l'ajustement à de nouvelles entités devient plus lent ou plus limité. Inversement, une **plasticité** très prononcée indique que la structure se remet en question à chaque itération, ce qui, si trop poussé, peut nuire à la **stabilité** et faire chuter $\mathcal{R}(t)$. Sur le plan **mathématique**, ces deux tendances (stabilité vs. réécriture) se retrouvent dans les phases d'**apprentissage continu**, où l'arrivée de nouveaux flux de données provoque un pic de **plasticité** ($\mathcal{P}(t)$), suivi d'une phase de **stabilisation** où la **rétention** $\mathcal{R}(t)$ remonte lorsque le réseau s'est ajusté.

Lorsque le SCN détecte un afflux de **nouvelles entités**, $\mathcal{P}(t)$ connaît une **hausse**, entraînant une réorganisation des liaisons afin d'intégrer ces données récentes. On observe alors un

glissement du réseau depuis un état stable (haute $\mathcal{R}(t)$, faible $\mathcal{P}(t)$) vers un état **transitoire** où $\mathcal{R}(t)$ diminue et $\mathcal{P}(t)$ monte. Après un laps de temps (variable selon les paramètres η, τ, etc.), la structure s'équilibre, et on retrouve un **niveau** stable de rétention plus élevé (mais éventuellement un peu moindre qu'avant si de nouveaux liens sont consolidés).

C. Observations Numériques (Exemple Schématique)

Pour illustrer ce phénomène, on peut conduire une **simulation**. Supposons qu'un SCN initialement composé de $n = 50$ entités s'organise sur 100 itérations, puis reçoive l'ajout de 10 entités supplémentaires à l'itération $t = 100$. Dans la phase initiale (itérations 40–100), $\mathcal{R}(t)$ demeure élevée (≈ 0.9) car le réseau a déjà convergé à un arrangement stable. On constate que la **plasticité** $\mathcal{P}(t)$ y est faible (ex. ≈ 0.05), signifiant qu'assez peu de liaisons se réactualisent d'un pas à l'autre.

Dès que les 10 **nouvelles entités** sont introduites, $\mathcal{R}(t)$ chute pour atteindre 0.65–0.70, tandis que $\mathcal{P}(t)$ grimpe à 0.3 ou 0.4 (une proportion importante de liaisons dépassent le seuil ε). Puis, au fil d'une trentaine d'itérations supplémentaires, la stabilisation d'une nouvelle topologie fait remonter $\mathcal{R}(t)$ (~0.85) et tomber $\mathcal{P}(t)$ sous 0.1. On y observe clairement le **cycle**, alternant entre stabilisation, perturbation due au flux récent, puis re-stabilisation.

D. Conclusion : Retenir ET S'adapter

L'analyse **quantitative** de la **rétention** $\mathcal{R}(t)$ et de la **plasticité** $\mathcal{P}(t)$ révèle clairement la tension entre la **stabilité** (préserver les acquis) et la **flexibilité** (reconnaître et intégrer le nouveau). Sur le plan **opérationnel**, on recherche souvent un point d'équilibre permettant au **SCN** de ne pas "oublier" trop vite les clusters établis (sinon on sombre dans un *catastrophic forgetting*), tout en autorisant les réajustements nécessaires pour rester à l'écoute d'un **contexte** changeant.

On ajuste pour cela plusieurs **paramètres**. D'abord, le **taux** d'apprentissage η influe directement sur l'équilibre du système. Une valeur trop grande accroît la plasticité mais réduit la rétention, tandis qu'une valeur trop petite favorise la rétention au risque de figer le système. On contrôle aussi le **coefficient** τ de décroissance, la gestion de l'**inhibition**, ou la présence de termes de **régularisation** (voir **9.6.2.1**). L'objectif est de caler le **SCN** dans une zone de **souplesse** suffisante pour absorber des nouveautés, sans pour autant perdre la **mémoire** constructive accumulée. Cette notion de "sweet spot" entre rétention et plasticité constitue l'un des pivots de la **théorie** et de la **pratique** des réseaux **DSL** évolutifs.

9.7. Méthodes d'Équilibrage entre Plasticité et Stabilisation

Dans un **SCN** (Synergistic Connection Network) évoluant en temps réel, la **plasticité** (capacité à se reconfigurer) et la **stabilisation** (tendance à figer les clusters acquis) forment deux **pôles** souvent en tension. D'un côté, trop de plasticité risque d'engendrer de la **volatilité** structurelle ; de l'autre, trop de stabilisation peut figer le réseau dans une **configuration** qui n'évolue plus malgré l'arrivée de nouvelles données. Le **chapitre 9** a développé l'idée d'**apprentissage continu** et d'**ajustement** des paramètres du SCN (notamment η, τ, la synergie $S(\cdot,\cdot)$, etc.) pour gérer l'arrivée ou la suppression d'entités (addEntity,removeEntity).

Dans cette **section 9.7**, l'accent est mis sur les **méthodes** visant à trouver un **équilibre** entre plasticité et stabilisation. En ajustant **dynamiquement** certains paramètres clés, tels que η, τ ou encore l'inhibition γ, le **SCN** peut conserver sa **flexibilité** tout en **convergeant** suffisamment pour identifier des clusters stables.

9.7.1. Paramétrage Dynamique de η et τ

La **règle** DSL classique :

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

met en évidence deux **paramètres** principaux :

- η : le **learning rate**,
- τ : le **facteur** de décroissance ou d'amortissement.

Leur réglage impacte directement la **vitesse** et la **sélectivité** de la mise à jour. Trop d'inertie (faible η, fort τ) conduit à un SCN lent ou exagérément stabilisé ; trop de réactivité (forte η, faible τ) le rend volatile ou éparpillé. Nous détaillons ici comment **adapter** η et τ *en cours de route* (dynamique) pour mieux gérer l'alternance entre phases de reconfiguration (plasticité) et phases de consolidation (stabilisation).

9.7.1.1. Réduire η (learning rate) au fur et à mesure que le SCN s'établit, puis le remonter si un nouvel afflux de données arrive

Lorsqu'on déploie un **SCN** (*Synergistic Connection Network*) dans une optique d'**apprentissage continu**, l'un des points clés pour assurer le bon compromis entre **plasticité** et **stabilisation** consiste à **faire varier** le **taux d'apprentissage** η au cours du temps. Dans un cadre mathématique, on peut définir $\eta(t)$ comme une **fonction** décroissante, ce qui permet au réseau de procéder à des ajustements rapides au début, puis de se **consolider** au fur et à mesure qu'il converge. On peut également réaugmenter η lorsqu'un **nouvel afflux** de données survient, autorisant ainsi une nouvelle phase d'ajustement plus flexible.

Principe mathématique du découpage en phases

La démarche la plus classique pour **faire diminuer** le learning rate consiste à définir une **loi** $\eta(t)$ décroissante, par exemple :

$$\eta(t) = \frac{\eta_0}{1+\alpha\,t} \quad \text{ou} \quad \eta(t) = \eta_0 \exp(-\alpha\,t),$$

où η_0 est le taux initial et $\alpha > 0$ un paramètre de décroissance. Dans les premières itérations, η est relativement élevé, ce qui favorise une **exploration** intense et permet au réseau de s'organiser rapidement. À mesure que t croît, $\eta(t)$ diminue, stabilisant les pondérations $\omega_{i,j}$ et évitant d'éternelles oscillations ou réajustements erratiques.

Il est parfois nécessaire, d'un point de vue **mathématique**, d'ajouter un mécanisme permettant de **remonter** η si un changement brutal de distribution survient. On peut alors adopter un schéma **multi-phase**, où $\eta(t)$ suit une décroissance progressive tant que le réseau ne subit aucune perturbation. Mais lorsqu'une **nouvelle** vague d'entités $\{\mathcal{E}_{n+1}, \dots, \mathcal{E}_{n+k}\}$ apparaît ou qu'un contexte inédit se manifeste, on peut :

$$\eta(t) = \begin{cases} \eta_1(t) & \text{si } t < t_1, \\ \eta_2(t - t_1) & \text{si } t \geq t_1, \end{cases}$$

avec une seconde phase η_2 plus élevée au moment de l'injection des nouveautés. Cette remontée du learning rate redonne de la **plasticité** au **SCN**, lui permettant d'**assimiler** efficacement les entités fraîchement introduites. Une fois l'ajustement effectué, on repart sur une courbe de décroissance, favorisant une nouvelle **stabilisation**.

Scénario de "nouvel afflux de données"

Lorsqu'un afflux de données survient à l'instant t_1, la structure $\{\omega_{i,j}\}$ doit rapidement s'adapter à la **synergie** modifiée. Sur le plan **mathématique**, si l'on conserve l'équation de mise à jour :

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

alors le **facteur** $\eta(t)$ amplifie (ou restreint) l'ampleur de $\Delta\omega_{i,j}(t)$. Si η est assez grand, le **SCN** se réorganise vivement, augmentant la **plasticité**. S'il est trop petit, la **réponse** à la nouveauté peut être trop lente, risquant un "apprentissage inadapté" ou un "glissement" insuffisant dans l'espace des pondérations. Faire **repartir** η à un niveau plus élevé, puis le **rebaisser** progressivement, crée un **cycle** :

3. **Réveil** de la plasticité pour adapter les poids au changement.
4. **Consolidation** lorsque le SCN se met à jour sur les nouvelles entités et que la synergie se stabilise.

Équations de mise à jour

Dans un tel schéma, la dérivée discrète :

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

montre clairement que si $\eta(t)$ décroît trop tôt, le SCN devient **rigide** et ne peut plus apporter de corrections significatives. À l'inverse, si η reste constamment élevé, il devient difficile d'atteindre un **plateau** stable. Il est donc naturel de **calibrer** $\eta(t)$ en commençant par une valeur initialement élevée (η_0) pour assurer un **démarrage** rapide, puis en la réduisant progressivement pour consolider la structure. En cas de **rupture** ou d'**afflux** massif de nouvelles entités, une réaugmentation peut être nécessaire.

Une **approche adaptative** plus avancée repose sur la surveillance de la **variance** globale du réseau, qui mesure la dispersion des pondérations ou de la synergie. Lorsque cette variance reste faible, η est progressivement réduit. En revanche, si elle s'emballe ou si une entrée soudaine de nouvelles données est détectée, η est temporairement réajusté à la hausse.

Avantage

La **modulation** du learning rate évite de rester en phase de forte **exploration** (avec un η constant et élevé) ou de tomber trop rapidement dans une **phase** de rigidité (avec η constamment faible). On bénéficie ainsi d'une **courbe** d'auto-organisation caractéristique. La mise à jour est **ample** au départ, permettant au SCN de structurer une **organisation** pertinente, puis elle se **réduit progressivement** pour assurer la convergence. Lorsqu'un changement majeur survient, comme un nouveau contexte ou l'ajout d'entités, une augmentation de η relance la plasticité. Une fois l'adaptation effectuée, le schéma se referme progressivement pour stabiliser la structure.

En pratique, cette technique, déjà familière en **apprentissage** neuronal classique (réduction progressive du taux d'apprentissage, *annealing* du learning rate), se transpose sans difficulté dans la dynamique DSL, où elle devient un **levier** d'**équilibre** entre plasticité (quand η est haut) et stabilité (quand η est bas).

Conclusion

La réduction progressive de η au fil de l'apprentissage est un **principe classique** qui, appliqué au **Deep Synergy Learning**, permet de maîtriser la **plasticité** initiale et la **stabilisation** ultérieure. La possibilité d'**augmenter** η lors de l'arrivée de **nouvelles données** ou d'un changement de distribution prolonge ce principe dans une **approche multi-phase**. La plasticité est réactivée lorsque nécessaire, puis le système retrouve progressivement sa convergence. Sur le plan **mathématique**, on manipule une équation de mise à jour paramétrée par $\eta(t)$ dont la loi d'évolution (exponentielle, en escalier, etc.) dicte le "rythme" selon lequel le **SCN** se reconfigure ou se fige. Cette logique assure un **apprentissage** plus robuste, capable de marier **convergence** et **réactivité** à un **contexte** changeant.

9.7.1.2. Adapter τ (décroissance) pour gérer l'équilibre : si trop haut, on écrase trop vite les liens, si trop bas on les retient trop

Dans la mise à jour standard d'un **SCN** (*Synergistic Connection Network*) en **Deep Synergy Learning**, la pondération $\omega_{i,j}$ obéit à l'équation :

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

où $\tau > 0$ sert de **facteur de décroissance** ou d'**amortissement**. Le rôle de τ est de réguler la **dynamique** des liaisons $\omega_{i,j}$ entre **croissance** et **décroissance**. Une valeur suffisamment grande empêche une augmentation **illimitée**, tandis qu'une valeur trop faible laisse la structure s'encombrer de liens superflus. L'objectif est de trouver un **équilibre** : si τ est trop élevé, les liaisons se voient « écrasées » avant même de pouvoir s'établir, tandis que si τ est trop faible, le réseau ne se débarrasse pas assez rapidement des liens de faible ou moyenne pertinence.

A. Rôle de τ dans l'Équilibre des Liens

Le **terme** $\tau\,\omega_{i,j}(t)$ dans la parenthèse $\big[S(i,j) - \tau\,\omega_{i,j}(t)\big]$ exprime la **pénalisation** de la liaison en fonction de sa valeur courante. Si τ est trop grand, même une synergie positive $S(i,j)$ peut être compensée ou dépassée par la pénalité $\tau\,\omega_{i,j}(t)$, faisant redescendre $\omega_{i,j}$. À l'inverse, un τ faible ne freine pratiquement pas la croissance de $\omega_{i,j}$. Sur le plan **mathématique**, on aboutit souvent à l'idée qu'en régime stationnaire, on a :

$$S(i,j) \approx \tau\, \omega_{i,j}^{*} \quad \Rightarrow \quad \omega_{i,j}^{*} \approx \frac{S(i,j)}{\tau}.$$

Si τ est grand, la valeur d'équilibre $\omega_{i,j}^{*}$ est plus petite, même si la synergie est relativement élevée. Si τ est petit, on tolère aisément des liaisons plus fortes.

Trop haut (τ trop grand). Les liaisons ont tendance à **rester faibles**, car il faut une synergie vraiment considérable pour continuer à alimenter la croissance de $\omega_{i,j}$. Le **SCN** peut alors peiner à **former** des clusters stables, basculant vers un réseau très parcimonieux, où beaucoup de liens restent proches de zéro. Dans des scénarios extrêmes, on obtient un graphe trop mince, voire déconnecté.

Trop bas (τ trop petit). Les liaisons s'auto-entretiennent trop facilement, même si $S(i,j)$ n'est pas très grande, car la pénalité $\tau\, \omega_{i,j}(t)$ demeure marginale. Le risque est de **retenir** de nombreux liens "moyens" : la structure peine alors à **séparer** correctement les entités en clusters distincts, car beaucoup de connexions demeurent actives, ralentissant la formation de pôles de synergie vraiment forts.

B. Comment Ajuster τ ?

Un premier choix consiste à fixer **une valeur** τ en fonction des échelles typiques de $S(i,j)$. Si, par exemple, la synergie se situe en moyenne entre 0 et 1, il est courant d'essayer $\tau \approx 0.5$ ou $\tau \approx 1$. On peut procéder par **essais–erreurs** pour observer le **comportement** du SCN : s'il ne parvient pas à stabiliser de vrais clusters, c'est peut-être que τ est trop grand ; s'il sature de liens moyennement forts, τ est sans doute trop petit.

On peut aussi implémenter une **adaptation** de τ au fil du temps, de la même manière qu'on adapte parfois η. L'idée est de démarrer avec une certaine valeur, puis de **l'augmenter** ou de **la diminuer** suivant l'état du réseau. Par exemple, si on mesure la **densité** de liaisons $\omega_{i,j}$ dépassant un certain seuil, et que cette densité est jugée trop grande, on peut relever τ pour forcer la décroissance de nombreux liens. Inversement, si le réseau paraît trop éteint (trop de liens à zéro), on peut diminuer τ pour encourager la formation de clusters.

C. Implication sur la Formation des Clusters

En choisissant judicieusement τ, on parvient à un **mode** où les liaisons très synergiques se renforcent, tandis que les liaisons moyennes ou peu pertinentes chutent vers zéro. Cela produit des **clusters** plus nets, car seules les entités manifestant une synergie **réellement** significative conservent un lien. Au contraire, un τ inadapté peut donner un réseau flou (multitude de liens moyens) ou un réseau trop fragmenté (clusters entravés par la décroissance trop forte).

De plus, τ interagit souvent avec η, le taux d'apprentissage. Un η élevé couplé à un τ petit peut pousser certaines liaisons à monter rapidement et rester en place, tandis qu'un τ grand contrarie cette montée. Dans la pratique, l'harmonie entre η et τ s'obtient via des heuristiques et une observation empirique de la **convergence**.

Conclusion (9.7.1.2)

Le facteur τ dans l'équation

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

agit comme un **levier** capital de l'**équilibre** dans un **SCN** : un τ trop élevé "écrase" les liaisons, empêchant la constitution de clusters robustes, tandis qu'un τ trop faible "laisse tout flotter",

prolongeant les liens moyennement utiles et risquant de brouiller la distinction entre différents groupes d'entités. Trouver un **juste milieu** ou moduler dynamiquement τ est donc essentiel pour réussir une **auto-organisation** sélective et stable dans la durée. Un calibrage correct procure la "bonne dose" de **décroissance** : on supprime ce qui n'est pas soutenu par une synergie forte, mais on laisse s'établir assez solidement les liaisons porteuses de clusters significatifs.

9.7.1.3. Automatisation via un indicateur de "changement de distribution"

Dans un **SCN** (*Synergistic Connection Network*) confronté à de **l'apprentissage continu** ou à des flux en **évolution permanente**, l'une des problématiques majeures consiste à **détecter** quand la **distribution** des données change. En l'absence de mécanisme de détection automatique, le réseau risque de rester **"figé"** dans un optimum adapté à l'ancienne distribution, alors même que l'environnement s'est transformé (arrivée de nouvelles classes, changements de contextes, etc.). Il est toutefois possible de **quantifier** cette transformation par un indicateur de *concept drift* ou de *distribution shift*, puis de déclencher automatiquement les **réactions** appropriées au sein du SCN (réinitialisation partielle, réajustement de η, recuit local, etc.).

A. Principes Généraux du "Changement de Distribution"

Un **changement de distribution** se produit lorsqu'une **loi** statistique $p(\mathbf{x})$ (ou $p(\mathbf{x}, y)$ dans un contexte supervisé) se modifie. Dans un **SCN**, l'**entraînement** repose sur des entités $\mathcal{E}_i$ dont les synergies $S(i,j)$ peuvent varier si de nouveaux motifs, de nouveaux signaux, ou de nouvelles caractéristiques apparaissent. On parle alors de *data shift*, *concept drift* ou *changement de distribution*.

Sur le plan **mathématique**, on peut imaginer qu'avant un certain instant t_0, la distribution des données restait stable $p_t(\mathbf{x}) \approx p_{t_0}(\mathbf{x})$. Puis, au moment d'un changement, la distribution $p_t(\mathbf{x})$ diverge nettement. Dans un **SCN**, de nombreux liens $\omega_{i,j}$ (antérieurement consolidés) risquent de devenir obsolètes, tandis qu'il faudrait en établir de nouveaux pour incorporer les entités ou classes inédites.

Sans **mécanisme** d'identification d'un tel changement, la mise à jour DSL ordinaire

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

peut se révéler trop lente pour corriger l'ancienneté des liens ou pour initier la réorganisation plus profonde nécessaire à l'apparition de nouveaux **clusters**.

B. Automatisation : Indicateur de "Changement de Distribution"

Pour **automatiser** la détection de shift, on définit un **indicateur** $\Delta(t)$ se basant sur la **dissimilarité** entre la distribution courante et la distribution **historique** (ou de référence). Dès que $\Delta(t)$ franchit un seuil θ_{change}, on en déduit l'existence d'un **changement significatif**.

Si chaque entité $\mathcal{E}_i$ ou chaque "donnée" est décrite par un **vecteur** $\mathbf{x} \in \mathbb{R}^d$, on peut estimer, sur une **fenêtre** glissante de taille ℓ, la distribution empirique $\hat{p}_t(\mathbf{x})$. On compare alors $\hat{p}_t$ à une distribution de référence $\hat{p}_{\text{ref}}$, souvent celle calculée lors d'une phase antérieure stable (ou glissante). Plusieurs mesures de divergence peuvent être employées :

- **Kullback–Leibler** : $\mathrm{KL}(\hat{p}_t \parallel \hat{p}_{\text{ref}})$

- **Jensen–Shannon**
- **Kolmogorov–Smirnov**
- **Page–Hinkley**
- **ADWIN**

ou d'autres tests de détection de drift (approches sur l'entropie, sur la statistique des moyennes, etc.).
On agrège cette dissimilarité en un **scalaire** $\Delta(t) \geq 0$. S'il **décolle** soudainement pour valoir $\Delta(t) > \theta_{\text{change}}$, on estime qu'un "changement de distribution" est en cours.

Une fois la **détection** confirmée, on peut engager différentes actions :

- **Relancer** un recuit ou un "shake" stochastique : on injecte du **bruit** ou une **température** plus élevée dans les pondérations, permettant au réseau de s'écarter de l'optimum ancien.
- **Réinitialiser** partiellement les liens $\omega_{i,j}$: on efface ou atténue les liaisons trop marquées par l'ancienne distribution, facilitant l'accueil de nouvelles entités.
- **Adapter** les **paramètres** η, τ, γ : par exemple, relever η pour gagner en plasticité, augmenter τ pour resserrer la sélection des liens, ou jouer sur l'inhibition.

Ce cycle agit comme un "**macro**-mécanisme" supervisant la **dynamique locale** (cf. section 9.6.2), assurant que le SCN puisse sortir ponctuellement de sa zone de confort et "réapprendre" plus intensément au moment opportun.

Il est possible de baser l'indicateur $\Delta(t)$ non pas sur la distribution **externe x**, mais sur la **configuration** même de $\{\omega_{i,j}\}$. Par exemple, en suivant l'**entropie** ou la **densité** du graphe, on peut repérer qu'un certain "pattern" (ω se concentrant ou se diluant) dévoile un changement de régime. Si cette mesure dépasse un seuil, on soupçonne un shift dans la structure sous-jacente.

C. Avantages et Mise en Œuvre dans un SCN

En incorporant un **indicateur** $\Delta(t)$, on rend le **SCN** plus **autonome**. Au lieu de demander à un opérateur humain de deviner quand le flux a muté, le réseau s'auto-diagnostique et déclenche l'évolution de ses paramètres ou une phase de réorganisation plus marquée.

Cela apporte plusieurs bénéfices :

Adaptation automatique : Le réseau ne reste pas figé, il sait qu'un nouvel environnement exige une adaptation plus rapide.

Convergence plus fine : Plutôt que de maintenir en permanence un recuit ou un bruit élevé (ce qui coûte en calcul et en instabilité), on ne l'active que lorsque les données en font la demande (concept drift détecté).

Exemple algorithmique : À chaque pas de temps t, on effectue la mise à jour locale $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$. En parallèle, on calcule $\Delta(t)$=distance$(\hat{p}_t, \hat{p}_{\text{ref}})$. Si $\Delta(t) > \theta_{\text{change}}$, on augmente η ou relance un recuit local, etc.

D. Perspectives et Limites

Le choix d'un **indicateur** $\Delta(t)$ fiable et la **détermination** d'un bon seuil θ_{change} ne sont pas triviales : un seuil trop faible provoquerait un déclenchement **excessif** des réorganisations (nuisant à la stabilité), tandis qu'un seuil trop élevé pourrait laisser passer un drift majeur non détecté, aboutissant à un "catastrophic forgetting".

Néanmoins, sur le plan **mathématique**, cet indicateur de "changement de distribution" formalise un contrôle du *concept drift*. Dès lors qu'on observe un écart statistique notable, on "libère" de la plasticité dans l'espace $\{\omega_{i,j}\}$, autrement dit on permet un réapprentissage ou un rééquilibrage. Cela garantit que le **SCN** ne se sclérose pas sur un ancien optimum, et qu'il demeure réactif lorsqu'un véritable saut de distribution se produit.

Conclusion (9.7.1.3)

En **apprentissage continu**, l'insertion d'un **indicateur** $\Delta(t)$ de "changement de distribution" au sein d'un **SCN** apporte un mécanisme **automatique** pour déclencher une reconfiguration plus profonde dès que les données entrantes s'écartent fortement de l'historique. Les gains sont clairs : on évite un recuit permanent, tout en restant capable de **réagir** lorsqu'un nouveau contexte survient. Du point de vue **mathématique**, il suffit d'établir un test statistique (divergence KL, KS, etc.), d'en suivre la valeur $\Delta(t)$, et de définir un **seuil** θ_{change} pour enclencher des modifications majeures (remise partielle à zéro des liens, augmentation temporaire du learning rate, ou re-activation de l'inhibition). Ce **pilotage** adaptatif soutient la **plasticité** nécessaire à suivre des flux de données évolutifs, tout en préservant la **stabilité** tant qu'aucune perturbation significative n'est détectée.

9.7.2. Mécanismes Inspirés de l'ART (Adaptive Resonance Theory)

L'**Adaptive Resonance Theory** (ART), proposée par Carpenter et Grossberg, se fonde sur l'idée qu'un système d'apprentissage doit gérer un **équilibre** entre la **stabilité** (préserver les catégories apprises) et la **plasticité** (intégrer des nouveautés). Dans le contexte du **DSL** (Deep Synergy Learning), ces mécanismes peuvent être exploités pour améliorer la **dynamique** de formation des clusters, notamment en introduisant un **concept** de "vigilance" ou de seuil d'erreur/dissimilarité : dès lors que la structure naissante ne satisfait plus les critères de cohérence, un "reset" se déclenche, forçant la création ou la reconfiguration de catégories (micro-clusters, macro-clusters).

9.7.2.1. Vigilance : notion de reset quand l'erreur ou la dissimilarité est trop grande

Dans le paradigme de l'**Adaptive Resonance Theory (ART)**, la **vigilance** constitue un dispositif primordial permettant d'éviter qu'un **nouveau** pattern (ou entité) vienne *déformer* une catégorie existante avec laquelle il n'est pas suffisamment compatible. Transposé dans un **SCN** (*Synergistic Connection Network*), ce mécanisme est mis en œuvre par un **reset** conditionnel : si la **dissimilarité** entre l'entité à intégrer et une catégorie déjà formée dépasse un certain niveau, on "abandonne" l'idée d'insérer ce nouveau pattern dans ladite catégorie. Cette logique permet de **protéger** la cohérence des clusters existants et, si besoin, de **créer** de nouvelles catégories lorsque la similarité est jugée trop faible.

L'ART classique introduit un **paramètre** de **vigilance** $\rho \in (0,1]$. Lorsqu'un nouveau pattern $\mathbf{x}$ se présente, on évalue la **similarité** $\mathrm{sim}(\mathbf{x}, \mathbf{C}_k)$ qui caractérise son adéquation à la catégorie $\mathbf{C}_k$. Sur le plan **mathématique**, la vigilance impose :

$$\mathrm{sim}(\mathbf{x}, \mathbf{C}_k) \geq \rho \quad \Rightarrow \quad \text{pas de reset pour la catégorie } k, \quad \text{sinon reset.}$$

Autrement dit, si la similarité est inférieure à ρ, un **reset** est appliqué et la catégorie $\mathbf{C}_k$ cesse d'être une candidate pour accueillir $\mathbf{x}$. Dans ce cas, plusieurs options sont envisageables :

- Identifier une autre catégorie présentant une proximité plus forte.
- Générer une **nouvelle** catégorie adaptée à $\mathbf{x}$.
- Réajuster plus largement la structure des clusters pour optimiser l'organisation globale.

Cette action, dans l'esprit d'un **DSL**, fige la règle selon laquelle une entité beaucoup trop dissemblable ne doit pas "forcer" son inclusion dans un cluster préexistant, préservant ainsi la cohérence d'ensemble.

En ART, la **similarité** sim s'appuie souvent sur un **recouvrement** (ex. $\| \mathbf{x} \wedge \mathbf{C}_k \| / \| \mathbf{x} \|$ en ART-1). Dans un **Deep Synergy Learning**, on peut employer la **synergie** $S(\mathbf{x}, \mathbf{C}_k)$ comme mesure de proximité : si la synergie est élevée, la nouvelle entité $\mathbf{x}$ "s'entend" avec la catégorie $\mathbf{C}_k$. Le paramètre ρ (ou un seuil $\theta_{\text{vigilance}}$) devient un **critère** rigoureux :

$$S(\mathbf{x}, \mathbf{C}_k) \geq \theta_{\text{vigilance}} \quad \Rightarrow \quad \mathbf{x} \text{ peut intégrer } \mathbf{C}_k, \quad \text{sinon reset.}$$

On réinterprète donc la vigilance comme un "garde-fou" qui interdit l'association de l'entité à un cluster déjà existant si la **dissimilarité** (ou l'**erreur**) est trop grande.

Ce mécanisme se manifeste à deux niveaux :

Niveau micro. Quand on insère une **nouvelle** entité $\mathcal{E}_{n+1}$, on regarde la synergie $S(\mathcal{E}_{n+1}, \mathbf{C}_k)$ pour chacun des clusters $\mathbf{C}_k$. Si aucun cluster n'atteint le seuil $\theta_{\text{vigilance}}$, on exécute un reset global pour cette entité et on crée une **nouvelle** catégorie. Cela renforce la **plasticité** du réseau : une entité très éloignée de tout cluster ne vient pas "polluer" un ensemble déjà cohérent.

Niveau macro. Dans certains scénarios, les **macro-clusters** (super-nœuds ou groupes plus vastes) peuvent aussi se voir infliger un reset. Si un nouveau pattern global se présente et qu'il est trop dissemblable de ce macro-cluster, on impose une scission ou on crée un nouveau macro-cluster. Cela assure une **modularité** plus fine du SCN quand le contexte se modifie.

Matériellement, on peut formaliser la vigilance comme une **contrainte** de la forme :

$$S(\mathbf{v}_k, \mathbf{x}) \geq \rho \quad \Rightarrow \quad \mathbf{x} \in \mathbf{C}_k, \quad \text{sinon reset.}$$

où $\mathbf{v}_k$ représente le "prototype" (ou un vecteur caractéristique) du cluster $\mathbf{C}_k$. Le reset opère alors comme un **rejet** explicite, inversant la tendance qui, en l'absence de vigilance, aurait peut-être vu la mise à jour $\Delta\omega$ essayant d'ajuster lentement les liaisons pour incorporer $\mathbf{x}$.

Le "reset" fonde une **autre** approche de l'**auto-organisation** : au lieu de laisser chaque mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$ peiner à corriger une inadéquation, on coupe court dès que la synergie (ou la similarité) ne franchit pas le **seuil** requis. De ce fait, on protège les clusters existants contre une déformation irréversible, et on crée automatiquement de nouveaux regroupements si un pattern ne s'imbrique pas dans la hiérarchie courante.

Le paramètre ρ dicte la **tolérance** : pour une ρ **élevée**, on scinde fréquemment (clusters plus "purs" et spécifiques). Pour une ρ **faible**, on tolère davantage de différences au sein d'un même cluster. Le reset agit donc comme un frein immédiat à toute tentative "d'assimilation" abusive.

Relation avec les Micro-Catégories et le DSL (9.7.2.2 et 9.7.2.3)

Les chapitres qui suivent (9.7.2.2 et 9.7.2.3) détaillent la **formation** de micro-catégories stables et la manière dont ces principes ART (vigilance, reset) se marient avec la boucle DSL. On peut injecter un **terme** de vigilance $\Delta_{\text{vigilance}}$ dans la mise à jour ou imposer une **remise à zéro** explicite de liaisons au sein d'un cluster dont la synergie est trop faible. Le *reset* de vigilance devient alors un **complément** du DSL, améliorant la **plasticité** et évitant un amortissement lent qui forcerait indûment certaines entités à se rallier à des clusters inadaptés.

Conclusion

La **vigilance** est un **concept** essentiel hérité de l'ART qui stipule qu'une **dissimilarité** (ou erreur) trop élevée déclenche un **reset** immédiat, au lieu de corrections incrémentales. Transposé dans un **SCN** et un cadre **DSL**, ce **seuil** ρ ou $\theta_{\text{vigilance}}$ incite la *séparation* immédiate lorsqu'un pattern ne s'inscrit pas dans la **ressemblance** souhaitée. Sur le plan **mathématique**, on instaure un **critère** explicite de validation du nouveau pattern par la synergie, plutôt que de s'en remettre à la seule équation de mise à jour. Le résultat est un **contrôle** plus direct de la **cohérence** interne des clusters et une meilleure **plasticité** quand les entités à inclure se révèlent trop éloignées des concepts déjà acquis.

9.7.2.2. Formation de "micro-catégories" stables, reconfiguration si un nouveau pattern dépasse un seuil

Lorsque l'on fait fonctionner un **SCN** (*Synergistic Connection Network*) en mode **continu**, plusieurs **micro-catégories** (ou petits clusters) se consolident progressivement dans le réseau à force d'itérations successives de la mise à jour des pondérations $\omega_{i,j}$. Par "micro-catégories", on désigne des **sous-ensembles** d'entités (ou de nœuds) possédant entre eux des liaisons ω suffisamment **élevées**, signe d'une forte cohérence interne. Toutefois, si un **nouveau** pattern (c'est-à-dire un regroupement naissant de certaines entités, ou l'insertion d'entités totalement inédites) manifeste une **synergie** particulièrement **haute**, le SCN peut se **reconfigurer** : il arrive alors que certains micro-clusters s'unissent (fusion), se scindent, ou se réagencent pour **accueillir** et intégrer efficacement ce nouveau motif.

A. Idée d'un Seuil pour la (Re)Configuration

Dans un **SCN**, il n'est pas rare qu'au bout d'un certain temps, on observe la **cristallisation** de micro-catégories $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_m\}$. Ces groupes présentent des liaisons internes $\{\omega_{i,j}\}$ fortes, trahissant une synergie $S(i,j)$ élevée. L'on peut parler de "micro-catégories stables" si :

- Les pondérations $\omega_{i,j}$ au sein de chaque $\mathcal{C}_\alpha$ excèdent un **seuil** $\theta_{\text{stabilité}}$, ou
- La somme $\sum_{i,j\in\mathcal{C}_\alpha} \omega_{i,j}$ dépasse une valeur critique, reflétant la compacité de ce sous-groupe.

Ces micro-catégories représentent donc des noyaux de coopération bien établis. Sur le plan **mathématique**, cela signifie que la dynamique DSL (Deep Synergy Learning) a suffisamment

itéré pour renforcer de façon marquée liaisons internes et affaiblir (ou couper) liaisons externes jugées peu synergétiques.

Un **nouveau pattern**, quant à lui, peut survenir par l'introduction de nouvelles entités $\mathcal{E}_{n+1}$, ... ou par la **montée** de synergies inédites entre entités déjà existantes. Si, par exemple, certaines entités $\{k_1, k_2, \dots\}$ voyaient leurs liaisons ω_{k_p,k_q} s'accroître au point de créer un regroupement fort, on pourrait dire qu'un nouveau **cluster** émerge. Pour décider qu'il s'agit d'un **motif** suffisamment "significatif" pour affecter l'ensemble de la structure, on peut imposer un **seuil** θ_{reconfig}. Dès que la "force cumulative" $\Omega(\{k_1, \dots, k_s\})$ – définie par la somme $\sum_{p,q\in\{k.\}} \omega_{p,q}$ – franchit θ_{reconfig}, on enclenche une **reconfiguration** à l'échelle du SCN.

B. Mécanisme de Reconfiguration

Lorsqu'on détecte un **nouveau** pattern dépassant le **seuil** θ_{reconfig}, on procède à des ajustements topologiques dans le SCN :

Si une entité i appartenait autrefois à un micro-cluster $\mathcal{C}_\alpha$, mais qu'elle présente maintenant une synergie plus élevée avec le nouveau regroupement $\mathcal{C}_\beta^{(\text{new})}$, il est logique que $\omega_{i,j}$ pour $(i \in \mathcal{C}_\alpha)$ diminuent, tandis qu'on **renforce** $\omega_{i,j}$ vers $\mathcal{C}_\beta^{(\text{new})}$. En pratique, la mise à jour DSL

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

va naturally **réorienter** les pondérations là où la synergie est forte. Mais on peut accélérer ce processus en y ajoutant des mécanismes d'**inhibition** ou de **recuit** local (cf. Chap. 7) qui aident à déloger une entité de son cluster précédent.

Dans certains cas, le nouveau motif provoque la **fusion** de plusieurs micro-catégories préexistantes. Il se peut en effet que deux petits groupes initialement distingués $\mathcal{C}_\alpha$ et $\mathcal{C}_\beta$ soient finalement unis via ce nouveau pattern $\mathcal{P}$. Inversement, on peut occasionner une **scission** : si l'émergence du pattern $\mathcal{P}$ révèle l'existence de sous-blocs faiblement connectés au sein d'un cluster, la portion la plus cohérente rejoint le nouveau motif, laissant l'autre fraction former un cluster annexe. Sur le plan **mathématique**, cela se traduit par l'analyse des liaisons $\omega_{i,j}$: si l'on décèle que $\omega_{i,j}$ est très élevé à l'intérieur d'un sous-bloc mais très faible entre ce sous-bloc et le reste, c'est un indice qu'une scission est opportune.

Après la fusion ou la scission, le SCN retourne à une **dynamique** plus localisée (réglage itératif des liens) jusqu'à ce qu'un autre seuil critique soit à nouveau franchi.

C. Justification Mathématique et Seuil Critique

Supposons un **nouveau** sous-ensemble $\mathcal{P} = \{k_1, k_2, \dots\}$. Pour en évaluer la pertinence, on définit la "force" interne :

$$\Omega(\mathcal{P}) = \sum_{p,q\,\in\,\mathcal{P}} \omega_{p,q}.$$

Si $\Omega(\mathcal{P})$ dépasse θ_{reconfig}, on décrète que ce regroupement est "assez fort" pour justifier un **réaménagement** du SCN. On se trouve ici dans une logique de "transition de phase" : en deçà du seuil, le motif demeure trop faible pour bousculer la configuration existante ; dès qu'on franchit θ_{reconfig}, on enclenche une réorganisation (fusion, scission, etc.).

Ce paramétrage θ_{reconfig} ne doit être ni trop bas ni trop haut. Avec un seuil trop faible, on risquerait de reconfigurer le SCN trop fréquemment, générant une **instabilité** chronique. Inversement, si θ_{reconfig} est trop élevé, le réseau pourrait s'empêcher d'évoluer, même quand un regroupement de taille moyenne, mais pertinent, se profile.

D. Bénéfices en termes d'Évolution Continue

Dans un **environnement** ou un **flux** de données évolutif, le SCN peut fonctionner sur la durée, accumulant des micro-clusters et s'y tenant, sauf lorsqu'une **innovation** (une synergie marquante entre entités) se révèle. Cette approche réduit la nécessité d'une réinitialisation totale (on ne jette pas l'ancienne structure, on l'ajuste partiellement), ce qui est bénéfique pour la **stabilité**. En même temps, le fait de décréter une reconfiguration dès qu'un seuil est franchi garantit la **plasticité** : on ne néglige pas un nouveau motif potentiellement crucial.

Le fait de disposer d'un **seuil** θ_{reconfig} évite une infinité de petits "clusters éphémères" et assure que seuls les regroupements non négligeables en termes de **pondérations** se transforment en modifications structurelles. Enfin, on peut imaginer des mécanismes où la somme $\Omega(\mathcal{P})$ doit franchir θ_{reconfig} pendant un certain nombre d'itérations consécutives, éliminant l'effet d'un simple pic passager (événement fugitif).

Conclusion (9.7.2.2)

La **formation** de micro-catégories stables et la **reconfiguration** en présence d'un **nouveau** pattern franchissant un certain **seuil** de synergie θ_{reconfig} démontrent la capacité d'un **SCN** à conjuguer **stabilité** et **évolution**. Les micro-catégories se consolident au fil du temps, mais restent susceptibles de se **réarranger** si une structure plus forte fait son apparition. Sur le plan **mathématique**, cela s'interprète comme un ajustement guidé par la somme Ω des liaisons internes, qui, une fois au-dessus du seuil critique, appelle à une refonte locale de la topologie. Cette combinaison de fixité et de plasticité est particulièrement recherchée dans le cadre d'un **apprentissage continu** (Chap. 9), où l'objectif est de préserver les acquis sans enfermer le réseau dans une structure incapable d'accueillir efficacement les stimuli ou entités futures.

9.7.2.3. Intégration de ces idées dans la boucle DSL

Le **Deep Synergy Learning (DSL)** repose sur une **boucle de mise à jour** itérative où, à chaque pas, on réajuste les **pondérations** $\omega_{i,j}$ en fonction d'une **synergie** $S(i,j)$. Dans sa forme la plus simple, cette boucle consiste, pour chaque liaison (i,j), à appliquer :

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

parfois agrémentée de mécanismes d'**inhibition** ou de **saturation**. Les sections précédentes (§9.7.1 et §9.7.2.1–2) ont détaillé différents **modules** pouvant aider à surmonter les écueils d'une simple descente locale : *recuit simulé*, *inhibition compétitive*, *vigilance/"reset"*, *reconfiguration de clusters* quand un nouveau pattern franchit un **seuil** critique, etc. Le propos de ce chapitre (9.7.2.3) est de montrer **comment** intégrer **conjointement** ces divers modules **dans** la boucle DSL, pour aboutir à un **SCN** plus complet et plus robuste.

A. Rappels de la Boucle DSL

La boucle DSL repose sur un **algorithme** qui, à chaque itération, "parcourt" (de manière synchrone ou asynchrone) l'ensemble des paires (i, j), et met à jour $\omega_{i,j}$ en se basant sur la **synergie** $S(i, j)$. Le **cœur** de cette itération locale est :

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

où η et τ encadrent respectivement la vitesse d'apprentissage et la **décroissance**. De plus, l'on sait que :

- Sans **mécanismes** de "grands sauts" ou de perturbations (ex. recuit), la boucle peut se bloquer dans des **minima locaux** peu satisfaisants.
- Sans **hiérarchie** ou procédure de vigilance/"reset", le réseau peut se **surcharger** de liaisons médiocres ou retomber dans un cluster unique.

Pour remédier à cela, on intègre au sein de cette **boucle** les méthodes d'**inhibition**, de **vigilance**, de **recuit**, etc.

B. Ajout du Recuit Simulé Directement dans la Mise à Jour

Le **recuit simulé** (chap. 7.3) agit comme une **perturbation** aléatoire contrôlée, aidant à échapper aux minima locaux. Concrètement, on rajoute un **terme** de bruit $\xi_{i,j}(t)$ à chaque mise à jour :

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

où $\xi_{i,j}(t)$ suit une **distribution** centrée (typiquement gaussienne) et d'**amplitude** $\sigma(t)$ régulée par une "température" $T(t)$ qui **décroît** au fil des itérations. Ainsi, la boucle DSL demeure la même, mais se voit augmentée par un facteur **stochastique** qui autorise des "sauts" fortuits. On peut écrire :

$$\xi_{i,j}(t) \sim \sigma(t)\,\mathcal{N}(0,1) \quad \text{et} \quad \sigma(t) \approx T(t).$$

Au début (t petit, T élevé), les modifications denses favorisent l'exploration de différentes configurations. Ensuite, quand $T \to 0$, le réseau se stabilise, limitant les perturbations.

C. Couplage Inhibition et Saturation dans la Boucle

On peut également incorporer **l'inhibition compétitive** (p. ex. un terme $-\gamma \sum_{k\neq j} \omega_{i,k}\,(t)$) ou la **saturation** (clampage $\omega_{\max}$) **directement** dans la mise à jour :

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

Ce faisant, on intègre simultanément la **compétition** (éviter trop de liens forts autour d'une même entité), la **pénalisation** d'amplitude (éviter que $\omega_{i,j}$ n'explose) et la **perturbation** stochastique (recuit). Chaque itération **combine** donc tous ces ajustements, aboutissant à une **dynamique** plus riche et plus stable.

D. Intégration du Feedback Hiérarchique

Si l'on gère plusieurs **niveaux** (micro, méso, macro) dans un SCN (voir §9.7.2.1–2 pour la vigilance ou la reconfiguration macro), on peut autoriser un "**feedback descendant**" $\Delta_{\text{down}}(i,j)$ reflétant des consignes de niveau supérieur. Dans la boucle DSL, la mise à jour devient :

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

où chaque **terme** incarne un mécanisme distinct :

- $\eta\big[S(i,j) - \tau\,\omega_{i,j}\big]$: **DSL** de base.
- $\Delta_{\text{down}}(i,j)$: **influence** du niveau macro (verrouillage d'un cluster, étiquetage global, etc.).
- $\xi_{i,j}(t)$: **recuit** (bruit aléatoire décroissant).
- $-\gamma \sum_k \omega_{i,k}$: **inhibition compétitive**.

Après ce calcul, on peut encore "clipper" $\omega_{i,j}$ si nécessaire. On dispose ainsi d'une **unique** boucle itérative englobant tous les **modules** présentés.

E. Conclusion sur l'Intégration dans la Boucle DSL

La **grande** force du DSL réside dans le fait qu'il s'agit d'une **mise à jour locale** sur chaque $\omega_{i,j}$. Pour atteindre une **auto-organisation** plus complète, on peut insérer :

- Un **recuit simulé** (terme stochastique) afin de sortir de minima locaux,
- Un **terme** d'**inhibition** (compétition) et/ou de **saturation**,
- Une **vigilance** (cf. 9.7.2.1) pour déclencher des *resets* immédiats,
- Un **mécanisme** de reconfiguration (cf. 9.7.2.2) lorsqu'un nouveau cluster franchit un **seuil** critique,
- Un **feedback hiérarchique** depuis un niveau macro.

Chaque composant se **greffe** dans la **même** itération, sous forme de termes Δ que l'on additionne à la mise à jour. Sur le plan **mathématique**, la règle itérative ressemble alors à :

$$\begin{aligned}\omega_{i,j}(t+1) = \omega_{i,j}(t) &+ \Delta_{\text{DSL}}(i,j) + \Delta_{\text{inhib}}(i,j) + \Delta_{\text{vigilance}}(i,j) + \Delta_{\text{reconfig}}(i,j) \\ &+ \Delta_{\text{down}}(i,j) + \Delta_{\text{stoch}}(i,j),\end{aligned}$$

où chaque $\Delta_{\cdot}$ correspond à un **module**. L'ensemble constitue une **boucle** unique, mis à jour éventuellement de façon "bloc par bloc" ou paires (i,j) par paires (i,j).

Synthèse : on aboutit à un **SCN** évolutif, capable de **s'auto-organiser** plus finement. Les micro-catégories stables restent protégées, mais s'il survient un **nouveau** pattern (seuil dépassé), la structure se reconfigure. Si le réseau se retrouve piégé, le recuit stochastique l'aide à s'échapper. Si un niveau **macro** détecte un problème, il corrige via la boucle hiérarchique. Cette **intégration** rend le DSL particulièrement flexible et apte à gérer des flux continus ou des

environnements changeants, en fusionnant ces **idées** directement dans la **mise à jour** itérative des pondérations $\omega_{i,j}$.

9.7.3. Auto-Tuning Inhibition ou Recuit

Dans la continuité des techniques d'optimisation et d'adaptation (recuit simulé, inhibition dynamique), la possibilité d'un **auto-tuning** — c'est-à-dire d'un réglage automatique — est essentielle lorsqu'on désire conserver un **SCN** (Synergistic Connection Network) flexible et efficace à long terme. L'idée est de **surveiller** certains **indicateurs** du réseau (par exemple, la densité de liens forts) et d'**ajuster** dynamiquement des paramètres clés comme γ (inhibition) ou la **température** d'un recuit (amplitude du bruit). Cette approche permet de réagir rapidement à une **crise** de surdensité, ou à un **blocage** dans la configuration, sans devoir interrompre l'apprentissage pour opérer un recalibrage manuel.

9.7.3.1. Surveiller la densité des liens : si trop de liens forts se forment, augmenter l'inhibition γ

Dans la **formule** de mise à jour d'un **SCN** (*Synergistic Connection Network*) munie d'un terme d'**inhibition**, on rencontre souvent une composante $-\gamma \sum_{k\neq j} \omega_{i,k}$ qui limite la prolifération de liens forts autour d'une même entité. Toutefois, il n'est pas trivial de **choisir** à l'avance la bonne valeur de γ. Une γ trop faible laissera se constituer un réseau **surdensifié** de liaisons, tandis qu'une γ trop forte bridera l'émergence de clusters. L'idée décrite ci-dessous est d'opérer un **auto-tuning** de γ : on **surveille** un **indicateur** global de la densité des liaisons et on **régule** γ en conséquence.

A. Mécanique de l'Auto-Tuning sur γ

Lorsque l'on dispose d'une mise à jour de type

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

la valeur γ dicte l'ampleur de la **compétition** : plus γ est grand, plus la liaison $\omega_{i,j}$ est freiné par l'accumulation d'autres liens $\omega_{i,k}$. Généralement, cette γ est fixée. Mais on peut l'**auto-régler** en **surveillant** si le réseau se remplit de trop de liens forts.

On définit un **indicateur** $\delta(t)$ qui mesure la **proportion** de liaisons $\omega_{i,j}$ excédant un certain **seuil** θ_{link}. Concrètement :

$$\delta(t) = \frac{1}{n(n-1)} \sum_{i\neq j} \mathbf{1}\left(\omega_{i,j}(t) > \theta_{\text{link}}\right),$$

où $\mathbf{1}(\cdot)$ vaut 1 si la condition est vraie, sinon 0. Ce $\delta(t) \in [0,1]$ reflète la **densité** de liens forts dans le SCN. Si $\delta(t)$ est grand, cela signifie que beaucoup de liaisons $\omega_{i,j}$ dépassent θ_{link}, signe d'un réseau très connect. À l'inverse, un $\delta(t)$ faible indique un graphe clairsemé.

Au fil des itérations, on **adapte** γ en fonction de $\delta(t)$:

$$\gamma(t+1) = \gamma(t) + \eta_\gamma [\delta(t) - \delta_{\text{cible}}],$$

où δ_{cible} est la fraction cible de liens forts qu'on juge acceptable, et $\eta_\gamma > 0$ est un pas de mise à jour. Si $\delta(t)$ dépasse δ_{cible}, alors $\gamma(t+1) > \gamma(t)$, on **accroît** l'inhibition pour limiter la poussée de liens forts. Si au contraire la densité $\delta(t)$ se révèle trop basse ($\delta(t) < \delta_{\text{cible}}$), on diminue γ, autorisant la consolidation de liaisons.

Ce mécanisme libère l'**opérateur** ou le **concepteur** de la tâche de devoir choisir "à la main" un γ fixe :

- **Équilibre automatique**

  Le SCN se **stabilise** autour d'une densité voulue δ_{cible}, évitant la prolifération incontrôlée des liens ou, au contraire, un étouffement complet.

- **Réponse adaptative**

  Quand un événement (ex. arrivée de nouvelles entités, renforcement global des liaisons) fait monter $\delta(t)$, la **compétition** augmente pour restaurer l'équilibre. Inversement, si le réseau devient trop vide, γ baisse, permettant la réémergence de connexions nécessaires.

Sur le plan **mathématique**, cela se formalise comme un **système dynamique** couplant $\{\omega_{i,j}(t)\}$ et $\gamma(t)$. L'une influe sur l'autre : la mise à jour locale sur ω dépend de γ, tandis que l'évolution de γ réagit à la densité δ (qui elle-même provient des ω).

C. Application en Continu

Lors d'un **apprentissage continu** (§9.6), l'introduction de nombreuses entités ou de nouveaux patterns peut faire **exploser** la proportion de liaisons fortes : le SCN découvre de multiples synergies et $\delta(t)$ grimpe. Le **terme** auto-régulé :

$$\gamma(t+1) = \gamma(t) + \eta_\gamma[\delta(t) - \delta_{\text{cible}}]$$

assure que, par rétroaction, γ monte quand δ excède la cible. Cela **maintient** une topologie moyennement éparse. D'autres scénarios (changement de distribution, reconfiguration) peuvent faire chuter δ. La formule ramène alors γ vers des valeurs plus basses, rendant la compétition moins aggressive, pour **permettre** au réseau de reformer des clusters.

D. Brève Esquisse Analytique

Si l'on modélise $\delta(t)$ comme une fonction **croissante** des pondérations $\omega_{i,j}$ et qu'en parallèle, la dynamique de $\omega_{i,j}$ dépend elle-même de γ, on obtient un système de la forme :

$$\begin{cases} \omega_{i,j}(t+1) = F\left(\omega_{i,j}(t),\, \gamma(t)\right), \\ \gamma(t+1) = \gamma(t) + \eta_\gamma \left[\delta\left(\omega_{i,j}(t)\right) - \delta_{\text{cible}}\right]. \end{cases}$$

Il est souvent possible de montrer (sous hypothèses de **régularité**) qu'il existe un **point fixe** (ω^*, γ^*) vérifiant $\delta(\omega^*) = \delta_{\text{cible}}$, garantissant un **niveau** de densité cible à l'équilibre. Les détails dépendent de la **forme** précise de la règle DSL et du calcul de δ.

Conclusion

La **surveillance** de la densité des liens et l'**ajustement** progressif de γ pour maintenir $\delta(t)$ proche d'une valeur cible δ_{cible} constituent un **mécanisme** efficace d'**auto-régulation** dans un

SCN. Sur le plan **mathématique**, on couple la règle DSL (mise à jour locale) avec une **équation** supplémentaire pilotant γ. On obtient ainsi :

- **Prévention** d'un réseau trop dense (on augmente γ dès que la densité s'emballe).
- **Évitement** d'une trop forte inhibition (on réduit γ si la densité devient trop faible).

Cette boucle d'**auto-tuning** rend le SCN plus **plastique** : il peut accueillir des phases de forte synergie tout en évitant la prolifération globale de liaisons, et inversement il sait relâcher la compétition lorsque le réseau se trouve trop épars. Couplée à d'autres techniques (recuit, vigilance, mécanismes hiérarchiques), elle contribue à un **apprentissage continu** fluide et stable.

9.7.3.2. Injection de bruit ponctuelle si l'on détecte un "enlisement"

Dans un **SCN** (*Synergistic Connection Network*) soumis aux mises à jour répétées de la logique **DSL**, on observe parfois que la **configuration** des pondérations $\{\omega_{i,j}\}$ se stabilise de manière prématurée. Le réseau ne bouge plus que très faiblement, donnant l'impression d'avoir atteint un **état d'équilibre** local, alors même qu'il pourrait exister une configuration plus favorable (par exemple, un meilleur découpage en clusters). Ce phénomène d'**enlisement** ou de blocage dans un *minimum local* signale qu'on ne bénéficie plus d'assez de **plasticité** pour franchir la barrière d'énergie (ou de synergie) séparant l'état actuel d'une organisation potentiellement meilleure.

Une **stratégie** pour résoudre ce problème consiste à **injecter** périodiquement ou ponctuellement un **bruit** sur un certain sous-ensemble de liaisons $\omega_{i,j}$. Contrairement au recuit simulé continu (qui persiste à chaque itération), la présente approche "par salves" n'est déclenchée qu'en cas de **détection** de stagnation ou de progrès insuffisant. Cette perturbation peut "**débloquer**" la configuration en causant une **reconfiguration** plus ample, sortie du *puits* local.

A. Principes Généraux de l'Enlisement

Un **enlisement** se produit quand la dynamique :

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

ou dans sa version plus complète (avec inhibition, etc.), **s'estompte** graduellement et que la somme des variations $\sum_{i,j} \left|\Delta\omega_{i,j}(t)\right|$ tend à devenir négligeable. On se retrouve dans un état d'**équilibre local** où la topologie formée (clusters, liens forts vs. faibles) n'évolue quasiment plus. Du point de vue **mathématique**, cela correspond à un *minimum local* de la "fonction d'énergie" $\mathcal{J}$ ou un *attracteur* stable, mais **pas** forcément celui qui procure la meilleure organisation pour le SCN.

Il existe plusieurs **indicateurs** pratiques pour déceler un enlisement :

- **Énergie** $\mathcal{J}$: son évolution $\Delta\mathcal{J}(t)$ peut devenir nulle ou quasi nulle pendant un certain nombre d'itérations.
- **Variation des liens** : $\sum_{i,j} \left|\omega_{i,j}(t+1) - \omega_{i,j}(t)\right|$ devient très faible ou stationnaire.
- **Score** (p. ex. modularité, qualité de cluster) stagne alors qu'un meilleur score est théoriquement possible (information externe, heuristique, etc.).

B. Nature de l'Injection de Bruit Ponctuelle

Au lieu de maintenir un recuit simulé dans le cœur de la boucle DSL, on peut décider qu'à chaque **détection** d'enlisement, on ajoute un **terme** aléatoire à un nombre restreint de liaisons :

$$\omega_{i,j}(t+1) \leftarrow \omega_{i,j}(t+1) + \beta_{i,j},$$

où $\beta_{i,j}$ est un **bruit** de distribution centrée (gaussienne, uniforme, etc.). L'amplitude $\| \beta_{i,j} \|$ est un paramètre crucial : un bruit trop faible ne débloquera pas la structure figée, tandis qu'un bruit trop élevé peut casser les configurations déjà cohérentes.

Il n'est pas nécessaire de perturber **toutes** les liaisons. On peut cibler par exemple :

- Les **liens inactifs** $\omega_{i,j} \approx 0$ pour leur donner l'opportunité de "s'élever" si la synergie y est en fait sous-exploitée.
- Les **liens moyens** proches d'un certain seuil, afin de les pousser soit vers la montée soit vers la chute, clarifiant la situation.
- Un **échantillon** aléatoire représentant une fraction fixe (10 %, 20 %) des liaisons.

On surveille l'état de stagnation. Dès qu'on constate que le réseau est "enlisé" (selon un critère $\Delta\mathcal{J} \approx 0$ pendant K itérations, par exemple), on exécute ce "**sursaut**" ponctuel. Cela rappelle la notion de "shake" dans certains algorithmes de recherche locale, où l'on secoue ponctuellement la configuration pour tenter d'atteindre un **meilleur** bassin d'attraction.

C. Expression Mathématique d'un "Sursaut" Ponctuel

Considérons le DSL standard :

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

Lorsqu'on détecte le temps t_0 comme **instant** de blocage, on applique juste après la mise à jour DSL :

$$\omega_{i,j}(t_0+1) \leftarrow \omega_{i,j}(t_0+1) + \beta_{i,j}, \quad \forall (i,j) \in \mathcal{S}.$$

Typiquement, $\beta_{i,j} \sim \mathcal{N}(0, \sigma^2)$ ou $\mathrm{Uniform}[-\sigma, \sigma]$. On peut moduler σ selon la durée du blocage : plus le réseau reste figé, plus on injecte un bruit grand.

D. Avantages et Limites

Les avantages de cette approche résident dans son **économie** d'intervention, puisqu'elle ne perturbe pas en continu, mais uniquement lorsqu'un blocage survient, évitant ainsi un recuit constant. Elle permet également un **ciblage** précis des liaisons à secouer, préservant ainsi les clusters stables et n'agissant que là où une amélioration est possible. Enfin, elle assure le **maintien** de la dynamique locale DSL pendant la majeure partie du temps, tout en offrant un mécanisme ponctuel permettant d'échapper aux minima.

Toutefois, certaines **limites** doivent être prises en compte. L'**efficacité** de cette méthode peut être incertaine lorsque l'enlisement est profond, un seul sursaut pouvant s'avérer insuffisant, nécessitant alors plusieurs injections et se rapprochant ainsi d'un recuit par paliers. Le **calibrage** du bruit représente un défi, car la sélection d'un paramètre comme σ ou la proportion de liens ciblés est délicate : un bruit trop faible risque de rester inefficace, tandis qu'un bruit excessif

pourrait compromettre la structure acquise. Enfin, la **détection** du blocage repose sur la définition d'un critère précis. Sans un indicateur fiable, on risque soit d'intervenir trop fréquemment, soit de ne pas agir suffisamment.

Conclusion

L'injection **ponctuelle** de bruit sur les liaisons $\omega_{i,j}$ en cas d'enlisement constitue un **compromis** intéressant entre un recuit permanent et une descente purement locale. **Mathématiquement**, il s'agit de repérer un temps t_0 où le SCN stagne, puis d'ajouter un **terme** aléatoire $\beta_{i,j}$ à certains liens ω. Cette action "secoue" la structure, brisant la rigidité éventuelle pour, potentiellement, dénicher une organisation plus adéquate. Un soin particulier doit être porté au **critère** de détection (comment savoir que l'on est "figé") et à l'**intensité** du bruit. Malgré ces nuances, cette technique demeure souvent efficace pour sortir d'un *puits local* et raviver la **plasticité** lorsque la descente DSL bute sur un optimum peu satisfaisant.

9.7.3.3. Comparaison de schémas adaptatifs vs. fixes

Dans les procédures de **recuit simulé** ou d'**injection de bruit** contrôlée, la **planification** de la température (ou plus généralement l'amplitude du **terme stochastique**) est cruciale pour l'**efficacité** de la recherche d'un optimum ou pour échapper à des minima locaux. On distingue généralement deux catégories de stratégies. Les schémas **fixes** définissent un planning de température (ou bruit) *a priori* et n'y dérogent pas ; les schémas **adaptatifs** ajustent au contraire la température en cours de route, d'après des **indicateurs** ou des **critères** liés à l'évolution courante. Cette section compare leurs forces et leurs limites, et discute de leur mise en œuvre dans un **SCN** (*Synergistic Connection Network*).

A. Rappels sur la Notion de Température

Dans un **recuit simulé**, la **température** $T(t)$ détermine l'**ampleur** du bruit ajouté à chaque itération, par exemple :

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

où $\xi_{i,j}(t)$ est un **terme stochastique** (gaussien, uniforme, etc.) dont l'**écart-type** ou l'**amplitude** dépend de $T(t)$. Une **température** élevée encourage des sauts importants, facilitant l'exploration de configurations variées. Quand la température se réduit, le système se "fige" plus facilement dans une configuration, cherchant la convergence. L'ensemble forme une **descente localement perturbée** qui, à terme, espère se rapprocher d'un **minimum** plus global que celui obtenu par une descente purement déterministe.

B. Schémas à Température Fixe ou à Décrément Programmé

Lois Fixes :
Un **schéma fixe** définit d'emblée $T(t)$ selon une **loi** invariable, *indépendamment* de l'état actuel du système. Les plans classiques incluent :

- Exponentiel :

$$T(t) = T_0\,\alpha^t, \quad \alpha < 1.$$

- Logarithmique :

$$T(t) = \frac{T_0}{\log(t + C)},$$

- Polynomiaux :

$$T(t) = \frac{T_0}{(1 + \beta t)^p}.$$

Ces formules ne **changent** pas leur rythme de refroidissement, même si le système est encore bloqué ou n'a pas fini d'explorer.

Ils sont simples à implémenter et à analyser. Certaines lois (exponentielle lente, logarithmique) bénéficient de **preuves** de convergence (dans des contextes classiques de recuit). Leur régularité et l'absence de conditions sur l'état interne rendent la démonstration mathématique plus aisée.

Ils sont **rigides**. Si la température diminue trop vite, on fige le SCN avant d'avoir pu échapper aux minima. Si elle baisse trop lentement, on perd beaucoup de temps à fluctuer inutilement. De plus, on ne peut pas "réchauffer" le système si l'on détecte tardivement un blocage — le planning ne prévoit pas de remonter T.

C. Schémas Adaptatifs : Ajustement Dynamique de $T(t)$

Contrairement aux approches fixes, un schéma **adaptatif** monitore l'évolution courante (acceptation de sauts, stagnation, etc.) et **ajuste** $T(t)$ en fonction. Il peut **augmenter** ou **diminuer** la température selon la situation. On cherche à rester dans un régime où la proportion de transitions acceptées ou la vitesse de convergence restent satisfaisantes.

On peut définir une boucle où, après un certain nombre d'itérations, on calcule un **taux d'acceptation** α_t. Si ce taux est trop bas (< 5 %), on se trouve dans un régime trop "froid" ; on **relance** la température de + 10 %. Si ce taux est trop élevé (> 60 %), on refroidit plus vite. On peut de même scruter l'énergie $\mathcal{J}$ ou la variance des pondérations : en cas de **stagnation**, on remonte T.
Mathématiquement, on obtient une loi :

$$T(t + 1) = T(t) + \eta_T[r(t) - r_{\text{cible}}],$$

où $r(t)$ est un indicateur (par ex. le ratio d'acceptation), et r_{cible} un objectif.

Ils donnent de la **flexibilité**. On peut se "réchauffer" si on s'aperçoit d'un blocage local, ou au contraire **refroidir** plus vite si on détecte qu'on oscille trop. Dans un **SCN** potentiellement **non stationnaire** (avec changements de distribution ou de synergie), cela s'avère très utile : un schéma fixe serait moins réactif.

On doit définir des **critères** parfois compliqués pour moduler la température (acceptation, stagnation, etc.), ce qui augmente la complexité d'implémentation. Les **garanties** théoriques de convergence sont moins nettes que pour les lois fixes bien connues en recuit. Dans certains cas, si on "réchauffe" trop fréquemment, on n'atteint pas de vraie stabilisation, demeurant dans un état oscillatoire.

D. Comparaison et Implications dans un SCN

Dans un **SCN**, la mise à jour des pondérations $\omega_{i,j}$ s'effectue sur un **paysage** qui peut lui-même **changer** (ex. arrivée de nouvelles entités, modifications de la synergie). Les schémas de recuit "**fixes**" (exponentiels, logarithmiques) ne tiennent pas compte de ces aléas : si, par exemple, un

nouveau cluster apparaît tardivement, le système peut être trop “refroidi” pour intégrer cette nouveauté. Un schéma **adaptatif** demeure plus réactif, car il peut rehausser la température si une situation non prévue se présente.

Les **points essentiels** à considérer incluent tout d'abord le **temps de convergence**. Les plans fixes offrent une trajectoire prévisible de décroissance de TT, tandis que les plans adaptatifs peuvent entraîner des oscillations. Ensuite, la **robustesse** des approches adaptatives leur permet de s'ajuster aux conditions réelles, évitant ainsi qu’un refroidissement trop rapide ne piège le SCN dans un minimum local ou qu’un refroidissement trop lent ne conduise à un gaspillage de ressources. Enfin, la **complexité** des plans adaptatifs repose sur la nécessité d'un mécanisme de **feedback**, intégrant des critères tels que le taux d'acceptation ou la détection de stagnation, ainsi que sur des paramètres définissant l’intensité du réchauffement ou du refroidissement.

Conclusion

D’un point de vue **mathématique** et pratique, on distingue :

- Les **schémas fixes** pour la température (ou amplitude de bruit), plus faciles à **analyser** et à **implémenter**, mais souvent **rigides** et mal adaptés aux changements contextuels en cours d’apprentissage.

- Les **schémas adaptatifs**, plus **souples** et potentiellement plus **efficaces** quand le SCN doit faire face à des **blocages** tardifs ou à des **flux** de données évolutifs, mais nécessitant un **système de feedback** et un paramétrage soigneux, sans garantie de convergence formelle aussi solide.

En **SCN** — où la synergie, le nombre d’entités, et les paramètres peuvent fluctuer — un schéma *semi-adaptatif* (par exemple, une loi de refroidissement de base modifiable ponctuellement si la stagnation est trop forte) offre un **compromis** : la descente globale conserve une structure relativement simple, tandis que des ajustements ponctuels du “niveau de bruit” évitent les pièges localement, préservant la **plasticité** du réseau face aux aléas de l’apprentissage continu.

9.8. Études de Cas et Applications

Au sein d'un **DSL** (Deep Synergy Learning) évolutif, un des enjeux cruciaux est de **valider** la capacité du **SCN** (Synergistic Connection Network) à gérer des **flux** continus et hétérogènes, tout en conservant une structure de clusters **adaptative**. Dans cette section (9.8), nous illustrons par divers **cas** comment un SCN s'**ajuste** dans des situations où de nouvelles **entités** ou de nouveaux **contextes** se présentent au fil du temps. Nous abordons d'abord (9.8.1) le flux **textuel** continu (documents, paragraphes), puis (9.8.2) la **robotique temps réel**, et enfin (9.8.3) le **streaming multimodal**. Ces scénarios mettent en évidence la **flexibilité** du DSL pour intégrer de nouveaux blocs (images, textes, capteurs) sans réinitialiser la totalité du réseau.

9.8.1. Flux Textuel Continu

Dans de nombreux systèmes d'information (forums, actualités, documentation en ligne), les **paragraphes** ou **documents** arrivent de manière **incrémentale** : on publie un nouvel article, on ajoute un nouveau chapitre, etc. Le **SCN** doit alors :

- **Insérer** ces entités (textuelles) à la volée,
- **Mettre à jour** $\{\omega_{i,j}\}$ localement et
- **Adapter** les clusters ou thèmes qui en résultent, sans effacer complètement l'historique.

9.8.1.1. Nouvelles entités = paragraphes / documents

Dans les systèmes où l'on traite des **textes** (paragraphes, documents, articles…), on peut interpréter chaque nouveau **bloc textuel** qui arrive comme une **nouvelle entité** $\mathcal{E}_{n+1}$ à insérer dans le **SCN** (*Synergistic Connection Network*). L'approche **DSL** (*Deep Synergy Learning*) se prête bien à l'intégration **incrémentale** de contenus textuels, puisqu'à chaque itération on peut facilement adjoindre une nouvelle ligne et colonne à la matrice ω correspondant à la nouvelle entité. On actualise ensuite les pondérations $\omega_{(n+1),j}$ en fonction de la **synergie** $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$, permettant ainsi au paragraphe ou document d'être **associé** (ou non) aux clusters ou thèmes déjà présents dans le réseau.

A. Arrivée Incrémentale de "blocs textuels"

On suppose qu'à chaque **instant** t, un nouveau **paragraphe** ou un nouveau **document** se présente, noté $\mathcal{E}_{n+1}$. Dans une application **continue**, il peut s'agir d'un flux de paragraphes venant d'une source d'actualités, d'un lot progressif de résumés, ou de simples **mises à jour**. Sur le plan **mathématique**, on modifie la **taille** de la matrice ω pour passer de $(n \times n)$ à $(n+1) \times (n+1)$. Pour chaque entité existante $\mathcal{E}_j$ avec $j = 1, \ldots, n$, on calcule la **synergie** :

$$S(\mathcal{E}_{n+1}, \mathcal{E}_j),$$

et on initialise la pondération $\omega_{(n+1),j}$ (et $\omega_{j,(n+1)}$) à une valeur adaptée, souvent très faible ou nulle, avant de la mettre à jour selon la règle DSL. Dans la pratique, pour gérer un grand nombre de documents, on peut se restreindre à un **voisinage** partiel (ex. *k*-NN), évitant ainsi l'explosion quadratique des calculs.

B. Mesure de Synergie Textuelle

Afin de définir **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ dans un contexte **textuel**, on se base volontiers sur des **embeddings** vectoriels. Chaque document ou paragraphe est projeté en un **vecteur** $\mathbf{v}_i \in \mathbb{R}^d$. On calcule alors une **similarité** (par ex. le cosinus) :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \frac{\mathbf{v}_i \cdot \mathbf{v}_j}{\| \mathbf{v}_i \| \, \| \mathbf{v}_j \|}.$$

Plus les vecteurs $\mathbf{v}_i$ et $\mathbf{v}_j$ sont proches, plus la synergie tend vers 1. On peut aussi employer des mesures de **co-occurrence** (mots-clés partagés, intersection de vocabulaire) ou une combinaison de métriques sémantiques. Sur le plan **mathématique**, le SCN se contente alors de reçevoir $S(i, j)$ en entrée, peu importe la méthode précise de calcul.

C. Insertion Locale dans le SCN

Une fois la synergie déterminée pour la **nouvelle** entité, la mise à jour DSL agit par :

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

pour j dans un sous-ensemble $\mathrm{N} \subseteq \{1, \ldots, n\}$. Ainsi, l'entité $\mathcal{E}_{n+1}$ **renforce** ses liens ω avec les documents les plus similaires et **laisse** décroître (ou n'augmente guère) ceux pour lesquels la synergie se révèle faible. Peu à peu, le nouveau document s'**agglomère** à un cluster existant (s'il s'avère sémantiquement proche) ou en forme un nouveau si aucune similarité notable ne l'associe aux ensembles préexistants.

D. Dimension Multi-Échelle

Lorsqu'il existe un **niveau macro** (chap. 6) gérant des **super-nœuds** (thèmes, catégories plus larges), on peut aisément répercuter l'insertion d'un document **au niveau** macro : si le document $\mathcal{E}_{n+1}$ se révèle très proche d'un super-nœud α (ex. "politique"), la pondération $\omega_{(n+1),\alpha}^{(\mathrm{macro})}$ s'accroît, signifiant une appartenance partielle à ce thème. Dans ce schéma, la synergie macro $\Psi(\cdot,\cdot)$ se construit à partir des synergies micro (somme ou moyenne de ω, ou d'autres agrégations), autorisant l'**actualisation** des super-nœuds (thèmes) à chaque insertion de document.

E. Implications sur les Clusters et Thèmes

En introduisant **incrémentalement** de nouveaux paragraphes/documents, le SCN met à jour les clusters existants ou en crée de nouveaux si la **similarité** demeure trop faible. Sur le plan **pratique**, un SCN ainsi modélisé peut suivre en continu l'**évolution** d'un corpus textuel :

- **Apparition de nouveaux thèmes** : si le document $\mathcal{E}_{n+1}$ n'est pas en affinité avec des clusters existants, il demeurera quasi isolé jusqu'à ce que d'autres documents similaires arrivent. Ainsi, un thème émergent se forme graduellement.

- **Maintien de thèmes anciens** : si on ajuste correctement τ, l'inhibition et autres paramètres, des clusters déjà formés ne disparaissent pas trop vite (pour ne pas perdre la mémoire d'anciens thèmes encore pertinents).

- **Adaptation en flux** : tout se fait **sans** réinitialisation complète. On modifie la matrice ω localement chaque fois qu'un nouveau document survient. Sur le plan **mathématique**, la structure du réseau s'agrandit, mais reste gouvernée par la même règle DSL.

Conclusion (9.8.1.1)

Quand on considère des **paragraphes** ou **documents** comme des **entités** successives, la dimension **incrémentale** du **SCN** se révèle très adaptée : chaque nouvelle entité $\mathcal{E}_{n+1}$ alimente une **ligne** et une **colonne** supplémentaires dans la matrice ω. On calcule la **synergie** textuelle (ex. cosinus d'embeddings) pour mettre à jour les pondérations. Peu à peu, le DSL incorpore ce document dans un **cluster** existant ou en **crée** un nouveau si la similarité s'avère trop faible. Le **point fort** de ce modèle est sa **souplesse** : il ne cesse pas de se reconfigurer au fil de l'**arrivée** de nouveaux blocs de texte, maintenant une organisation adaptative et potentiellement multi-niveau (macro-nœuds). D'un point de vue **mathématique**, la matrice ω se **met à jour** localement, évitant un recalcul global coûteux. Sur le plan **algorithmique**, on peut limiter la croissance en ne recalculant la synergie que sur un **voisinage** partiel (ex. k plus proches documents), pour demeurer **scalable** même lorsque le nombre de documents croît.

9.8.1.2. Mise à jour incrémentale de ω. Apparition de nouveaux thèmes

Dans un **SCN** (*Synergistic Connection Network*) opérant en mode **incrémental**, on fait face à l'arrivée progressive de **nouvelles entités** $\mathcal{E}_{n+1}, \mathcal{E}_{n+2}, \ldots$. Dans un contexte textuel (cf. §9.8.1.1), ces entités peuvent être des **paragraphes** ou des **documents** ; dans d'autres contextes, elles peuvent être de nouvelles données sensorielles, de nouvelles classes, etc. Sur le plan **mathématique**, l'**insertion** de l'entité $\mathcal{E}_{\text{new}}$ signifie qu'on **augmente** la taille de la matrice ω (passant de $n \times n$ à $(n+1) \times (n+1)$) et qu'on initialise les liaisons $\omega_{\text{new},j}$ pour $j = 1 \ldots n$. La **dynamique** DSL (Deep Synergy Learning) se charge alors d'intégrer $\mathcal{E}_{\text{new}}$ dans le réseau : si la **synergie** $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ est forte, les pondérations $\omega_{\text{new},j}$ se renforcent ; dans le cas contraire, elles restent faibles ou s'annulent. Peu à peu, une entité très différente des anciennes peut initier un **nouveau** cluster (ou thème).

A. Contexte d'une arrivée de "nouveau thème"

On envisage un flux de données : à chaque pas, le SCN reçoit une entité $\mathcal{E}_{m+1}$. Sur le plan conceptuel, si cette entité se rattache à un **thème** ou "sous-thème" déjà existant, ses liaisons ω monteront vite vers un cluster établi. Mais si elle apporte du **contenu** ou des **caractéristiques** vraiment inédits, il peut s'ensuivre la création d'un *nouveau* "macro-nœud" dans la hiérarchie (un **nouveau thème**). D'un point de vue **DSL**, on calcule $\omega_{\text{new},j}(t+1)$ selon la règle habituelle, permettant de voir se dessiner un cluster jusqu'alors absent.

En d'autres termes, quand $\mathcal{E}_{\text{new}}$ arrive, on met à jour :

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

pour chaque j. Si la **similarité** (ou synergie) $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ est faible pour la quasi-totalité des entités, $\mathcal{E}_{\text{new}}$ reste **isolée**, avant que d'autres entités comparables ne se présentent. Une fois suffisamment de synergies accumulées avec quelques entités, cette entité (et son groupe éventuel) peut donner naissance à un **nouveau** cluster, c.-à-d. un **nouveau thème**.

B. Méthode de Mise à Jour Incrémentale

Lorsqu'une entité $\mathcal{E}_{\text{new}}$ arrive, on crée la ligne/colonne $\omega_{\text{new},j}$, $\omega_{j,\text{new}}$ et on peut les initialiser à 0 ou à un epsilon faible ($\epsilon > 0$). On effectue (ou on approximate) la **synergie** $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ pour des j pertinents (ex. un k-NN en termes de similarité textuelle). Puis, à chaque étape de la dynamique DSL, on applique :

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

et de même pour $\omega_{j,\,\text{new}}(t+1)$. À terme, si certaines de ces liaisons atteignent un niveau non négligeable, $\mathcal{E}_{\text{new}}$ se rattache au cluster que forment les entités $\mathcal{E}_j$ concernées.

Si la somme $\sum_{j\in C}\omega_{\text{new},\,j}$ devient notable au sein d'un cluster C, on décrète que $\mathcal{E}_{\text{new}}$ s'y agrège. Inversement, si cette somme reste minime pour tous les clusters existants, $\mathcal{E}_{\text{new}}$ fonde un **nouveau** cluster, susceptible de représenter un "nouveau thème". Sur le plan **algorithmiquement**, la dynamique DSL se combine aux mécanismes d'**inhibition** ou de **recuit** pour réallouer éventuellement d'autres entités proches vers ce nouveau groupe.

C. Apparition de Nouveaux Thèmes : Dynamique Globale

La venue de plusieurs entités $\mathcal{E}_{\text{new}_1}, \ldots, \mathcal{E}_{\text{new}_m}$ très corrélées (forte similarité mutuelle) peut donner lieu à un **macro-nœud** inexistant auparavant. En effet, si chacune de ces entités affiche un niveau important de synergie S au sein de leur sous-groupe, la dynamique DSL leur attribuera des ω internes élevées, surpassant parfois les liens qu'elles avaient (potentiellement) avec les anciens clusters. Ainsi se **révèle** un nouveau thème. Du point de vue **mathématique**, on observe une consolidation incrémentale : $\Omega(\{\text{new}_1, \ldots \text{new}_m\})$ (somme des liaisons internes) croit, dépassant à un moment donné un seuil θ_{macro}, ce qui autorise la création d'un super-nœud.

Pour des documents textuels, chaque nouveau document arrivé peut très peu ressembler aux documents existants. Au fur et à mesure que des documents analogues affluent, la synergie de ce "nouveau" groupe sémantique grimpe ; on reconnaît alors l'**émergence** d'un nouveau sujet (nouveau "thème"). Le SCN n'a pas eu besoin d'une **reconstruction** globale ; il s'est ajusté localement, par incréments, et a laissé ce nouveau thème "pousser" dans la structure.

D. Aspects Mathématiques Clés

La force de cette approche réside dans le fait qu'on ne **recalcule** pas tout le SCN depuis zéro lorsque $\mathcal{E}_{\text{new}}$ arrive, mais qu'on met à jour localement les liaisons $\omega_{\text{new},\,j}$ pour un sous-ensemble de j. Cela conserve la **scalabilité** (en évitant de toucher $0(n^2)$ paires) et la **continuité** de l'apprentissage.

Une entité qui n'entre pas immédiatement dans un cluster demeure en veille (liens faibles). Si, plus tard, un **effet** de recuit ou un nouveau sous-ensemble apparenté se présente, elle pourra s'agréger ou contribuer à un nouveau macro-nœud. Sur le plan **mathématique**, l'entité ne redevient pas ω-nul par défaut ; elle conserve un embryon de liaisons susceptibles de grandir si la synergie future le justifie.

Conclusion

La **mise à jour incrémentale** de ω constitue un **mécanisme** central pour incorporer de **nouveaux** thèmes ou clusters dans un **SCN**. À chaque arrivée d'entité, on calcule localement $S\big(\mathcal{E}_{\text{new}}, \mathcal{E}_j\big)$ et on applique la règle DSL (et éventuellement d'autres modules comme l'inhibition ou le recuit). Si la synergie demeure trop faible vis-à-vis des clusters existants, l'entité forme une **nouvelle** catégorie ; sinon, elle se **rattache** (en quelques itérations) à un groupe déjà cohérent. De la sorte, le SCN se **déploie** en continu, voyant naître des thèmes inédits dès qu'un véritable **noyau** de forte synergie apparaît. Sur le plan **mathématique**, cette insertion incrémentale s'avère **locale** et **peu coûteuse**, tout en assurant une **structure** d'ensemble qui s'ajuste progressivement aux **flux** de données.

9.8.1.3. Éviter que les anciens thèmes (clusters) disparaissent trop tôt

Dans un **SCN** (*Synergistic Connection Network*) évolutif, la mise à jour continue des pondérations $\{\omega_{i,j}\}$ peut conduire à la **dissolution** prématurée de certains **clusters** plus anciens. Lorsqu'un thème (un ensemble cohérent d'entités) n'est plus régulièrement sollicité, il peut subir la décroissance $-\tau\,\omega_{i,j}(t)$ ou des mécanismes d'inhibition qui finissent par **effacer** ses liaisons internes. Le **risque** est alors de **perdre** la mémoire d'un thème encore potentiellement valable, puisqu'il pourrait **redevenir** pertinent plus tard. La question se pose donc : comment **préserver** les clusters vieillissants pour ne pas devoir "tout reconstruire" s'ils se réactivent ?

A. Contexte et risques

Les équations DSL classiques incluent un **terme** de décroissance ou d'inhibition :

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

auquel on ajoute éventuellement $-\gamma \sum_{k\neq j} \omega_{i,k}\,(t)$ (inhibition) ou des règles de coupure si $\omega_{i,j}(t)$ passe sous un certain **seuil**. Ces mécanismes forcent la disparition de liens moyennement utilisés, mais peuvent de surcroît **anéantir** un cluster jadis utile dès qu'il n'est plus activement renforcé. Dans un **flux** de données, un "thème" peut cesser d'être alimenté momentanément, puis réapparaître. Si on l'a totalement dissous, on devra **repartir de zéro** pour le réformer.

B. Mécanismes mathématiques pour conserver les clusters en veille

On peut atténuer la décroissance pour un cluster ancien en introduisant un paramètre de "dormance" δ_α. Si $\mathcal{C}_\alpha$ est un cluster à préserver, on réduit la force de τ. Par exemple :

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

Le facteur $(1-\kappa\,\delta_\alpha)$ limite la chute $-\tau\,\omega$ pour les liaisons $(i,j) \in \mathcal{C}_\alpha$. On parle de "dormance" ou d'**inertie** du cluster : tant qu'il n'est pas déclaré obsolète, on retarde la décroissance totale de ses liens.

Si la suppression d'un lien se base sur un **seuil** (ex. $\omega_{i,j}(t) < \theta \Rightarrow \omega_{i,j}(t) \leftarrow 0$), on peut abaisser θ pour les liens appartenant à un cluster historique. Concrètement, on définit :

$$\theta_\alpha(t) = \theta_{\text{base}}(t) - \gamma_{\text{mem}} \times \text{score\_historique}(\mathcal{C}_\alpha),$$

la notion de *score_historique* reflétant l'importance ou la fréquence passée de $\mathcal{C}_\alpha$. Cela empêche la **coupure** des liens internes au cluster tant que ce "score historique" n'est pas tombé.

Une autre idée consiste à **inoculer** ponctuellement un **bruit** ou un **plancher** sur certains liens d'un cluster ancien, afin d'éviter leur effondrement complet. Par exemple, toutes les $\omega_{i,j}(t)$ se trouvant dans un cluster $\mathcal{C}_\alpha$ reçoivent :

$$\omega_{i,j}(t+1) = \max\{\omega_{i,j}(t),\, \zeta(t)\},$$

avec $\zeta(t)$ un petit plancher pouvant dépendre d'une logique interne (par ex. $\zeta = 0.001$). On ne laisse donc pas $\omega_{i,j}(t)$ descendre en dessous de ζ. Cela **maintient** un socle minimal de lien, prêt à être réactivé si la synergie remonte.

C. Compromis : éviter la saturation de liens inactifs

Si on **protège** trop vigoureusement un cluster en veille, on risque de "polluer" la matrice ω de liens inactifs mais jamais coupés. Cela peut ralentir la dynamique et saturer le SCN. On doit donc régler l'**amplitude** de la protection (dormance, seuil adaptatif) pour que :

- Un cluster réellement **désuet** finisse par s'éteindre,
- Les **clusters** qu'on veut conserver demeurent au-dessus d'un minimum de liaisons, permettant une éventuelle **réanimation** plus tard.

L'**inhibition** (voir §7.4) agit aussi dans ce sens : si un nouveau thème attire les entités, elles quittent l'ancien cluster, forçant la mort progressive de ce dernier s'il n'est plus du tout pertinent. Il s'agit donc de trouver un **juste milieu** entre *trop d'oubli* et *trop peu de compétition*.

D. Gains pour l'apprentissage continu

La possibilité de "dormance" évite l'**oubli catastrophique** dans un flux continu. Un thème ancien n'est pas effacé du jour au lendemain : son cluster interne n'est pas dissous tant qu'un léger flux d'entités y demeure ou qu'on a décrété un "score historique" suffisamment élevé.

Si la situation fait réapparaître ce thème, le SCN retrouve vite les liens (puisqu'ils n'ont pas été mis à zéro), accélérant la "réactivation". On gagne en **stabilité** et en **capacité** à gérer des thèmes qui évoluent dans le temps, tels que des sujets d'actualité, des projets, ou des contextes conversationnels intermittents.

Conclusion

Pour **ne pas** laisser un cluster ancien **disparaître** trop vite, on peut :

- Introduire un **facteur de dormance** ralentissant la décroissance τ pour ces liaisons,
- Ajuster un **seuil** de coupure θ en tenant compte de la mémoire du cluster,
- "Relancer" ponctuellement les liens via un plancher $\zeta(t)$ évitant qu'ils ne tombent à zéro.

Dans tous les cas, la logique **mathématique** consiste à moduler ou retarder l'effet de la réduction de $\omega_{i,j}$, pour **préserver** la capacité du SCN à "**réactiver**" ces liaisons si le thème devient de nouveau pertinent. C'est là un **compromis** entre la compétitivité / la place pour les nouveaux thèmes et la **stabilité** nécessaire pour conserver des acquis historiques, essentiel en **apprentissage continu**.

9.8.2. Robotique Temps Réel

Les **systèmes robotiques** s'inscrivent souvent dans des contextes hautement **dynamiques**, où le flux de données sensorielles et les configurations d'action évoluent en permanence. Dans un **SCN** (Synergistic Connection Network) dédié à la robotique, l'**apprentissage continu** en temps réel (voir chapitres précédents) permet de gérer simultanément l'arrivée de nouvelles informations et la reconfiguration nécessaire pour maintenir un comportement cohérent du robot (ou de l'essaim de robots).

9.8.2.1. Essaim de robots ou un robot unique multi-capteur

De nombreuses applications **robotiques** se prêtent naturellement à une modélisation **SCN** (*Synergistic Connection Network*) où la mise à jour $\omega_{i,j}(t)$ se déroule **en continu**. On distingue deux cas majeurs : l'**essaim** de robots multiples, chacun muni de divers capteurs, et le **robot unique** mais aux nombreux senseurs/actionneurs (caméra, LIDAR, bras articulés, etc.). Dans les deux scénarios, la **synergie** $S(i,j)$ peut exprimer la pertinence ou la complémentarité entre deux entités ($\mathcal{E}_i$ et $\mathcal{E}_j$), puis la **dynamique** DSL ajuste les liaisons $\{\omega_{i,j}\}$ en continu, permettant à des "clusters" (ou des regroupements) de **s'auto-former** ou de s'**auto-casser** au fil du temps.

A. Scénario d'un Essaim de Robots

Supposons qu'un **essaim** se compose de plusieurs robots mobiles. Chaque robot est modélisé comme une **entité** $\mathcal{E}_i$, ou un sous-réseau si l'on veut pousser la granularité (chaque robot subdivisé en modules). Les pondérations $\omega_{i,j}$ quantifient la **coopération** ou la **synergie** entre deux robots i et j. Dans un **DSL**, la mise à jour suit :

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

où $\Delta_{\text{inhibition ou bruit}}(i,j)$ peut modéliser de la **compétition** (inhibition latérale) ou du **recuit** (chap. 7.3–7.4). Sur le plan **mathématique**, $S(i,j)$ dépend par exemple de la **proximité** spatiale, de l'**historique** d'action commune, ou de la **complémentarité** en capteurs. Les robots se **synchronisent** alors en sous-groupes plus adaptés à chaque sous-tâche (cartographie d'une zone, exploration coordonnée, etc.). Si plus tard la mission change, la dynamique ω se reconfigure, un sous-ensemble formant un nouveau cluster.

Dans l'**essaim**, tout se fait sans recourir à un entraînement global hors-ligne. Les liaisons ω se réactualisent **au fil** des mesures (positions, capteurs), permettant la formation "**à la volée**" de **groupes** coopératifs.

B. Cas d'un Robot Unique mais Multi-Capteur

Ici, on ne traite pas plusieurs robots, mais un **robot** disposant de multiples **capteurs** (caméras, lidars, ultrasons, etc.) et **actionneurs** (roues, bras, pinces). On assimile chaque **capteur** ou **actionneur** à une entité $\mathcal{E}_k$. La **synergie** $S(\text{capteurA,capteurB})$ indique leur degré de **cohérence** ou de **complémentarité** (ex. redondance de mesures d'obstacles). La pondération $\omega_{\text{capA,capB}}$ se met à jour ainsi :

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

où ξ est éventuellement un bruit (recuit). **Mathématiquement**, cela autorise l'**auto-organisation** sensorimotrice : lorsque le robot change de mode (navigation vs. manipulation), les synergies évoluent, et ω se réadapte (une caméra se coordonne davantage avec le capteur de force, etc.).

On ne nécessite pas un module central *a priori* réglé : chaque capteur voit ses liaisons **renforcées** ou **affaiblies** au fil des itérations, favorisant la **synergie** adaptative. En temps réel, ce mécanisme DSL reste **local** et scalable.

C. Enjeux Temps Réel

Dans un essaim ou un robot multi-capteur, les mises à jour ω doivent opérer sous des **contraintes** de calcul et de réactivité :

- On ne peut se permettre un **recalcul** total à chaque instant. Des heuristiques de **voisinage** (k-NN) ou de **sparsité** (inhibition) sont cruciales.
- Les **paramètres** (η, τ, γ, amplitude du bruit) peuvent être ajustés dynamiquement pour s'adapter à l'environnement changeant.
- Si une formation de cluster s'avère inadéquate (p. ex. un robot coincé dans un sous-groupe improductif), on recourt à de la **compétition** plus forte ou à un **recuit** ponctuel (cf. ch. 7.3) pour "s'échapper" de l'attracteur local.

Conclusion sur l'Essaim / Robot Multi-Capteur en Temps Réel

Que l'on parle d'un **essaim** constitué de multiples robots ou d'un **robot** unique aux nombreux senseurs, le **DSL** en mode temps réel procure une **mise à jour** des pondérations $\omega_{i,j}$ continue et **auto-organisée**. Chaque entité (robot ou capteur) se **connecte** plus étroitement à celles avec lesquelles la **synergie** s'avère élevée, et la configuration ω se reconfigure **au fil** de l'évolution des tâches ou de l'environnement. Le **recuit** (bruit) et l'**inhibition** font partie des outils clés pour :

- **Briser** les configurations trop figées,
- **Sélectionner** les liaisons essentielles dans la compétition,
- **Permettre** une plasticité et une reconnaissance de nouveaux groupes fonctionnels.

Cela offre une approche **flexible** et **robuste** pour gérer l'apprentissage **en continu** de la coordination robotique, sans avoir à relancer un nouveau **entraînement** global à chaque changement de contexte.

9.8.2.2. Les capteurs évoluent (différents environnements), le SCN s'adapte sans redémarrer

Dans un **SCN** (*Synergistic Connection Network*) appliqué à la **robotique** ou à la gestion de **systèmes multi-capteurs**, l'un des grands atouts du **Deep Synergy Learning (DSL)** consiste en sa capacité à **s'actualiser** en continu lorsque les **capteurs** ou l'**environnement** se transforment. Concrètement, même si la configuration sensorielle est modifiée (un capteur en plus, un capteur défaillant, ou un changement de conditions extérieures), le réseau de liaisons $\{\omega_{i,j}\}$ peut se **reconfigurer** progressivement, sans qu'on ait à relancer un "reset" complet ni à réentraîner une structure de zéro. Cette faculté naît de la **logique** DSL : une mise à jour locale en temps réel, incrémentée à chaque itération, suivant des mécanismes d'adaptation continue (chap. 7.6).

A. Scénario : Capteurs évoluant selon l'environnement

Imaginons un **robot** naviguant dans divers contextes. Ses capteurs peuvent être plus ou moins **fiables** selon la luminosité, la météo, la géographie. Par exemple, une caméra infrarouge s'avère très efficace la nuit mais moins utile le jour ; un LIDAR peut perdre de la pertinence sous un soleil intense ou dans un brouillard épais. On peut aussi ajouter ou supprimer un capteur (ex. brancher un GPS, débrancher une caméra). Dans un **SCN**, la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j, t)$ reflète cette dépendance au contexte : pour chaque entité (capteur) i, on évalue à chaque instant t la cohérence ou la complémentarité avec un autre capteur j.

Formellement, la mise à jour DSL :

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

montre que la pondération $\omega_{i,j}(t)$ s'adapte si $S(\cdot,\cdot,t)$ change de valeur au cours du temps. S'il devient plus utile de combiner la caméra infrarouge et le LIDAR sous un certain éclairage, la synergie $S(\text{camIR,LIDAR},t)$ croît et pousse $\omega_{\text{camIR,LIDAR}}$ vers le haut, renforçant le **couplage** de ces deux capteurs. À l'inverse, si un capteur n'est plus employé, ses pondérations se réduisent progressivement, sans pour autant imposer un "reboot" global.

L'**incrémentation** progressive de ω apporte un **apprentissage** fluide dans des environnements variables. On ne détruit pas la structure établie dans un environnement précédent (ce qui autorise la réutilisation de connaissances si on revient à un contexte similaire), tout en permettant l'**émergence** de nouveaux liens pertinents pour la situation actuelle.

B. Adaptation Incrémentale sans "reset"

Au lieu d'une stratégie batch (où l'on bloque le robot pour re-entraîner un "gros" modèle si un capteur change), le **SCN** se prête à un ajustement local : on ne modifie que les liaisons $\omega_{i,j}$ concernées, en lien avec la synergie $S(\cdot,\cdot,t)$. Si un nouveau capteur $\mathcal{E}_{\text{new}}$ est ajouté, on initialise ses ω à 0 ou ϵ et on laisse la boucle DSL croître ou amortir ces pondérations selon la corrélation relevée avec d'autres capteurs.

Lorsque la distribution $S(\cdot,\cdot,t)$ se "fixe" un moment (par ex. on reste dans un même type d'environnement quelques minutes), la dynamique DSL s'approche d'un **nouvel** état stable où les liaisons $\{\omega_{i,j}\}$ cohérentes deviennent fortes, tandis que celles non pertinentes se sont affaiblies. Ce mécanisme se déroule **sans** qu'il soit nécessaire de "tout effacer" (reset) pour relancer l'apprentissage.

Si un capteur est définitivement débranché ou s'il ne fournit plus d'information utile (synergie ≈ 0 avec les autres), ses liaisons ω retomberont naturellement. Dans la perspective DSL, c'est un **apprentissage** par érosion : nulle part on impose qu'il faille *retirer* ce capteur du SCN ; simplement, ses ω rejoignent zéro, le rendant inactif. De même, un capteur "inédit" inséré dans le SCN trouve ses liens $\omega_{\text{new},j}$ se développer s'il prouve sa complémentarité, ou rester faibles sinon.

C. Approche Mathématique de la Continuité

On peut concevoir le SCN comme un **système** dont l'**énergie** ou la **fonction** $\mathcal{J}$ dépend explicitement de l'environnement et du temps. À chaque instant t, $\mathbf{\Omega}(t)$ se déplace dans l'espace des pondérations en suivant la règle DSL modulée par $S(\cdot,\cdot,t)$. Sous des hypothèses de régularité, la suite $\{\mathbf{\Omega}(t)\}$ s'adapte à mesure que la synergie évolue.

En **apprentissage continu**, on veut qu'un cluster apparu précédemment ne disparaisse pas totalement dès que le contexte change (cf. 9.8.1.3). Mais on veut aussi éviter qu'il *s'oppose* à la création de nouvelles liaisons plus pertinentes. Les mécanismes d'**inhibition** ou d'**injection de bruit** (chap. 7.3–7.4) régulent ce trade-off : on autorise des reconfigurations plus tardives, tout en gardant un minimum de mémorisation.

Cette situation s'apparente à un *système thermodynamique* dont les paramètres externes (température, pression) varient. Le SCN demeure proche d'un **minimum** local "instantané", mais comme le paysage $S(\cdot,\cdot,t)$ se déforme graduellement, la configuration $\mathbf{\Omega}(t)$ "glisse" ou "suit" ce mouvement, sans exiger de recréer toute la structure.

D. Avantages Concrets

En robotique temps réel, il est **coûteux** de relancer un apprentissage complet dès qu'on change d'environnement. Le SCN procédant de façon **incrémentale** économise un recalcul global. Il modifie seulement quelques liaisons $\omega_{i,j}$ en fonction du changement de S.

Des liaisons acquises par le passé ne sont pas supprimées tant qu'elles ne s'avèrent pas incohérentes. Si le système revient dans un ancien état (par ex. ré-entre dans un entrepôt sombre), on réutilise ou **ravive** des liaisons déjà hautes au lieu de devoir tout réinventer.

Si l'environnement oscille entre deux types de conditions (intérieur/extérieur, mode navigation/manipulation), la mise à jour DSL se contente de "pivoter" la configuration ω d'un état stable à un autre, minimisant les temps d'adaptation et préservant la cohérence globale.

Conclusion

Quand un système multi-capteurs traverse **différents** environnements, le **SCN** en mode DSL s'ajuste **sans** requérir de nouveau "reset" ou d'**apprentissage batch**. La formule :

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

incorpore naturellement le fait que $S(\cdot,\cdot,t)$ puisse changer au fil du temps. Les liens ω reflètent cette variation, s'érodant ou se renforçant, et aboutissent à un **nouvel** équilibre sans passer par une remise à zéro de la structure. Cela illustre la **plasticité** continue caractéristique du DSL, qui demeure vital pour une robotique ou un système multi-capteurs confronté à la **variabilité** du réel, tout en maintenant la **stabilité** permettant de réutiliser les connaissances passées si l'environnement revient à une configuration antérieure.

9.8.2.3. Gains en Flexibilité et Robustesse Comparés à un Pipeline Figé

Dans de nombreux systèmes d'apprentissage ou de traitement continu, une démarche **classique** consiste à mettre en place un **pipeline** figé : on segmente en étapes fixées (extraction de caractéristiques, classification, post-traitement, etc.), avec des **modules** peu adaptatifs. Une telle approche se révèle souvent **limitée** lorsque l'on souhaite intégrer en continu de **nouvelles** données, de **nouvelles** modalités, ou lorsque l'environnement évolue. Par contraste, la **philosophie** DSL (*Deep Synergy Learning*) ou l'usage d'un **SCN** (*Synergistic Connection Network*) "vivant" confère des **avantages** majeurs en termes de **flexibilité** et de **robustesse**, en grande partie grâce à l'**auto-organisation** et à l'**apprentissage continu**.

A. Limites d'un Pipeline Figé

Les architectures **pipeline** consistent généralement en une suite **intangible** de modules : un bloc d'extraction (features), un bloc de classification, un module de fusion, etc. Chaque composant est **optimisé** ou paramétré de manière indépendante, et l'on se retrouve ensuite avec une **chaîne** rigide : pour **modifier** l'un de ces modules (changement de la méthode de feature extraction ou de la classe finale), il faut souvent **réentraîner** ou au moins réajuster tout le pipeline. Sur le plan **mathématique**, on peut voir cela comme une composition de fonctions fixes $\Phi_1 \circ \Phi_2 \circ \ldots$, où chaque Φ_k n'échange pas réellement de **feedback** avec les autres.

Une fois que ce pipeline est défini, s'il arrive un **nouveau** type de données (p. ex. un capteur additionnel) ou si l'environnement change drastiquement, le **pipeline** ne peut guère s'**adapter**

localement. Il faut généralement **reconcevoir** la chaîne ou tout revalider, ce qui est coûteux et lent.

En situation de **flux** de données ou d'évolution, on se retrouve à devoir régulièrement "patcher" ou "finement retoucher" le pipeline, risquant de **casser** une cohérence globale. Un pipeline figé n'exploite pas non plus l'idée qu'une entité $\mathcal{E}_i$ puisse **co-opérer** avec plusieurs composants simultanément : toute la structure d'information est décidée **en amont** de l'exécution, limitant la plasticité à l'exécution.

B. Flexibilité du DSL ou du SCN Évolutif

À l'inverse, un **SCN** ou un **DSL** évolutif envisage l'**apprentissage** comme un **processus** permanent d'**auto-organisation** : chaque entité $\mathcal{E}_i$ (capteur, modalité, document) entretient des liaisons $\{\omega_{i,j}\}$ qui se renforcent ou s'affaiblissent en fonction de la **synergie** $S(i,j)$. L'**absence** d'une structure de pipeline rigide se traduit par une **topologie** pouvant changer librement au fil du temps, en particulier lorsque arrivent de nouvelles entités ou de nouvelles dimensions.

Dans un SCN, la synergie $S(i,j)$ incite certains nœuds $\mathcal{E}_i$ et $\mathcal{E}_j$ à **former** un cluster s'ils sont très complémentaires ou statistiquement cohérents. De plus, si le système intègre des mécanismes **multi-niveau** ou d'**agrégation hiérarchique**, ces micro-clusters se regroupent parfois en macro-nœuds. Cette flexibilité ne nécessite pas de *définir* a priori un trajet d'information : la synergie émerge localement et oriente la **mise à jour** $\omega_{i,j}(t)$.

Chaque fois qu'un **nouveau** type de donnée apparaît, on l'insère dans la matrice ω, en calculant ses synergies $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$. Le reste du SCN se réajuste peu à peu, **sans** que l'on doive réentraîner l'ensemble "from scratch". L'effort se borne à la **dynamique** locale, gérée par la formule :

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

ou ses variantes (avec recuit, inhibition, etc.). Ainsi, on peut accueillir un nouveau flux de capteurs ou de documents sans bouleverser l'existant.

C. Robuste aux Perturbations ou aux Évolutions

Lorsqu'un capteur ou un module interne devient défaillant, un pipeline rigide est susceptible de produire des sorties erronées ou de s'effondrer. Dans un SCN, si la fiabilité de l'entité $\mathcal{E}_i$ chute, la **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ se répercute sur $\omega_{i,j}$: ces poids se **réduisent**, isolant le module défectueux et conservant la cohérence globale.

Si un **contexte** évolue fortement (nouvelles catégories, changement total de l'environnement), un pipeline nécessiterait une "mise à jour" globale. Le SCN, lui, **adapte** la topologie ω peu à peu : les liaisons obsolètes sont érodées, de nouvelles surgissent pour refléter la synergie dans la situation contemporaine, permettant un "**réapprentissage**" local continu.

En fusion multimodale : un pipeline classique prévoit un module d'extraction "visuel" et un module "sonore", puis fusionne les features et envoie la classification. Introduire une nouvelle modalité (capteur infrarouge ou un autre canal) s'avère complexe à insérer dans ce pipeline. Inversement, un SCN peut "brancher" le nouveau capteur comme une entité, initialiser ses synergies $S(\text{new}, \text{old})$ puis laisser la dynamique DSL le **connecter** aux parties du réseau où se révèle une cohérence réelle.

D. Conclusion : Gains en Flexibilité et Robustesse

Le SCN (ou DSL) n'impose pas de "chemin" figé. Les nœuds s'**associent** selon leurs synergies ; si le système reçoit un **nouveau** flux de données ou détecte un module changeant, la **descente** locale s'adapte en **continu**.

Contrairement au pipeline figé où la panne d'un module peut entraîner un blocage total, le SCN réalloue les liaisons : le module défaillant se voit relégué à un rôle marginal (faibles ω), laissant le reste continuer à fonctionner. En cas de grand changement, on **réévalue** la structure sans avoir à tout recompiler.

En somme, un pipeline classique, séquentiel et figé, se montre peu apte à un **flux** évolutif ou à l'**insertion** de nouvelles modalités. Un **SCN** évolutif, au contraire, se **reconfigure** organiquement au fil des itérations, maintenant une **mémoire** des configurations passées tout en restant **ouvert** aux nouvelles sources ou aux changements de l'environnement. Cette **auto-organisation** en continu assure un **apprentissage** plus souple et **robuste**, essentiel en robotique temps réel ou dans les applications multimodales où l'on ne peut se permettre de "rebâtir" la chaîne à chaque évolution.

9.8.3. Multimodal Streaming (Images / Audio / Textes)

Dans certains environnements de **streaming multimodal**, un **flux** contient simultanément plusieurs types de contenus : segments vidéo, extraits audio, légendes textuelles, etc. Les approches DSL (Deep Synergy Learning) doivent donc gérer l'**intégration** continue de ces différents canaux et maintenir une **cohérence** dans la formation des clusters. La notion de **synergie** devient alors multidimensionnelle, englobant (vidéo–vidéo), (audio–audio), (texte–texte), mais aussi des couplages **cross-modaux** (ex. vidéo–audio, audio–texte). Cette section (9.8.3) illustre comment l'on peut insérer de nouveaux segments (images, sons, descriptions) dans un **SCN** (Synergistic Connection Network) déjà partiellement formé, et observer la **dynamique** d'auto-organisation qui en résulte.

9.8.3.1. Insertion de segments vidéo + audio + légendes

Dans une situation **multimodale**, on peut recevoir de façon **continue** des segments composés de plusieurs **canaux** : vidéo, audio, et légende (texte). L'objectif, dans un **SCN** (*Synergistic Connection Network*) géré par un **Deep Synergy Learning (DSL)**, est d'**insérer** chacun de ces nouveaux segments $\mathcal{E}_{\text{new}}$ dans le réseau au fil de l'eau, en évaluant la **synergie** avec les segments préexistants et en **mettant à jour** les liaisons $\omega_{(\text{new}),\,j}$.

On suppose qu'à chaque instant t, arrive un segment $\mathcal{E}_{\text{new}}$ intégrant trois composantes :

$$\mathcal{E}_{\text{new}} = (\text{trame vidéo},\ \text{extrait audio},\ \text{légende textuelle}).$$

Chacune est convertie en un **vecteur** embedding : $\mathbf{v}^{(\text{vid})}$, $\mathbf{v}^{(\text{aud})}$, $\mathbf{v}^{(\text{txt})}$. Dans le **SCN**, ces entités sont indexées pour que l'on puisse comparer $\mathcal{E}_{\text{new}}$ aux entités plus anciennes $\{\mathcal{E}_j\}$.

Pour un segment déjà existant $\mathcal{E}_j$, on définit un **score** :

$$S(\mathcal{E}_{\text{new}}, \mathcal{E}_j) = \alpha\,\text{Sim}\left(\mathbf{v}_{\text{new}}^{(\text{vid})},\, \mathbf{v}_j^{(\text{vid})}\right) + \beta\,\text{Sim}\left(\mathbf{v}_{\text{new}}^{(\text{aud})},\, \mathbf{v}_j^{(\text{aud})}\right) + \gamma\,\text{Sim}\left(\mathbf{v}_{\text{new}}^{(\text{txt})},\, \mathbf{v}_j^{(\text{txt})}\right),$$

où $\text{Sim}(\cdot,\cdot)$ est une mesure de similarité (par ex. cosinus) et α, β, γ des poids reflétant l'importance relative de chaque canal (vidéo, audio, texte).

Une fois la **synergie** $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ calculée, la **règle** DSL actualise les liaisons :

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

avec éventuellement des **termes** d'inhibition ou de recuit (voir chap. 7). Dans l'implémentation, on peut calculer $S(\mathcal{E}_{\text{new}},\, \mathcal{E}_j)$ seulement pour un **voisinage** pertinent, afin d'éviter un coût $O(n)$ trop grand quand n croît (chap. 7.2.3.3). Au fil de cette mise à jour, $\omega_{(\text{new}),\,j}$ monte s'il existe une **forte** corrélation multimodale, ou baisse si la synergie se révèle faible.

L'arrivée **successive** de multiples segments (chaque segment associant vidéo, audio, texte) engendre, dans le SCN, un **renforcement** des liens ω entre entités similaires ou complémentaires. À mesure que plusieurs segments partagent un même sujet (par ex. des extraits "concert de rock"), leurs pondérations ω internes se solidifient, **émergent** alors un **cluster** regroupant les entités "concert". On retrouve la logique standard de l'auto-organisation DSL : la **somme** ou la densité de liaisons dans ce sous-groupe devient notable, dépassant un **seuil** θ, marquant la création d'un **thème** clairement identifiable.

En parallèle, si un autre flux (p. ex. segments "conférence scientifique") survient, la dynamique DSL peut :

- **Maintenir** le cluster "concert" avec ses liaisons élevées,
- **Faire naître** un deuxième cluster "conférence", si les synergies multimodales y sont conséquentes,
- **Réorganiser** certains segments "intermédiaires" qui pourraient basculer entre les deux clusters, selon la mise à jour de leurs liaisons ω.

Puisqu'on est en **flux**, la matrice ω se met à jour **sans** arrêt. À tout moment :

- Un **cluster** en place peut continuer de grandir si de nouveaux segments multimodaux sont cohérents,
- Un **nouveau** cluster peut émerger si un groupe d'entités affiche un niveau ω interne substantiel,
- Les **anciens** clusters peuvent s'affaiblir s'ils ne sont plus alimentés, à moins qu'on ne prévoit un **mécanisme** (cf. 9.8.1.3) pour éviter une dissolution trop rapide.

Ceci se déroule **incrémentalement**, sans reconstruction générale, grâce aux mises à jour locales :

$$\omega_{i,j}(t+1) = \omega_{i,j}(t) + \Delta_{\text{DSL}}(i, j; t).$$

A. Caractère "Streaming" : Variation Continue

Comme dans tout **SCN** évolutif, l'arrivée de segments audio–vidéo–textes ne cesse pas. Chaque **insertion** $\mathcal{E}_{\text{new}}$ déclenche un calcul de synergie $S(\mathcal{E}_{\text{new}}, \mathcal{E}_j)$ et l'**extension** de la matrice ω. On peut appliquer l'équation :

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

à chaque itération, éventuellement en “mini-lots” (pour traiter plusieurs segments d’un coup). La **topologie** du réseau demeure dynamique : des clusters naissent, d’autres se dissolvent ou fusionnent, reflétant l’évolution du **contenu** multimodal au fil du temps.

B. Complément Mathématique : Mesure de Synergie Multimodale

Pour **combiner** la similarité de plusieurs canaux (vidéo, audio, texte), on peut faire une **somme** pondérée :

$$S(\mathcal{E}_i, \mathcal{E}_j) = \alpha_1 \,\mathrm{Sim}\left(\mathbf{v}_i^{(\mathrm{vid})}, \mathbf{v}_j^{(\mathrm{vid})}\right) + \alpha_2 \,\mathrm{Sim}\left(\mathbf{v}_i^{(\mathrm{aud})}, \mathbf{v}_j^{(\mathrm{aud})}\right) + \alpha_3 \,\mathrm{Sim}\left(\mathbf{v}_i^{(\mathrm{txt})}, \mathbf{v}_j^{(\mathrm{txt})}\right).$$

On peut également intégrer des termes *cross-modaux* (image–texte, audio–texte) si l’on souhaite détecter la **cohérence** entre canaux différents d’un même segment. La mise à jour DSL s’accommode de **toutes** ces variantes, puisqu’il lui suffit de recevoir un score $S(\cdot,\cdot)$ global pour ajuster ω.

Conclusion

Dans un environnement **multimodal** où des **segments** (vidéo + audio + texte) affluent en continu, la **mise à jour** DSL permet d’**insérer** chaque segment $\mathcal{E}_{\mathrm{new}}$ au sein du SCN en :

- **Calculant** la synergie $S(\mathcal{E}_{\mathrm{new}}, \mathcal{E}_j)$ pour chaque entité $\mathcal{E}_j$ pertinente,
- **Appliquant** la règle d’évolution $\omega_{(\mathrm{new}),j}(t+1) = \omega_{(\mathrm{new}),j}(t) + \cdots$,
- **Observant** l’émergence ou la dissolution de clusters correspondant à des **thèmes** (“concert rock”, “conférence science”, etc.).

Ce **cadre** se marie avec des modules d’**inhibition** ou de **recuit** (cf. chap. 7) pour contrôler la prolifération de liens et éviter la stagnation en minima locaux. Le **SCN** se **réorganise** ainsi de manière incrementale, ce qui s’avère essentiel en **flux** multimodal permanent, où il n’est pas question de *reconstruire* la topologie à chaque nouvelle donnée. En pratique, cette **architecture vivante** favorise la détection spontanée de **clusters** (thèmes) en streaming, reflet de la **plasticité** et de la **cohérence** offertes par la logique DSL.

9.8.3.2. Observer la dynamique : un cluster se forme autour d’un thème (ex. “concert”), puis d’autres flux arrivent

Dans un **SCN** (*Synergistic Connection Network*) traitant des **flux** de données (messages textuels, extraits multimédias, requêtes, etc.), il est courant de voir surgir un **cluster** associé à un **thème** spécifique lorsque certaines **entités** (messages, segments vidéo-audio, etc.) s’avèrent cohérentes entre elles. Par exemple, une série de messages centrés sur un mot-clé “concert” ou un ensemble de concepts comme “billets”, “artistes”, “spectacles” peut amorcer la formation d’un **cluster** qui se consolide au fil des itérations. Parallèlement, dès que de **nouveaux** flux apparaissent, ces messages peuvent se greffer au cluster existant, le scinder ou en fonder un autre.

A. Formation initiale du cluster “concert”

On suppose qu’un flux de données arrive, incluant divers entités $\{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$, dont un **sous-ensemble** se réfère explicitement à un sujet comme “concert”. Par exemple, des tweets ou documents où apparaissent des mots-clés “billets de concert”, “acheter place de spectacle”,

“artiste X”. La **synergie** $S(\mathcal{E}_i, \mathcal{E}_j)$ entre deux entités $\mathcal{E}_i$ et $\mathcal{E}_j$ indique leur **proximité** sémantique ou leur **co-occurrence**.

Dans la logique **DSL**, la matrice de pondérations ω\omega se met à jour par :

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

Si “concert” est fortement **corrélé** à “billet de concert” ou “place de concert”, on a $S(\mathcal{E}_{\text{concert}}, \mathcal{E}_{\text{billet}})$ élevé, d’où un **renforcement** de ω. Au fil des itérations, cela **construit** un cluster $\mathcal{C}_{\text{concert}}$ autour des entités relatives à “concert”.

On peut évaluer la **cohésion** du cluster $\mathcal{C}_{\text{concert}}$ en calculant la somme de ses liaisons internes :

$$\Omega(\mathcal{C}_{\text{concert}}) = \sum_{i,j \in \mathcal{C}_{\text{concert}}} \omega_{i,j}.$$

Plus Ω croît, plus ces entités partagent un fort degré de synergie, **convergeant** vers un **thème** unique. À mesure que d’autres entités mentionnant “concert” ou “artiste X” arrivent, elles s’intègrent dans ce cluster si leur synergie avec les nœuds du bloc “concert” s’avère assez élevée.

B. Arrivée de nouveaux flux et réorganisation

Après qu’un cluster “concert” s’est stabilisé, de nouveaux **messages** arrivent, certains tout à fait cohérents avec le sujet (par ex. “concert en plein air”, “parking pour le concert”), d’autres sans aucun lien (“football”, “politique”). Si un message $\mathcal{E}_{\text{new}}$ est partiellement lié au “concert” mais évoque aussi un autre thème, sa **synergie** $S(\mathcal{E}_{\text{new}}, \mathcal{E}_{\text{concert}})$ sera positive, mais il pourra aussi avoir une synergie notable avec un autre cluster (ex. “concert + foot” = un mélange inhabituel, menant potentiellement à la création d’un sous-cluster “concert-sport”).

Si le flux $\mathcal{E}_{\text{new}}$ se révèle très **proche** du thème “concert”, ses liaisons $\omega_{(\text{new}),i}$ prendront vite de la valeur, agrandissant le cluster. Si, au contraire, le flux n’est que partiellement relié, on peut assister à la naissance d’un **nouveau** regroupement, ou à la **scission** d’une partie du bloc “concert” si ces nouveaux messages incitent une **spécialisation** (par ex. “concert virtuel”, “concert caritatif”).

À mesure que le flux s’enrichit, la **dynamique** DSL peut faire **évoluer** la répartition des entités. Un sous-groupe du bloc “concert” peut se détacher si sa synergie interne est plus forte qu’avec le reste, illustrant une scission. Cela reflète une **logique** de clustering hiérarchique local : si un sous-thème “concert-metal” se distingue fortement, on observera l’apparition d’un cluster dédié.

C. Observation et pilotage

On peut visualiser l’**évolution** de la matrice ω, repérer la **courbe** d’énergie $\mathcal{J}(t)$ ou le score de cohésion $\Omega(\mathcal{C}_{\text{concert}})$. On observe comment le **cluster** “concert” s’intensifie lorsque de nouveaux flux cohérents affluent, et comment il se scinde ou se réorganise face à un afflux différent.

On peut intervenir via des mécanismes d’**inhibition** (ch. 7.4) pour éviter un cluster trop massif ou un recuit (ch. 7.3) si on craint un blocage local. De plus, des modules hiérarchiques (ch. 6) permettent de **détecter** qu’un sous-groupe distinct apparaît et le promouvoir comme un “sous-thème”.

Conclusion

Lorsque le **SCN** reçoit un flux continu d'entités multimédias ou textuelles, on voit apparaître un **cluster** autour d'un sujet (ex. "concert") dès que les synergies S s'accumulent. Puis, avec l'arrivée de **nouveaux** messages, ce bloc "concert" se **renforce** (accueillant d'autres entités) ou peut se **scinder** si des sous-thèmes émergent (ex. "concert virtuel", "concert rock"). Le **DSL** gère cela **incrémentalement**, sans nécessiter de réinitialisation globale. Sur le plan **mathématique**, les pondérations ω se reconfigurent au fil des itérations, soutenues par des techniques d'inhibition ou de recuit pour éviter la stagnation et permettre la formation — ou la fragmentation — de clusters selon la cohérence des nouveaux flux.

9.8.3.3. Rôle du recuit pour *reshaper* les clusters si la distribution change radicalement

Dans un **SCN** (Synergistic Connection Network) en **apprentissage continu**, il n'est pas rare que la **distribution** des entités ou des synergies $\{S(i,j)\}$ subisse des transformations majeures : une modification d'environnement, l'arrivée d'une nouvelle modalité capteur, un changement profond de contexte (p. ex. dans un flux de tweets, on bascule d'un sujet "concert" à un sujet "actualité politique", etc.). Dans ces situations, un **réseau** DSL (Deep Synergy Learning) peut rester "coincé" dans un **minimum local** hérité de l'ancienne distribution, si l'on ne prévoit pas de mécanisme pour le "**sortir**" de cette configuration dépassée. Le **recuit simulé** (ou son équivalent, l'injection de **bruit** avec une "température" maîtrisée) fournit alors un **levier** décisif pour "bousculer" la topologie $\{\omega_{i,j}\}$ et **reconstruire** des clusters adaptés au nouvel état de la distribution.

A. Contexte : changement radical de distribution

Plusieurs cas peuvent conduire à un brutal **changement** de la distribution sous-jacente à un **SCN** :

- **Krach boursier** ou crise économique, rendant caduques les anciennes corrélations entre actions.
- **Robotique** : arrivée de nouveaux capteurs, passage d'un environnement intérieur à un extérieur.
- **Système textuel** : changement de langue, ou bascule vers un nouveau domaine lexical (ex. passage de "concerts" à "élections").

Dans chacun de ces cas, la simple mise à jour $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$ s'avère parfois **insuffisante** : la structure ω se trouve **gravée** dans les liaisons fortes issues de l'ancienne configuration, sans parvenir à spontanément "décoller" de ce minimum local désormais inadapté.

B. Principes du recuit pour re-façonner les clusters

La mise à jour DSL ordinaire :

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

peut se compléter d'un **terme stochastique** $\sigma(t)\,\xi_{i,j}(t)$, où $\sigma(t)$ est la "température" et $\xi_{i,j}(t)$ un bruit gaussien ou uniforme. Dans un scénario de "changement radical", on "**réchauffe**" le

système en augmentant σ à un certain moment t_0. Cela permet aux pondérations $\omega_{i,j}$ d'effectuer des **sauts** suffisamment amples pour rompre la structure précédente.

Lorsqu'on **détecte** (par un indicateur statistique, un test de drift, etc.) que la distribution a basculé, on élève $\sigma(t)$ à un niveau plus fort que celui du mode normal, autorisant de grands bouleversements dans la matrice ω. Puis on laisse σ **redescendre** en suivant un plan ($\sigma(t+1) = \alpha\sigma(t)$) pour figer le SCN dans une configuration mieux adaptée aux $S(i,j)$ du moment.

Ces perturbations stochastiques **brisent** les liaisons moyennes ou fortes n'ayant plus de justification dans la nouvelle distribution, tout en laissant la dynamique DSL **reconsolider** les liens cohérents. Ainsi, on observe un "**reshaping**" : d'anciens clusters se dissolvent, de nouvelles associations plus conformes à la distribution modifiée émergent.

C. Analyse Mathématique : évasion d'un puits d'énergie

Si Ω_{old} était un **minimum local** pour l'ancienne fonction $\mathcal{J}$ (ou l'ancienne répartition de synergie), la bascule de distribution fait que Ω_{old} n'est plus un minimum pour la nouvelle $\mathcal{J}$. Pourtant, la descente locale reste **coincée**. Le recuit injecte des "sauts" potentiellement assez grands pour franchir la **barrière d'énergie**.

En termes **Markoviens**, chaque mise à jour ($\boldsymbol{\Omega} \rightarrow \boldsymbol{\Omega}'$) a une probabilité plus élevée de "quitter" la vallée si $\sigma(t)$ est grand. On peut alors atteindre un **nouveau** minimum local Ω_{new} plus conforme à la distribution modifiée.

Après un laps de temps (quelques itérations) où l'on a "**chauffé**" le SCN, on peut abaisser σ pour stabiliser la nouvelle configuration. On retrouve la logique d'un **recuit simulé** classique, mais appliqué sur un **SCN** dont la distribution $S(\cdot,\cdot)$ a changé.

D. Intérêt Pratique et Implémentation

Sans recuit, on pourrait avoir besoin d'un "reset" complet pour forcer le réseau à **oublier** l'ancienne structure. En recourant plutôt au recuit, on ne part pas de zéro : on garde la matrice ω, tout en la perturbant suffisamment pour s'éloigner de Ω_{old}.

On peut s'appuyer sur (i) un test de changement de distribution (chap. 9.6.1 ou 9.6.2), ou (ii) un simple constat de stagnation dans $\mathcal{J}$. À ce moment, on rehausse la température σ. Au fur et à mesure, on la fait redescendre pour "figer" la nouvelle organisation.

Les **mécanismes** (inhibition, saturation) présentés au chap. 7.4–7.5 restent actifs durant le recuit. Cela peut rendre le **reshaping** plus sélectif : si la distribution a vraiment basculé, l'inhibition accélère la perte de liens obsolètes, tandis que de nouveaux liens se forment.

Conclusion

Lors d'un **changement radical** de la distribution, un **SCN** évolutif peut s'attacher à un ancien **minimum** local inadapté. Le **recuit** agit comme un **choc thermique**, permettant un "**reshaping**" des clusters, autrement dit une **restructuration** de la matrice ω. On "réchauffe" le système pour provoquer de grands sauts, franchir la barrière d'énergie et atteindre un nouvel arrangement plus cohérent avec la distribution modifiée. On "refroidit" ensuite, stabilisant la nouvelle topologie. D'un point de vue **mathématique**, ce protocole stochastique facilite l'évasion hors du puits ancien vers un nouveau **minimum** local (ou global) mieux adapté, sans avoir à réinitialiser totalement le SCN ni perdre la mémoire des liaisons qui peuvent encore

servir. Cela confirme la **puissance** du recuit simulé dans l'**apprentissage continu**, surtout quand des modifications majeures apparaissent dans l'environnement ou le flux de données.

9.9. Limites, Défis et Pistes de Recherche

Dans les sections précédentes, nous avons mis l'accent sur l'évolution en flux (Chap. 9) et la façon dont un **SCN** (Synergistic Connection Network) peut s'adapter en continu à l'arrivée de nouvelles entités (capteurs, événements, tokens, etc.). Malgré ces acquis, il persiste des **défis** et **limites** nécessitant des travaux complémentaires. La section 9.9 vise à en dresser un panorama, en insistant sur le **coût computationnel** (9.9.1), le **contrôle de la plasticité** (9.9.2), l'**évaluation continue** (9.9.3) et d'autres pistes d'intégration ou de couplage (9.9.4).

9.9.1. Coût Computationnel

L'un des enjeux majeurs en mode **flux** est la croissance potentiellement illimitée du nombre d'entités $\{\mathcal{E}_i\}$. Sur un plan **mathématique** et algorithmique, plus n (le nombre d'entités) croît, plus le réseau risque de se retrouver avec $O(n^2)$ liens potentiels $\omega_{i,j}$ à gérer, ce qui peut **déborder** en mémoire, en CPU, ou en latence si on n'y prend garde.

9.9.1.1. Avec un flux continu, les entités peuvent croître de manière illimitée

Lorsqu'un **SCN** (*Synergistic Connection Network*) opère en mode **flux**, il n'existe pas forcément de borne sur le **nombre** d'entités $\{\mathcal{E}_i\}$. Au fil du temps, on peut recevoir une série ininterrompue de **nouvelles** données (capteurs, documents textuels, etc.), que le réseau se doit d'intégrer. Sur le plan **mathématique**, plus n (le nombre d'entités) grandit, plus la matrice ω (de dimension $n \times n$) enfle, et plus la gestion de ses $O(n^2)$ éléments peut s'avérer **intraitable** : temps de calcul, mémoire, latence. L'enjeu est donc de **limiter** la croissance ou de trouver des **techniques** maintenant la dynamique DSL viable malgré l'augmentation ininterrompue de n.

Si, à chaque arrivée d'entité $\mathcal{E}_{n+1}$, on l'**ajoute** au SCN sans jamais en **supprimer** d'anciennes, on aboutit à un **réseau** d'entités $\{\mathcal{E}_1, \dots, \mathcal{E}_n\}$ en inflation continue. Chaque itération de la mise à jour DSL théorique, qui balaye **tous** les couples (i,j), exige alors $O(n^2)$ opérations. **Mathématiquement**, cela devient prohibitif quand $n \to \infty$. De surcroît, le **stockage** de la matrice ω (taille $n \times n$) cause un coût mémoire $O(n^2)$. On se heurte vite à un plafond pratique, voire à un "overflow" en cas d'application sur de très grands flux.

Au-delà du **temps** de calcul, le simple fait d'allouer une matrice $\{\omega_{i,j}\}$ pouvant atteindre des millions d'entités revient à gérer des téraoctets, ce qui n'est pas durable dans la plupart des scénarios. Une solution naive consistant à "tout garder" n'est donc pas réaliste pour de grands flux de données, et on doit imaginer des **palliatifs**.

On peut définir une politique où, passé un certain **âge** ou une mesure d'inactivité, les entités "dépassées" se voient **supprimées** du SCN. Concrètement, si une entité $\mathcal{E}_i$ n'a plus suscité de renforcement $\omega_{i,j}$ depuis un temps T, on considère ses liens comme obsolètes et on la retire. Sur le plan **mathématique**, ce "nettoyage" limite la taille n en l'empêchant de croître au-delà d'un certain seuil. Cela évite un emballement quadratique, mais fait **perdre** la mémoire de ces entités — ce qui peut gêner si elles redeviennent pertinentes.

Plutôt que d'entretenir un seul SCN global, on peut diviser les entités en plusieurs **sous-réseaux** thématiques, géographiques ou temporels. Chaque sous-SCN traite $O(m)$ entités, avec $m \ll n$. Cela confère un coût $O(m^2)$ par sous-réseau, ce qui peut rester gérable, tandis qu'un "méta-nœud" relie ou agrège les sous-SCN dans une architecture hiérarchique. On sacrifie en partie la

globalité (les entités de sous-SCN différents ne calculent pas directement ω), mais on préserve la scalabilité.

Une autre piste est de maintenir une **structure** de liens ω fortement sparse (voir chap. 7.2.3.3), en ne conservant que les plus grandes pondérations (k plus fortes liaisons par nœud) et en mettant à zéro les autres. Ainsi, l'**espace** mémoire consacré aux liens se ramène à $O(n \cdot k)$. On doit alors adapter l'algorithme DSL pour gérer ces mises à jour partielles.

A. Conséquence sur la Dynamique

Toute forme de suppression d'entités "anciennes" ou de fragmentation impacte la "pureté" du SCN, puisqu'on interrompt la possibilité de calculer $\omega_{i,j}$ pour des entités retirées ou séparées. Cela peut engendrer :

- **Perte** de synergies historiques (on ne peut plus relier $\mathcal{E}_{\text{new}}$ à une entité $\mathcal{E}_k$ supprimée même s'il existait potentiellement une corrélation).
- **Dilution** : scinder le SCN en sous-ensembles contrarie la libre émergence d'un cluster transversal.

D'un point de vue **mathématique**, on modifie la règle DSL en ne mettant à jour que **certaines** paires (i, j). On aboutit à une **dynamique** locale contrainte, plus parcimonieuse.

B. Avantages et Inconvénients

Parmi les **avantages**, la **scalabilité** devient un atout majeur, rendant la croissance illimitée de nn gérable grâce à des mécanismes de suppression d'entités, de distribution de charge via des sous-SCN ou encore de maintien d'une certaine sparsité. Un autre avantage réside dans la **concentration** sur l'actualité, car en ne conservant que les entités récentes ou actives, le système se focalise sur les synergies les plus pertinentes du moment, suivant une logique de **mémoire glissante**.

Cependant, ces mécanismes entraînent également des **inconvénients**. L'un des principaux est la **perte d'historique** : une fois supprimé, un cluster ou une entité ancienne ne peut plus être réactivé, même si le contexte futur justifierait son retour. De plus, la **complexité** liée à la segmentation ajoute un défi supplémentaire, notamment en matière de **paramétrage**. Il devient alors crucial de définir des critères précis pour la gestion des seuils, le découpage des sous-SCN et les politiques d'élimination des entités.

Conclusion

Le **coût computationnel** constitue un **défi** crucial pour un **SCN** en flux, car le nombre d'entités n peut croître sans limite, entraînant un coût en $O(n^2)$. Des solutions techniques s'imposent pour rester **viable** :

- **Suppression** ou "pruning" d'entités et de liaisons trop anciennes ou inactives,
- **Fragmentation** en sous-SCN, réduisant le problème par blocs,
- **Sparsification** poussée, en ne gardant qu'un nombre limité de liaisons par entité.

Ces dispositifs garantissent la **scalabilité** tout en maintenant une dynamique DSL incrémentale, même si l'on doit composer avec le **risque** de perdre certaines synergies passées ou de

restreindre l'interaction globale. Malgré ces compromis, ils se révèlent **indispensables** lorsque l'on vise un fonctionnement en flux potentiellement illimité.

9.9.1.2. Approches : suppression d'entités trop anciennes, fragmentation en sous-SCN, etc.

Lorsqu'un **SCN** (*Synergistic Connection Network*) opère en **flux continu**, le nombre d'entités $\{\mathcal{E}_i\}$ peut croître sans limite, rendant la matrice ω et sa mise à jour $\mathrm{O}(n^2)$ de plus en plus **lourdes**. Pour éviter cette dérive, on recourt à diverses **approches** visant à **réduire** la taille active du réseau ou à le **segmenter** afin de maintenir un coût raisonnable et une cohérence dynamique. Trois familles de solutions se distinguent :

1) Suppression d'entités trop anciennes

On définit un **score** d'obsolescence ou un "temps de vie" pour chaque entité $\mathcal{E}_i$. Si cette entité n'est plus du tout "active" (faible somme de liaisons, absence de synergie récente) ou qu'elle dépasse un certain **âge**, on la **supprime** du SCN, retirant ainsi la ligne/colonne correspondante dans ω.

Par exemple, on peut calculer un indicateur

$$O(i) = \alpha_0 \,\mathrm{Age}(i) - \alpha_1 \sum_j \omega_{i,j}(t),$$

où $\mathrm{Age}(i)$ est l'ancienneté de $\mathcal{E}_i$ et $\sum_j \omega_{i,j}(t)$ mesure sa participation courante. Si $O(i) > \theta$, l'entité $\mathcal{E}_i$ est jugée **obsolète** et retirée (ou fortement pénalisée). On peut aussi appliquer un "**decay**" accéléré de ses liaisons :

$$\omega_{i,j}(t+1) \;\leftarrow\; \omega_{i,j}(t)\,(1-\delta_i) \quad (\delta_i > 0),$$

jusqu'à extinction complète de $\omega_{i,j}$.

2) Fragmentation en sous-SCN

Si le SCN devient très gros ou très hétérogène, on peut le **scinder** en sous-réseaux plus compacts. Chaque sous-SCN gère un sous-ensemble $\mathcal{V}_a \subset \{1, \dots, n\}$ d'entités ayant des synergies internes plus fortes que leurs liens avec le reste.

On repère un découpage $\{\mathcal{V}_1, \dots, \mathcal{V}_m\}$ tel que, pour chaque sous-SCN, la sous-matrice $\boldsymbol{\Omega}_{\mathcal{V}_a,\mathcal{V}_a}$ reste cohérente, alors que les blocs "hors diagonale" $\boldsymbol{\Omega}_{\mathcal{V}_a,\mathcal{V}_b}$ (avec $a \neq b$) sont très faibles. Chaque sous-SCN évolue localement selon la règle DSL, limitant la complexité à $\mathrm{O}(m^2)$ par sous-réseau, plutôt que $\mathrm{O}(n^2)$. On peut définir un **niveau meta** reliant ces sous-SCN pour des interactions moins fréquentes.

3) Mécanismes "hybrides"

Au-delà de la suppression pure et simple ou de la fragmentation, on peut imaginer des **stratégies** mixtes :

- **Score d'activité** : conserver une entité $\mathcal{E}_i$ tant qu'elle contribue à un certain "score d'actualité" ; la **transférer** dans un sous-SCN "inactif" sinon.

- **Sparsification** stricte : ne garder, pour chaque entité, que les k plus fortes liaisons $\omega_{i,j}$ (chap. 7.2.3.3). Cela abaisse le coût en maintenant $O(n \cdot k)$ liens au lieu de $O(n^2)$.

Toutes ces approches concourent à la **scalabilité** du SCN en flux. La suppression d'entités obsolètes évite d'accumuler des nœuds sans synergie actuelle, allégeant le graphe et réduisant le coût $O(n^2)$. La fragmentation en sous-SCN autorise une gestion locale des entités fortement reliées entre elles, ce qui circonscrit les calculs à des blocs de taille plus modeste. Les mécanismes hybrides (scores d'activité, *k*-NN, passage en mode inactif) apportent une fine granularité dans la façon de gérer l'obsolescence, permettant une adaptation continue et sélective. On retient ainsi la mémoire récente et réellement utile, tout en limitant la surcharge quand le flux de données grossit.

Ces approches introduisent cependant une **perte** potentielle de la mémoire historique, puisque des entités supprimées ou isolées ne pourront plus se réactiver si jamais le contexte s'y prête de nouveau. La fragmentation en sous-SCN implique une séparation structurelle qui peut empêcher certaines entités de découvrir des synergies transverses. La suppression ou la sparsification doivent être calibrées (quel seuil ? quelle politique d'inactivation ?) pour ne pas amputer de possibles liaisons intéressantes. En outre, la gestion multi-SCN engendre une **complexité logique** (métaniveau à définir, transitions entre sous-réseaux, etc.).

Conclusion

Afin de **maîtriser** la croissance illimitée d'entités dans un **SCN** traitant un flux continu, on met en place des **approches** telles que (1) la **suppression** d'entités trop anciennes, (2) la **fragmentation** en sous-SCN, ou (3) des **mécanismes** hybrides (scores d'actualité, *k*-NN, etc.). Ces solutions **limitent** la dérive quadratique ($O(n^2)$) et **ciblent** les entités effectivement pertinentes, permettant au SCN de conserver un fonctionnement **viable** et **réactif** malgré la poursuite potentielle du flux dans le temps.

9.9.2. Contrôle Fin de la Plasticité

Le **DSL** (Deep Synergy Learning), lorsqu'il est maintenu dans un mode **ouvert** et évolutif (voir §§9.9.1), doit sans cesse ajuster la **plasticité** de la dynamique $\{\omega_{i,j}(t)\}$:

- Une **plasticité** trop élevée conduit à **détruire** aisément les **clusters** antérieurs, car les liens $\omega_{i,j}$ se remodèlent à l'excès dès qu'une nouveauté apparaît,

- Une **plasticité** trop faible, au contraire, provoque un **enfermement** : le système n'intègre pas efficacement les informations nouvelles (capteurs, entités, synergies), restreignant sa capacité d'adaptation.

C'est ce **dilemme** du "juste milieu" qui motive la section **9.9.2**. Nous discutons ici du **contrôle** (ou pilotage) de la plasticité, illustrant comment trouver un **équilibre** dynamique.

9.9.2.1. Difficile de trouver le “juste milieu” : si trop souple, on détruit les clusters antérieurs ; si trop rigide, on ignore les nouveautés

L’**apprentissage continu** dans un **SCN** (*Synergistic Connection Network*) implique un **dilemme** de **plasticité** : on veut pouvoir **réorganiser** la matrice $\{\omega_{i,j}\}$ pour intégrer des données **nouvelles**, mais sans effacer immédiatement l’information accumulée et les **clusters** préexistants. L’**équation** DSL typique :

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

met en exergue divers **paramètres** $(\eta, \tau, \Delta_{\text{ext}})$ qui influent directement sur le degré de **plasticité** ou de **rigidité**.

1. Paramétrages Contradictoires

Dans la règle DSL, η contrôle la vitesse d’**apprentissage** local : plus η est grand, plus on modifie $\omega_{i,j}$ dès qu’une nouveauté survient. À l’inverse, τ représente la **décroissance** ; un τ fort incite le SCN à oublier plus vite les synergies antérieures. Δ_{ext} peut inclure du **bruit** (recuit) ou un **signal** externe (feedback, inhibition) renforçant la plasticité.

Si η est énorme ou si Δ_{ext} (p. ex. un bruit stochastique) est trop ambitieux, alors la moindre **variation** de $S(i,j)$ cause un sursaut qui “casse” les **clusters** formés, les pondérations $\omega_{i,j}$ étant **déstabilisées** et remaniées de façon trop radicale. Les clusters passés se voient vite dissous, même s’ils pouvaient redevenir pertinents.

Si τ domine, ou si η est trop minuscule, la **dynamique** DSL ne se remet presque pas en question. Les **nouveaux** flux (entités $\mathcal{E}_{n+1}$ ...) ne parviennent pas à faire évoluer la structure ω. Les liens $\omega_{n+1,\dots}$ restent quasi nuls trop longtemps, et le SCN **ignore** la distribution actuelle.

2. Destruction vs. Ignorance

La **destruction** des anciens clusters se produit si la plasticité est trop haute : un petit arrivage de données “disparate” peut **fragiliser** le réseau existant et entraîner une redéfinition complète, alors qu’on n’aurait pas voulu sacrifier la connaissance historique.

L’**ignorance** survient au contraire si le SCN est trop **lent** à se réadapter : de nouveaux signaux n’arrivent pas à **gravir** la barrière d’inertie, la mise à jour $\Delta\omega$ reste trop faible, et le SCN demeure dans son attracteur usé, ratant l’opportunité de **capturer** une évolution thématique ou un nouveau sous-groupe.

Sur le plan **mathématique**, on peut quantifier ce phénomène via la somme de variations $\Delta\Omega(t) = \sum_{i,j} |\,\omega_{i,j}(t+1) - \omega_{i,j}(t)|$. Un **faible** $\Delta\Omega(t)$ signale un régime rigide, un **grand** $\Delta\Omega(t)$ illustre au contraire une volatilité risquée.

3. Recherche d’un Équilibre

On souhaiterait une **plasticité** “optimale”, où les **nouveautés** sont correctement intégrées, sans remettre à plat l’ensemble. Cela requiert souvent un **réglage** dynamique des paramètres (η, τ) ou de Δ_{ext}, afin que, par exemple, le réseau se montre plus réactif quand un **choc** ou une “nouveauté forte” est détectée, mais moins s’il s’agit d’un léger bruit.

(A) Indice Illustratif : Mémoire vs. Innovation

Pour **formaliser** l’arbitrage, on peut introduire deux indicateurs :

- **Mémoire** $\mathrm{Mem}(t)$: le degré de permanence des clusters initiaux.
- **Innovation** $\mathrm{Inv}(t)$: la proportion de liens récents ou de nouveaux clusters nés.

Un indicateur $\mathrm{Bal}(t) = \alpha\,\mathrm{Mem}(t) - \beta\,\mathrm{Inv}(t)$ peut guider la plasticité : si $\mathrm{Bal}(t) < 0$ (trop de nouveautés), on tempère η ou renforce τ ; si $\mathrm{Bal}(t)$ est trop élevé (on ne bouge plus), on injecte du **bruit** ou on augmente η.

Conclusion

Il est **délicat** de **trouver** le "juste milieu" en termes de **plasticité** dans un **SCN** évolutif : une config trop souple aboutit à la destruction des clusters antérieurs dès qu'un flux nouveau arrive, alors qu'une config trop rigide empêche la **détection** des nouveautés. Sur le plan **mathématique**, le compromis se manipule via les **paramètres** η (vitesse), τ (décroissance), Δ_{ext} (bruit, feedback), qu'il convient de **régler** ou d'**adapter** dynamiquement pour conserver une **mémoire** utile tout en restant ouvert aux entités et synergies fraîchement apparues.

9.9.2.2. Exploitation de signaux externes (ex. un indice de "nouveauté")

Dans le fonctionnement d'un **SCN** (*Synergistic Connection Network*) en **flux** évolutif, la dynamique auto-organisée (basée sur la synergie $S(i,j)$) peut être **complétée** par des **signaux** extérieurs, apportant des informations ou des consignes complémentaires. Un exemple marquant est l'**indice de "nouveauté"** : il sert à indiquer dans quelle mesure une entité ou un lien est *inédit*, apportant un contenu différent de l'existant. Ce **signal** peut influencer la manière dont la mise à jour $\omega_{i,j}$ se fait, afin de réagir plus vivement à de nouvelles entités ou, au contraire, calmer l'intégration si elles sont jugées peu novatrices.

A. Motif et Définition de l'Indice de Nouveauté

En apprentissage continu, certaines entités ou informations peuvent se révéler moins utiles si elles n'apportent rien de vraiment **nouveau** (déjà redondant avec l'existant), alors que d'autres entités fraichement arrivées pourraient être cruciales car elles couvrent un champ inexploré. Un **indice** de nouveauté, calculé "hors du SCN" (par un module externe, un algorithme d'évaluation, etc.), vient ainsi pondérer la façon dont le DSL traite chaque entité.

On peut définir un **score** $N(\mathcal{E}_i, t)$ pour chaque entité, reflétant sa **"nouveauté"** au temps t. Par exemple,

$$N(\mathcal{E}_i, t) = \Phi(\text{date d'arrivée}(\mathcal{E}_i),\ \text{redondance sémantique},\ \text{similarité aux entités existantes},\ \ldots).$$

Plus ce score est élevé, plus $\mathcal{E}_i$ est considérée **innovante** ou peu couverte par la structure actuelle. On peut alors en tenir compte lors de la mise à jour ω.

B. Incorporation du Signal Externe dans la Mise à Jour ω

Dans la règle DSL :

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

on peut **définir** $\Delta_{\mathrm{ext}}(i,j,t)$ comme dépendant de $N(\mathcal{E}_i, t)$ ou $N(\mathcal{E}_j, t)$. Par exemple, si l'on souhaite favoriser la liaison entre deux entités **toutes deux** jugées nouvelles, on peut ajouter un **terme** positif :

$$\Delta_{\text{ext}}(i,j,t) = \alpha \times N(\mathcal{E}_i,t) \times N\big(\mathcal{E}_j,t\big).$$

À l'inverse, si une entité $\mathcal{E}_i$ est vieille (faible nouveauté) et qu'on veut la pénaliser, on introduit un terme négatif.

Une autre version consiste à **pondérer** la synergie $S(i,j)$ par un facteur lié à la nouveauté :

$$\tilde{S}(i,j) = S(i,j) \times \big[1 + \beta\, N(\mathcal{E}_i,t) + \beta\, N\big(\mathcal{E}_j,t\big)\big].$$

Cela amplifie la synergie S dès lors qu'une entité est dite "innovante", accélérant la création de liens forts.

C. Intérêt Mathématique et Opérationnel

Évasion des Patterns Récurrents. En intégrant la **nouveauté**, on évite que le SCN se contente de répliquer les **synergies** déjà connues. On pousse davantage la mise à jour ω pour explorer des configurations qui impliquent ces nouvelles entités. C'est un peu comme injecter un bruit stochastique "orienté" : au lieu d'être aléatoire, le renforcement se trouve motivé par la note de nouveauté.

Faciliter l'Intégration. On ne veut pas qu'une entité récemment introduite reste trop longtemps **marginale**, faute de liens initiaux. Un fort "score de nouveauté" incite le DSL à **trouver** ou **construire** des liens ω, permettant la découverte de *clusters* adaptés à ce nouvel arrivant.

Recalibrage. La nouveauté n'est pas éternelle : si $\mathcal{E}_i$ s'intègre bien, elle perd peu à peu ce statut "nouveau". *Mathématiquement,* on fait décroître $N(\mathcal{E}_i,t)$ avec le temps, ou on le recalcule pour refléter la redondance accumulée. Ainsi, le SCN encourage initialement l'entité $\mathcal{E}_i$ à trouver sa place, puis la traite en entité "normale".

D. Limites et Configurations

Déterminer la "nouveauté" n'est pas triviale : base-t-on le score sur la date d'arrivée, la distance par rapport aux clusters existants, l'originalité sémantique ? Un mauvais design du signal peut engendrer un **excès** de perturbation (si on surestime la nouveauté) ou un bénéfice maigre (si la nouveauté est sous-évaluée).

Si on "sur-pondère" la nouveauté, le SCN peut devenir trop **instable** : un flux d'entités jugées "fraîches" agite en permanence la structure. On doit choisir un paramètre β (ou équivalent) contrôlant la force d'intégration de N.

Conclusion

L'exploitation d'un **signal externe** comme un **indice de "nouveauté"** (ou tout autre critère) peut grandement **enrichir** la dynamique DSL d'un **SCN** évolutif. On sort du seul cadre $S(i,j)$ pour prendre en compte un regard **extérieur** sur l'entité (nouveauté, confiance, priorité). *Mathématiquement,* on intègre ce signal dans la mise à jour ω en ajoutant un **terme** Δ_{ext} ou en pondérant S. Cela favorise :

- Une **intégration** plus rapide des entités jugées innovantes,
- Une **plasticité** mieux contrôlée (puisqu'on sait quand encourager ou freiner la redéfinition des liens),
- Un **pilotage** plus fin du SCN, où l'auto-organisation tient compte des dimensions hors-synergie (date, score, priorité), utiles en contexte industriel ou applicatif avancé.

9.9.3. Évaluation en Continu

Dans un **SCN** (Synergistic Connection Network) soumis à un flux permanent de données (ou d'entités), la question de l'**évaluation** ne se limite pas à une configuration statique. Au contraire, l'**architecture** DSL, en perpétuelle évolution (agrégation, division, réorganisation de clusters), exige des **métriques** capables de refléter la qualité instantanée ou glissante du partitionnement. Le **chapitre** 9.9.3 s'intéresse donc à la façon de **mesurer** en continu la cohésion et la pertinence des clusters, sans disposer d'un **ensemble de test fixe** (comme on pourrait l'avoir en apprentissage classique).

9.9.3.1. Mesurer la qualité des clusters en flux (pas d'ensemble de test fixe)

Dans un **SCN** (*Synergistic Connection Network*) évoluant en **flux**, l'évaluation **continue** de la qualité du clustering (ou de la structure du réseau) se heurte à deux difficultés : les **données** arrivent au fil du temps (pas de dataset complet à l'avance) et on ne dispose pas toujours d'un **ground-truth** (labels ou "vrais" clusters) pour comparer directement les résultats. La question devient alors : **comment** surveiller la "pertinence" des clusters au fil des itérations, de façon **incrémentale** et **non supervisée** ?

Lorsque l'on pratique un **clustering** *off-line,* on dispose généralement :

- D'un **ensemble** fini de données,
- Éventuellement d'un **ground-truth** (labels) permettant un calcul d'exactitude, de Rand index, etc.

Or, dans un **contexte flux** :

- Les entités $\{\mathcal{E}_t\}$ parviennent **progressivement**. Le SCN n'a pas d'ensemble complet pour calculer un score "global" immuable.
- Les clusters $\mathcal{C}_1^{(t)}, \dots, \mathcal{C}_k^{(t)}$ se modifient (fusionnent, se scindent, etc.). L'absence de **référence** (labels) empêche de valider en continu la justesse du regroupement.

Sur le plan **mathématique**, on veut donc des indicateurs internes (cohésion, séparation, modularité) qui puissent être **mis à jour** au fil du temps, reflétant l'état courant du réseau sans devoir tout "recalculer" ni disposer d'une base de vérité.

Les **indices** de clustering classiques (type Silhouette, Davies–Bouldin, modularité graphe) partent d'une partition fixée. Dans un SCN, la partition (clusters) fluctue à chaque itération t. On peut donc définir une **version** glissante ou incrémentale :

- On identifie la **composition** de chaque cluster $\mathcal{C}_\alpha^{(t)}$ à l'instant t (quitte à imposer un seuil de liaison pour dire "$\mathcal{E}_i$ appartient à $\mathcal{C}_\alpha$").
- On calcule un score de cohésion intra-cluster et de séparation inter-cluster, puis on obtient un **indice** global (ex. silhouette).

Exemple (Silhouette modifiée)
Pour chaque entité $i \in \mathcal{C}_\alpha^{(t)}$, on définit :

$$\text{cohésion}(i,t) = \frac{\sum_{(i,j)\in\mathcal{C}_\alpha^{(t)}} \omega_{i,j}^{(t)}}{|\mathcal{C}_\alpha^{(t)}|}, \quad \text{séparation}(i,t) = \min_{\beta\neq\alpha}\left[\frac{\sum_{(i,j)\in\mathcal{C}_\beta^{(t)}} \omega_{i,j}^{(t)}}{|\mathcal{C}_\beta^{(t)}|}\right].$$

Le **score** silhouette local :

$$s(i,t) = \frac{\text{séparation}(i,t) - \text{cohésion}(i,t)}{\max\{\text{séparation}(i,t),\ \text{cohésion}(i,t)\}}.$$

Un **score** global s'obtient en moyennant $s(i,t)$ sur tous les i. Cet indicateur, mis à jour à chaque itération, montre si les entités sont mieux regroupées (cohésion plus forte) ou si les clusters restent confus (séparation faible).

Pour éviter un **coût** $O(n^2)$ complet à chaque pas, on peut calculer l'indice **par incrément** quand une entité $\mathcal{E}_{\text{new}}$ arrive ou quand un lien $\omega_{i,j}$ change sensiblement. On met à jour :

- Les sommes intra-cluster,
- Les sommes inter-clusters,
- Les dénominations de clusters,
- Puis on recalcule le score local cohésion / séparation pour les entités affectées.

Cet **algorithme** incrémental permet de conserver un feedback continu sur la **qualité** du regroupement, sans rafraîchir tout le SCN.

A. Absence d'Ensemble de Test Fixe

Puisqu'il n'y a pas toujours de **ground-truth** (labels) disponible pour chaque entité en flux, on recourt à :

- Des **indicateurs internes** (Silhouette, modularité, Davies–Bouldin, etc.)
- Possiblement quelques **échantillons** ponctuels sur lesquels on dispose de labels pour un test supervisé partiel (contrôle sporadique).

La plupart du temps, on se contente d'évaluations **internes** : cohésion vs. séparation, ou gain de l'énergie $\mathcal{J}$ (si $\mathcal{J}$ est définie).

B. Indicateurs "Online" ou "Fenêtre Glissante"

On peut aussi définir un **cadre** "fenêtre glissante" : ne considérer que les entités $\{\mathcal{E}_{t-W+1}, \dots, \mathcal{E}_t\}$ pour évaluer la cohésion. Cela reflète l'état **courant** du flux, évite d'inclure des entités trop anciennes, et permet un calcul plus restreint.

Conclusion (9.9.3.1)

En **flux** continu, on ne dispose ni d'un **ensemble** de données figé ni toujours de **labels** pour guider l'évaluation des clusters. Il faut donc :

- **Employer** des **indices internes** (cohésion-séparation, modularité),
- Les **adapter** à la dynamique DSL (calcul glissant, incrémental),

- **Potentiellement** coupler à des fenêtres temporelles pour se concentrer sur les entités récentes.

Ces techniques assurent un **retour** sur la qualité du regroupement **au fil** de l'évolution du SCN, sans exiger de test statique ni de set de labels constants. En somme, elles permettent de **monitorer** la formation de clusters, leur stabilité et leur pertinence, même en présence d'un **flux** non supervisé et sans "vrai" ensemble de test.

9.9.3.2. Méthodes d'évaluation online : indices de cohésion glissants, etc.

Dans un **SCN** (*Synergistic Connection Network*) fonctionnant en flux continu, la structure des **clusters** et des **lignes** $\{\omega_{i,j}\}$ se reconfigure sans cesse au gré des nouvelles entités. Il est alors souhaitable de **surveiller** la qualité du regroupement de façon *continue* ou *en ligne*, plutôt que de réaliser un recalcul **complet** (hors-ligne) à chaque fois qu'une entité apparaît ou disparaît. Des **indices** de cohésion "glissants" et d'autres mesures similaires permettent d'évaluer **incrémentalement** la "pertinence" ou la "solidité" des clusters.

En mode **flux**, on ne dispose pas d'un jeu de données figé ni d'un ensemble de test complet, et la structure du **SCN** évolue en continu (fusions de clusters, nouvelles entités, dissolutions partielles). Pour **prévenir** une dérive ou une sur-fragmentation, on souhaite un **indice** qui indique si les regroupements sont toujours "cohérents" ou non. Ce **feedback** pourra alors déclencher des mécanismes correcteurs (par ex. un recuit partiel, une inhibition plus poussée, etc.), évitant la stagnation ou la prolifération de micro-clusters inefficaces.

Les indices classiques de **clustering** (Silhouette, Davies–Bouldin, modularité) partent d'un ensemble statique et d'une partition fixée. Dans un SCN continu, la partition peut n'être qu'**implicite** — on ne définit pas toujours un "cluster = liste d'entités", mais plutôt un ensemble de liaisons ω. Il faut donc :

- **Identifier** (ou extraire) les clusters courants (par ex. en prenant les connexions ω supérieures à un certain seuil).
- **Calculer** un score de cohésion–séparation sur ces groupes, tout en gérant le fait que les ω ont changé depuis la dernière itération.

Soit $\mathcal{C}_\alpha^{(t)}$ le cluster α au temps t. On peut définir :

$$\mathrm{Coh}\left(\mathcal{C}_\alpha^{(t)}\right) = \frac{1}{|\mathcal{C}_\alpha^{(t)}|^2} \sum_{i,j \in \mathcal{C}_\alpha^{(t)}} \omega_{i,j}\,(t),$$

plus la cluster est "compacte", plus la somme de ses liaisons internes est forte.

Pour évaluer la qualité, on confronte la cohésion d'un cluster à sa "distance" ou "faible liaison" aux autres groupes :

$$\mathrm{Sep}\left(\mathcal{C}_\alpha^{(t)}\right) = \min_{\beta \neq \alpha} \left[\mathrm{link}\left(\mathcal{C}_\alpha^{(t)}, \mathcal{C}_\beta^{(t)}\right)\right],$$

où $\mathrm{link}(\cdot,\cdot)$ évalue la somme (ou la moyenne) des ω inter-groupes. Plus cette somme est petite, plus la séparation est grande.

Pour **éviter** un recalcul $O(n^2)$ à chaque itération, on propose un **schéma incrémental** : on ne recalcule la cohésion ou la séparation que pour les clusters ou liaisons ω affectées par un changement significatif (nouvelles entités, liaisons modifiées). Ainsi, l'indice de cohésion "glisse" au fil du temps :

$$\mathrm{Coh}(t+1) = \mathrm{Coh}(t) + \Delta_{\mathrm{coh}}\big(\{\Delta\omega_{i,j}\}\big).$$

Si le flux est potentiellement infini, on peut n'évaluer la cohésion que sur les dernières entités (fenêtre Window_t) :

$$\mathrm{Coh}(t) = f\big(\{\omega_{i,j}(t)\}_{i,j\in\mathrm{Window}_t}\big), \quad \mathrm{Window}_t = \{t-W+1, \dots, t\}.$$

On peut y inclure uniquement les entités jugées "récentes" pour maintenir la pertinence et la complexité raisonnable.

Chaque insertion d'une entité $\mathcal{E}_{n+1}$ ou variation de $\omega_{i,j}$ déclenche :

- L'**ajout** dans un cluster (ou création d'un cluster),
- Un **changement** du score de cohésion / séparation pour ce cluster,
- Une **somme** ou **moyenne** globale pour tous les clusters, fournissant un indicateur final (ex. "score silhouette" ou "modularité instantanée").

Comme on ne possède pas toujours de **ground-truth** statique (labels), on utilise :

- **Indices internes** (Coh, Sep, modularité) pour juger du regroupement,
- **Occasionnellement**, si on a un lot de référence labellisée (même partiel), on peut comparer : "Le cluster formé correspond-il aux labels sur cette portion du flux ?".

Conclusion

Pour **évaluer** la qualité d'un **SCN** en flux — où les entités et les liaisons ω évoluent sans cesse —, on s'appuie sur des **méthodes** "online" :

5. **Indices de cohésion** (et séparation) glissants, calculés de manière **incrémentale** à chaque arrivée / retrait d'entités,
6. **Fenêtres** temporelles ou "mises à jour partielles" pour limiter la complexité,
7. **Indicateurs internes** plutôt que comparaisons à un ensemble de test figé.

De cette façon, on **surveille** en continu la **qualité** de la structure, on détecte la **formation** ou la **dissolution** de clusters, et on peut **agir** (recuit, inhibition, etc.) si l'on constate une dégradation. Cela fait partie intégrante du fonctionnement **adaptatif** et **non supervisé** d'un SCN évolutif.

9.9.4. Autres Pistes

Dans la continuité des approches d'**apprentissage continu** et de **synergie adaptative**, il est possible de renforcer encore la flexibilité et la capacité d'exploration du **DSL** (Deep Synergy

Learning) en le couplant à d'autres paradigmes ou mécanismes. La section 9.9.4 recense quelques pistes d'extension ou de collaboration.

9.9.4.1. Couplage avec un RL multi-agent permanent (repris de Ch. 7.7)

Un **SCN** (*Synergistic Connection Network*) qui évolue en flux s'adapte naturellement à la notion de **collaboration** (ou de **confrontation**) entre **agents** dans un scénario **multi-agent**. Au Chap. 7.7, on avait suggéré qu'un **Deep Synergy Learning (DSL)** peut être **couplé** à un **schéma** de renforcement (*Reinforcement Learning*, RL) pour tenir compte d'une **récompense** globale ou locale, et pas seulement de la synergie $S(i,j)$ intrinsèque. L'objectif est de faire cohabiter :

- La **dynamique DSL**, où la matrice ω évolue selon la synergie perçue ($S(i,j)$, éventuellement complétée par le signal de récompense),
- Un **RL multi-agent** où chaque agent A_i dispose d'une politique π_i visant à **maximiser** une récompense partagée $\mathcal{R}$ ou un ensemble de récompenses locales $\{\mathcal{R}_i\}$.

On peut se placer dans un **environnement** (ex. robotique coopérative, jeux multi-joueurs, réseau distribué) contenant un nombre N d'agents $\{A_1, \dots, A_N\}$. Chaque agent effectue des **actions**, reçoit une **récompense** plus ou moins synchronisée avec les autres. Parfois la récompense $\mathcal{R}(t)$ est **globale**, reflétant la performance collective ; parfois elle est **locale** $\{\mathcal{R}_i(t)\}$ pour chaque agent.

Sur le plan **SCN**, on peut assimiler chaque agent (ou un sous-ensemble de ses états / actions) à une entité $\mathcal{E}_i$. Les pondérations $\{\omega_{i,j}\}$ reflètent la **synergie** entre agents i et j : plus ils coopèrent efficacement, plus $\omega_{i,j}$ se renforce.

La mise à jour DSL typique :

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

peut être **modifiée** en intégrant la **récompense** $\mathcal{R}$. Par exemple :

$$\tilde{S}(i,j) = S(i,j) + \alpha\,\mathcal{R}(t)\,\delta_{ij}\,,$$

où δ_{ij} représente une fonction mesurant à quel point l'agent i et l'agent j ont contribué à l'action récompensée. Ainsi, quand $\mathcal{R}(t)$ est **positive**, le terme $\alpha\,\mathcal{R}(t)\,\delta_{ij}$ vient **augmenter** la synergie $\tilde{S}(i,j)$ et renforcer $\omega_{i,j}$. À l'inverse, un score $\mathcal{R}(t) < 0$ dévalorise la liaison qui a contribué à une mauvaise performance.

On obtient la **mise à jour** :

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

On peut voir la **DSL** comme un mécanisme visant à minimiser (ou maximiser) une fonction $\mathcal{J}\big(\{\omega_{i,j}\}\big)$ impliquant la synergie S et un terme de régularisation τ. L'ajout d'une **récompense** $\mathcal{R}$ injecte dans $\mathcal{J}$ un terme dépendant de $\mathcal{R}$: plus $\mathcal{R}$ est grande, plus on encourage les liaisons $\omega_{i,j}$ correspondantes. En parallèle, chaque agent suit une **politique** π_i pour maximiser la récompense.

On se retrouve avec un **cadre** multi-objectif : améliorer à la fois la structure de **coopération** (du point de vue DSL) et la **politique** (du point de vue RL). Cela ouvre la voie à une **co-évolution** :

- Les agents adaptent leur **politique** $\{\pi_i\}$ pour accroître la récompense,
- Le SCN (le "réseau" de ω) se **modifie** pour promouvoir les liens entre agents "bénéfiques",
- Le système gagne ainsi en **robustesse** et en **exploration** : un agent isolé pourrait se voir "poussé" à collaborer si ω favorise sa synergie avec un autre agent lui donnant accès à des stratégies plus rémunératrices.

Boucle RL–DSL

Côté RL
Chaque agent choisit ses actions selon π_i, reçoit $\mathcal{R}(t)$. On peut utiliser Q-learning, policy gradient, etc. pour mettre à jour la **politique**.

Côté DSL
On met à jour ω en continu, en calculant $\tilde{S}(i,j)$ intégrant la récompense partagée $\mathcal{R}(t)$. Ainsi, si deux agents i et j ont **coopéré** pour générer un gain, leurs liaisons $\omega_{i,j}$ se renforcent, les encourageant à réitérer cette collaboration.

Il s'agit d'une **co-adaptation** : d'une part, les politiques $\{\pi_i\}$ évoluent pour exploiter la structure ω, d'autre part, la structure ω se "moule" sur les réussites/échecs constatés.

Ce couplage DSL–RL multi-agent permanent confère plusieurs bénéfices. D'abord, la **coopération** n'est pas seulement dictée par une mesure de similarité S, mais aussi par l'**efficacité** démontrée (récompense $\mathcal{R}$). Cela encourage une **exploration** plus riche : les liaisons ω se construisent non seulement sur la synergie perçue, mais aussi sur le résultat concret des actions entreprises conjointement. De plus, la structure évolutive du SCN soutient un **apprentissage** flexible : si la récompense change (environnement modifié), ω se réoriente, appuyant une collaboration différente. En somme, la boucle RL–DSL favorise l'**émergence** dynamique de coalitions d'agents mieux adaptées à l'état actuel du monde, surpassant un RL isolé ou un DSL sans finalité de récompense.

6. Conclusion

Le **couplage** d'un **SCN** évolutif avec un **RL** multi-agent permanent étend la logique du DSL, en y incorporant la notion de **récompense**. Concrètement, cela crée un mécanisme où les liens $\omega_{i,j}$ se **renforcent** non seulement selon la synergie $S(i,j)$, mais aussi en proportion du gain réel $\mathcal{R}$ obtenu. *Mathématiquement*, on modifie la fonction de synergie ou la descente d'énergie pour prendre en compte $\mathcal{R}$. Les agents s'**adaptent** via RL, et la **structure** (pondérations ω) s'adapte via DSL : on obtient un **réseau vivant** d'agents qui auto-organise leur coopération (ou compétition) en temps réel. Cela ouvre la voie à de riches applications (robotique coopérative, systèmes distribués) où l'on veut un **apprentissage** collectif piloté à la fois par la **ressemblance** (synergie) et par la **performance** (récompense).

9.9.4.2. ART + DSL : synergie entre vigilance adaptative et auto-organisation continue

L'**Adaptive Resonance Theory** (*ART*) met en avant l'idée de **vigilance** : un paramètre ρ qui exige qu'un nouveau pattern **x** satisfasse un **degré** minimum de similarité (ou de recouvrement) avec un prototype existant avant de s'y rattacher ; sinon, on crée un **nouveau** cluster. Le **DSL** (Deep Synergy Learning), quant à lui, se fonde sur la **dynamique** d'un **SCN** (*Synergistic Connection Network*), dans lequel les **liens** $\{\omega_{i,j}\}$ se reforment sans cesse selon la synergie $S(i,j)$. Combiner **ART** et **DSL** revient à insérer le **mécanisme** de vigilance (ρ) au sein d'une **auto-organisation** continue (mise à jour $\omega_{i,j}$), apportant un contrôle plus **top-down** sur la formation, la fusion et la scission des clusters.

A. Rappel sur ART : Vigilance Adaptative

Dans ART, chaque catégorie est définie par un **prototype** $\mathbf{w}_k$. Lorsqu'on présente un nouveau pattern **x**, on cherche un $\mathbf{w}_k$ tel que :

$$\frac{\| \mathbf{x} \wedge \mathbf{w}_k \|}{\| \mathbf{x} \|} \geq \rho,$$

ou une variante. Si cette **vigilance** ρ n'est pas atteinte, **x** provoque la **création** d'un nouveau prototype ou la révision d'un existant. La valeur de ρ peut s'adapter au fil du temps : *trop haute* ρ fragmente beaucoup, *trop basse* ρ agglomère trop.

B. DSL : Auto-Organisation Continue via Synergie

Dans le DSL, on ne fixe pas un seuil de similarité a priori : on laisse la **synergie** $S(i,j)$ influencer la mise à jour :

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

Les pondérations ω s'accumulent ou faiblissent, façonnant progressivement un **cluster** (ou sous-ensemble) quand plusieurs entités $\mathcal{E}_i, \mathcal{E}_j$ affichent une synergie élevée. Si des entités arrivent en flux, la structure s'ajuste sans reset complet, *mais* on n'a pas un mécanisme explicite de "vigilance" pour refuser une fusion si les entités sont trop différentes.

C. Combiner ART et DSL : Vigilance Adaptative dans l'Auto-Organisation

Introduire la **vigilance** : on impose qu'avant de renforcer $\omega_{i,j}$ (ou avant d'intégrer $\mathcal{E}_i$ dans un cluster existant), un certain **seuil** ρ de similarité (au sens d'ART) soit franchi. Ainsi, on contrôle la **fusion** de sous-clusters : si $\mathcal{E}_{\text{new}}$ est jugée insuffisamment proche d'un cluster, on ne "force" pas $\omega_{(\text{new}),\,j}$ à augmenter, on préfère créer un nouveau regroupement (nouveau proto).

ART suggère que ρ peut **varier** selon la situation : on rehausse ρ si on constate qu'un cluster devient trop large (surdilué), on l'abaisse si on observe trop de micro-clusters. Ceci se combine parfaitement avec la règle DSL, où ω continue de se mettre à jour localement, mais ne dépassera pas un certain seuil si ρ n'est pas satisfait.

D. Gains pour l'Auto-Organisation Continue

La vigilance ρ agit comme un garde-fou : si elle est trop basse, on pourrait noyer le SCN dans un "mega-cluster" ; si elle est trop haute, on crée trop de petits clusters. En DSL pur, on se contente souvent d'un paramètre τ ou d'un seuil $\omega_{\min}$, mais pas d'un critère "top-down" vérifiant la **similarité** explicite. La vigilance ART sert donc de **verrou** supplémentaire, évitant la confusion des entités trop dissemblables.

L'**ART** est connue pour sa capacité à traiter les données *à la volée*, sans devoir réentraîner tout un modèle. Couplée à la **mise à jour** ω, on obtient un SCN qui **grandit** et **scinde** ses clusters en respectant un niveau de similitude imposé. Ceci peut s'avérer crucial si l'on veut éviter la contamination d'un cluster par des entités qui n'y ont pas vraiment leur place.

Si, avec le temps, la distribution change, la vigilance (paramètre ρ) peut se réajuster pour forcer ou éviter certaines fusions : dès qu'on perçoit trop de disparités internes, on augmente ρ, poussant la scission ; si au contraire tout le SCN se morcelle, on diminue ρ. L'**ART** confère ainsi un niveau de contrôle plus précis sur la hiérarchie des regroupements.

Conclusion

Le **mélange** ART + DSL, c'est la **vigilance** adaptative (concept phare d'ART) insérée dans la **dynamique** d'**auto-organisation** (propres au DSL). On y gagne :

- **Un critère** explicite de fusion/scission ("vigilance" ρ)
- **Une dynamique** continue et incrémentale (règle DSL, ajout de liens ω)
- **Un contrôle** plus fin : ni sur-fusion (clusters trop gros) ni sur-fragmentation (clusters trop nombreux).

Les **clusters** émergent tout en tenant compte d'un **seuil** de similitude minimal, ajusté au fil du flux. Ce couplage ouvre la voie à un **système** plus résilient et plus précis, combinant la force de l'**auto-organisation** DSL et le **contrôle** d'ART sur la formation stable de catégories.

9.10. Conclusion et Ouverture

Au terme de ce **chapitre 9**, nous avons exploré l'idée d'un **DSL** (Deep Synergy Learning) capable de s'adapter en continu, à travers des mécanismes incrémentaux et des protocoles temps réel (introduits en sections précédentes). Cette approche vise à gérer la **plasticité** du réseau, la reconfiguration automatique de ses liens $\omega_{i,j}$ et l'insertion ou la suppression dynamique d'entités $\mathcal{E}_i$. Nous concluons ici en récapitulant les apports du chapitre (9.10.1), en exposant les liens vers les prochains chapitres (9.10.2) et en ouvrant des **perspectives** plus larges (9.10.3).

9.10.1. Récapitulatif du Chapitre

9.10.1.1. Mise en évidence de la dimension continue du DSL, gérant flux et évolutions sans repasser par un calcul offline

Dans ce chapitre, il a été souligné que l'un des **piliers** du Deep Synergy Learning (DSL), appliqué à un **SCN** (*Synergistic Connection Network*) évolutif, est sa **capacité** à gérer de manière **continue** l'afflux de nouvelles données ou entités. Contrairement aux approches traditionnelles de type "batch" où, dès qu'un lot de données s'achève, on ré-entraîne le modèle de façon complète et on le fige ensuite, le DSL peut mettre à jour *incrementalement* la matrice $\{\omega_{i,j}\}$ sans recourir à un "recalcul offline". Cette **plasticité** s'illustre par la **formule** de mise à jour, complétée par divers mécanismes (inhibition, recuit) permettant au SCN de s'adapter en temps réel.

Dimension Continue et Flux

Le SCN peut être vu comme un **réseau vivant**, dans lequel les **pondérations** $\omega_{i,j}(t)$ changent à chaque itération t. Dès qu'une entité $\mathcal{E}_{n+1}$ arrive, on initialise ses liaisons $(\omega_{(n+1),j})$ et on laisse la **règle** :

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

faire évoluer le réseau localement. Ainsi, au lieu d'effectuer un recalcul massif de toute la structure, on **intègre** progressivement la nouvelle entité, ajustant les liaisons ω de manière ciblée. Sur le plan **mathématique**, cela crée une **suite** $\{\omega(t)\}_{t=0,1,\dots}$ qui ne subit jamais de réinitialisation globale ; au contraire, elle "glisse" dans l'espace des configurations.

Pas de Recalcul Offline

Dans de nombreux algorithmes classiques, dès que les données changent ou s'agrandissent, on réalise un **nouvel entraînement** offline (batch) : on reclasse toutes les entités, on reconstruit un clustering de zéro, etc. Le DSL rompt avec cette logique : il suffit de définir **localement** les liaisons ω pour l'entité $\mathcal{E}_{n+1}$, puis de laisser la boucle DSL "propager" cette information. En **flux**, on gagne un énorme avantage de **continuité** :

- **Insertion d'entités** : on ajoute $\mathcal{E}_{n+1}$ avec $\omega_{(n+1),\cdot} \approx 0$; la suite DSL augmente ces ω si $\mathcal{E}_{n+1}$ partage des synergies élevées avec certaines entités.
- **Poursuite de la mise à jour** : on ne fige pas le modèle, on ne rejoue pas toutes les étapes ; on modifie simplement la **topologie** $\{\omega\}$ au fil de l'eau.

Cet aspect incrémental se montre crucial en **temps réel** ou face à des **données massives** qui affluent sans discontinuer.

Stratégies Clés Présentées

Pour faire fonctionner cette mise à jour continue de façon **viable** et **cohérente**, on a rappelé diverses **stratégies** :

- **Mise à jour incrémentale** : ne modifier que les pondérations ω affectées par l'arrivée ou le départ d'entités, plutôt que de recalculer la totalité.
- **Insertion/suppression** d'entités : gérer la création de liens $\omega_{(n+1),\,j}$ avec une initialisation (p. ex. un voisinage k-NN) et éventuellement retirer les entités obsolètes.
- **Contrôle de la plasticité** : inhibition plus ou moins forte, recuit local en cas d'enlisement, ajustement du taux η si le flux devient plus important ou si la cohésion interne diminue, etc.

4. Synthèse

Au final, la **dimension continue** du DSL permet un **SCN** qui ne cesse de se **reconstruire** à chaque itération, sans interrompre ou re-lancer un entraînement offline dès qu'un changement survient. C'est ce qui rend la **philosophie** DSL si adaptée aux environnements dynamiques, où l'on veut allier un **apprentissage** permanent (incrémental) à la **stabilité** des connaissances déjà apprises. Les **techniques** décrites dans ce chapitre offrent alors un **cadre** théorique et pratique pour absorber un flux potentiellement infini de nouvelles entités, ajuster la structure ω en permanence, et faire évoluer les **clusters** sans jamais repasser par un *recomputing* général.

9.10.1.2. Stratégies clés : mise à jour incrémentale, insertion/suppression d'entités, contrôle de la plasticité (inhibition, recuit, "fenêtre glissante"), etc.

Dans un **SCN** (*Synergistic Connection Network*) soumis à des évolutions **en continu**, des mécanismes spécifiques doivent être mis en œuvre pour conserver une **auto-organisation** cohérente et éviter la saturation ou la stagnation. Les **stratégies** majeures concernent :

- La **mise à jour incrémentale** des pondérations ω,
- L'**insertion** et la **suppression** d'entités selon leur pertinence,
- Le **contrôle** de la **plasticité** via inhibition, recuit ou fenêtres glissantes.

Chacune de ces approches se décline en formules mathématiques ou algorithmes concrets, assurant la **viabilité** du SCN dans un contexte flux.

A. Mise à jour incrémentale

La "**règle**" DSL $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \cdots$ peut être mise en œuvre **localement** : quand un nouveau flux (ou une nouvelle entité $\mathcal{E}_{n+1}$) arrive, on initialise $\omega_{(n+1),j} \approx 0$ et on **évalue** la synergie $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ pour un voisinage restreint (k plus proches entités, etc.). De cette manière, on **épargne** un recalcul exhaustif $O(n^2)$.

Avantages :

- On maintient la **continuité** du réseau : les **clusters** et liens préexistants restent en place.
- On ne “repasse” pas par un entraînement batch complet.

Limites :

- On doit définir un **voisinage** pertinent, au risque sinon de passer à côté de certaines synergies plus lointaines.

B. Insertion / Suppression d’entités

Insertion
Lorsqu’une entité $\mathcal{E}_{n+1}$ se présente, on calcule sa synergie avec un sous-groupe d’entités (kNN), on met à jour $\omega_{(n+1),j}$. Si, au fil de la dynamique, ces liens se renforcent, $\mathcal{E}_{n+1}$ s’insère dans un cluster ; sinon, elle reste isolée ou fonde un nouveau regroupement.

Suppression
Pour éviter l’encombrement, on peut retirer des entités obsolètes. Celles-ci voient leurs pondérations $\omega_{i,j}$ décroître graduellement (par un “**decay**” imposé), jusqu’à un seuil near 0, après quoi on les élimine. Cette **méthode** freine la croissance ininterrompue du SCN et se justifie si l’entité n’a plus d’activité ou de synergie depuis longtemps.

C. Contrôle de la plasticité

Pour ne pas sombrer dans une volatilité excessive (où chaque nouveau flux réécrit toute la structure), on régule la **plasticité** :

Inhibition. On peut introduire un terme d’inhibition latérale pour **limiter** le nombre de liaisons fortes par nœud. *Mathématiquement*, un schéma $\omega_{i,j}(t+1) = \omega_{i,j}(t) + \eta\big[S(i,j) - \tau\,\omega_{i,j}(t) - \gamma \sum_{k\neq j} \omega_{i,k}\,(t)\big]$ force l’entité $\mathcal{E}_i$ à **sélectionner** seulement quelques liens dominants.

Recuit (bruit stochastique). Si la dynamique DSL reste figée dans un minimum local, on injecte un **terme** $\sigma(t)\,\xi_{i,j}$. Cela secoue la structure, permettant de franchir des barrières d’énergie et de réarranger le SCN pour s’aligner sur un nouveau flux. On peut moduler σ (température) selon un protocole (périodique, adaptatif, etc.).

Fenêtre glissante. Si la **synergie** S dépend de l’historique (par ex. co-occurrences passées), on peut adopter une “fenêtre” de taille W. On ne garde que les interactions ayant eu lieu dans la fourchette $[t-W, t]$. Les liens forgés sur des interactions “anciennes” se diluent et disparaissent, ce qui **favorise** l’adaptation lorsque le flux évolue.

Avantages :

- Ces mécanismes empêchent l’explosion de liaisons, empêchent aussi la stagnation. Ils offrent un **contrôle** fin sur la vitesse d’évolution du SCN.

Limites :

- Un paramétrage délicat (quelle intensité de l'inhibition, quelle amplitude de bruit recuit, quelle taille de fenêtre) est requis pour trouver un bon compromis entre stabilité et flexibilité.

Conclusion

Pour qu'un **SCN** en flux demeure **opérationnel**, il doit :

- **Mettre à jour** les liaisons ω **incrémentalement**, sans reconstruction globale.
- Gérer **l'insertion** ou la **suppression** d'entités, maintenant ainsi une dimension et un contenu de réseau pertinents.
- Réguler la **plasticité** via l'inhibition, le recuit (évitant l'enlisement) ou une fenêtre glissante (oublier l'ancien).

Chacune de ces stratégies se **traduit** par des formules (locales ou globales) et des algorithmes d'**implémentation**, assurant un SCN toujours "vivant" et capable de s'adapter au **flux** sans un recalcul offline systématique.

9.10.2. Liens vers Chapitres Suivants

Après avoir exploré l'**incrémentation avancée**, la **persistance** et les **cas d'usage** du DSL en temps réel, il est utile de se projeter vers les **chapitres** ultérieurs, qui approfondiront plusieurs dimensions complémentaires : la **coordination** multi-niveau (Chap. 10) et la **robustesse** globale du système (Chap. 11). Les points ci-dessous explicitent les **connexions** entre ces thèmes et les développements abordés dans ce chapitre 9.

9.10.2.1. Chap. 10 : Feedback Coopératif** dans le DSL**

La perspective introduite dans les sections antérieures (voir **Chap. 9** sur la gestion d'un **flux** en **temps réel**) ouvre la voie à une problématique plus générale : la capacité du **Deep Synergy Learning (DSL)** à intégrer des **boucles de rétroaction** et à orchestrer un **feedback coopératif** qui transcende la simple auto-organisation locale. Le **Chapitre 10** développe précisément cette idée d'une **dynamique multi-niveau**, où un *macro-nœud* ou un *niveau supérieur* d'abstraction vient **influencer** la structure des clusters inférieurs, de sorte à maintenir ou à réorienter la configuration globale du réseau selon des **objectifs** évolutifs.

A. Émergence et Flux d'Information dans un Contexte Multi-Niveau

Le **DSL** se conçoit souvent comme un **SCN** (*Synergistic Connection Network*) dont la mise à jour $\omega_{i,j}$ (voir références des sections précédentes, notamment 9.10.1) traduit la **synergie** entre entités $\mathcal{E}_i$ et $\mathcal{E}_j$. Lorsque ce réseau est confronté à un **flux continu** de données (référence **Chap. 9**), il se réorganise en permanence, créant ou dissociant des **clusters**. Néanmoins, on peut vouloir un mécanisme de **cohérence globale**. Le **Chap. 10** montrera comment un *palier macro* ou un *meta-nœud* injecte un **feedback** top-down, guidant la formation des liens dans un **sens coopératif**.

Matériellement, on suppose que le **DSL** gère plusieurs **couches** ou **strates**, avec un niveau "macro" plus abstrait (qui peut être un super-nœud englobant un cluster complet, ou un module de pilotage global). Ce niveau n'est pas inerte : il **analyse** la configuration courante, détecte les signaux critiques (par exemple la performance d'ensemble, la cohérence thématique) et envoie en retour un **signal** de correction ou d'encouragement à certains sous-ensembles d'entités.

B. Feedback Coopératif et Boucles d'Information

Contrairement à une approche strictement bottom-up, où la dynamique ω se base essentiellement sur la **synergie** perçue localement, on introduit ici la possibilité de **rétroaction**. Sur le plan mathématique, la mise à jour $\omega_{i,j}$ n'est plus seulement

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

mais inclut un **terme** coopératif $\Delta_{\text{coop}}(i,j,t)$ (cf. chap. 7.7 pour l'analogie RL, ou chap. 9.7 pour l'extension). Ce terme, qui peut dépendre d'un **signal** global (p. ex. la satisfaction d'un palier macro, la productivité d'un cluster, la nécessité de rapprocher deux sous-groupes), oriente ω vers un arrangement plus **collectivement** optimal.

Ainsi, la *boucle d'information* ne se limite plus à "calculer la synergie S localement et réajuster ω". Le *niveau macro* supervise ou module l'**évolution** de ω en émettant un "ordre" qui peut s'assimiler à un coefficient additionnel, ou à une inhibition latérale plus agressive pour certains segments, ou encore à un "boost" pour certains liens promus comme critiques pour la tâche globale.

C. Corrélation avec l'Apprentissage Continu du Chap. 9

Les chapitres précédents (référence à **9.10.1** ou **9.9.4**) décrivaient la manière dont un SCN gère un **flux** d'entités, en insérant ou supprimant des nœuds, et en contrôlant la **plasticité**. Cette dimension continue demeure, mais le **Chap. 10** enrichit l'architecture d'une **coordination** multi-niveau. On ne se contente pas d'un "réseau plat" : des *super-nœuds* (clusters d'entités) ou un "sous-réseau" de contrôle surplombent la structure, envoyant des **feedback**s plus complexes.

D. Intérêt Théorique et Pratique

Il devient possible de **stabiliser** plus rapidement un cluster si le palier macro détecte son utilité (p. ex. un groupe sensoriel crucial en robotique). À l'inverse, si un regroupement se révèle improductif ou si un changement de contexte l'a rendu obsolète, le macro-nœud peut "**casser**" certaines liaisons en injectant un signal d'inhibition accentué. Mathématiquement, la formule devient :

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

où $\Delta_{\text{coop}}^{(\text{macro})}$ traduit la consigne de feedback. Une telle **articulation** combine la **spontanéité** de l'auto-organisation DSL (basée sur la synergie) et la **conscience** d'un niveau supérieur, ce qui rend le SCN encore plus adaptable aux variations du flux, car il reçoit en permanence des **informations** multiples (local synergy + global feedback).

Avantages

Dans un seul paragraphe, on retient que l'approche de **feedback coopératif** multi-niveau accroît la **robustesse** et la **coordination** : elle prévient les dérives purement locales, met en œuvre un **objectif** ou une **vision** plus haute (ex. performance globale, indicateurs

macroscopiques), et assure une **vitesse** de reconvergence supérieure quand le flux amène de nouveaux défis. Les clusters détectés ne sont plus simplement le fruit de la synergie brute, mais reflètent également les priorités imposées par le niveau macro, ce qui permet une adaptation plus sémantique ou plus utilitaire.

Limites, Inconvénients

Néanmoins, introduire un palier macro suppose un **module** additionnel, capable d'émettre un feedback fiable. Cela alourdit la conception (on doit définir la logique de ce niveau) et peut parfois **contrarier** la dynamique locale en imposant des liaisons qui ne respectent pas la simple synergie S. Le paramétrage de $\Delta_{\text{coop}}^{(\text{macro})}$ peut aussi mener à des oscillations si le feedback est trop réactif ou s'il manque de consistance avec les signaux locaux. Enfin, l'efficacité dépend de la **qualité** du signal global, qui doit être cohérent avec les intentions du SCN et les données effectives.

Conclusion

Le **Chapitre 10** approfondira donc ce **feedback coopératif**, montrant comment un SCN en flux peut s'appuyer sur des boucles multi-niveaux, où un *macro-nœud* ou un *niveau supérieur* pilote (encourage, inhibe) la formation ou la fusion de clusters. Cela prolonge la logique d'**adaptation continue** du DSL (chap. 9) en y incorporant une **vue** plus holistique et coordonnée, afin de répondre aux finalités globales du système (robotique, conversation, collaboration multi-agent). Ainsi, on entre dans un **cadre** où l'auto-organisation n'est plus strictement locale (synergie S) mais **orientée** par des signaux extérieurs, assurant une **convergence** plus fiable et plus rapide vers une structure performante.

9.10.2.2. Chap. 11 : Robustesse, Sécurité** et **Fiabilité** (suite logique pour gérer les perturbations temps réel et menaces)

Le **Deep Synergy Learning (DSL)**, tel qu'il a été étudié tout au long de ce chapitre (et plus largement dans la section dédiée à l'apprentissage en flux **Chap. 9**), se retrouve confronté à diverses formes de **perturbations** ou de "menaces" susceptibles de compromettre la stabilité ou la pertinence des **clusters**. L'**intérêt** de consacrer un Chap. 11 entier à la **robustesse**, à la **sécurité** et à la **fiabilité** est donc évident : une fois que l'on maîtrise la capacité du **SCN** (*Synergistic Connection Network*) à évoluer en temps réel et à se reconfigurer, il faut garantir que cette dynamique ne tombe pas sous l'emprise de données manipulées ou de pannes entraînant une dégradation irréversible de la structure $\{\omega_{i,j}\}$.

A. Continuité avec la Gestion Temps Réel

La **gestion temps réel** (voir **Chap. 9**) montre que le **SCN** est sans cesse en **restructuration** : de nouvelles entités peuvent arriver en continu, d'autres être supprimées, et le réseau met à jour ses liaisons $\omega_{i,j}$ localement ou globalement. Dans un tel contexte, il importe de veiller à ce que ces **mises à jour** restent fiables, même si des agents malveillants insèrent de la fausse information dans le flux, ou si des capteurs défectueux produisent un **bruit** anormalement élevé. Il devient donc nécessaire de détecter, isoler, voire bloquer ces perturbations. **Mathématiquement**, cela peut prendre la forme de **filtres** sur ω, où l'on définit des fonctions de "confiance" ou de "cohérence" par rapport à la synergie attendue. Cela peut aussi se traduire par une **analyse d'anomalies** : on étudie la distribution $\{\omega_{i,j}\}$ ou $\{S(i,j)\}$ pour repérer les valeurs trop extrêmes et les neutraliser.

B. Menaces et Vulnérabilités dans un DSL en Évolution

Un **flux** d'entités n'est pas toujours "innocent" : certaines données peuvent être délibérément **corrompues** ou trompeuses. Le **Chap. 11** vise à recenser les méthodologies de robustesse. Un **SCN** en flux continu, si ses liaisons ω se réajustent trop docilement, peut se retrouver confondu par des informations parasites, voire se faire "hacker" pour forger des **clusters** fictifs (sorte de "poisoning attack"). Le **Chap. 11** explorera comment évaluer la **confiance** de chaque entité $\mathcal{E}_i$, comment potentiellement **verrouiller** ou restreindre la mise à jour $\omega_{i,j}$ en cas de suspicion. L'autre aspect est la **panne partielle** ou la **défaillance** d'un sous-SCN : si un module ou un bloc d'entités cesse de répondre de manière cohérente, la structure globale doit s'en accommoder, isolant le module ou réduisant son impact pour maintenir une cohésion sur le reste du réseau.

C. Cohérence avec la Suite de l'Ouvrage

Le **Chap. 11** se place comme un prolongement naturel : après avoir maîtrisé la **plasticité** du SCN et son fonctionnement en **flux**, on en tire qu'un DSL "vierge" n'est pas à l'abri de manipulations ou de dysfonctionnements. Les approches de **robustesse** (détection d'anomalies, validations croisées de la synergie, éventuels protocoles de signature ou de cryptographie) viennent consolider la fiabilité du réseau. On y traite aussi la **tolérance aux fautes** : dans un **SCN** distribué, la panne d'un sous-réseau ne doit pas conduire à l'effondrement total du graphe. **Mathématiquement**, on pourra modéliser la **résilience** par des règles de "délégation" ou de "duplication" : si un nœud $\mathcal{E}_i$ tombe, un autre nœud $\mathcal{E}_k$ prend le relais, maintenant ainsi la connectivité.

Conclusion

L'élargissement vers la **robustesse** et la **sécurité** (*Chap. 11*) apparaît comme la suite logique : après avoir conféré au **SCN** un haut degré de **plastique** (Chap. 9–10), il importe d'assurer que cette **adaptation** ne soit pas exploitable par des intrusions malveillantes ou des pannes en cascade. Le **DSL**, pour être déployé dans des environnements critiques ou fortement évolutifs, doit être muni de dispositifs de **contrôle** de l'intégrité de la structure $\{\omega\}$, de **fiabilité** des flux, et de résistance aux pannes. En d'autres termes, la **dimension** "temps réel" et "réorganisation continue" (cf. Chap. 9) se complète par une "**dimension** robustesse et sécurité" qui sera approfondie dans le **Chapitre 11**, englobant toutes les techniques visant à rendre le **SCN** capable de se **défendre** ou de se **ressaisir** malgré des perturbations imprévues.

9.10.3. Perspectives

À l'issue de ce chapitre dédié aux **évolutions temps réel** et à l'adaptation continue, il apparaît clairement que le **DSL** (Deep Synergy Learning) doit se doter de mécanismes suffisamment flexibles pour **survivre** et **prospérer** dans des environnements **dynamiques**, où de nouvelles entités (capteurs, modules, données) apparaissent sans cesse, tandis que d'autres disparaissent ou changent de comportement. Avant de conclure (9.10.4) et d'enchaîner avec la suite du livre, nous mettons en avant quelques **pistes** clés pour la poursuite de la recherche et de la mise en œuvre de solutions **auto-organisées** de nouvelle génération.

9.10.3.1. Rôle central de l'apprentissage continu dans de nombreux domaines (IoT, robotique, streaming massifs)

Dans la logique d'un **Deep Synergy Learning (DSL)** appliqué à un **SCN** (*Synergistic Connection Network*) fonctionnant sur des **flux** de données, il apparaît que l'**apprentissage continu** (ou "incrémental") devient une pièce maîtresse pour bon nombre de secteurs. Les exemples de l'**Internet des Objets**, de la **robotique** ou du **traitement de streams massifs** illustrent pleinement le besoin de gestion adaptative en temps réel, où l'on ne peut se permettre de ré-entraîner un modèle complet dès qu'un nouvel élément (capteur, entité, message) arrive. Les **sections** et **chapitres** précédents (chap. 9 en particulier) ont souligné les fondements de cette approche, démontrant comment le SCN se module localement sans reconduire tout un calcul *offline*.

A. Environnements hautement évolutifs

L'**IoT** illustre un scénario où des milliers, voire des milliards, d'objets connectés (capteurs, appareils) interagissent dans un réseau en croissance permanente. Un **SCN** classique ne suffirait pas : il faudrait reconstruire la matrice $\{\omega_{i,j}\}$ à chaque ajout d'un nouveau node, engendrant un coût $O(n^2)$. À l'inverse, l'**apprentissage continu** du DSL, basé sur des mises à jour $\omega_{(n+1),j}$ localisées et incrémentales, s'intègre sans rupture de service. Sur le plan **mathématique**, on peut, pour chaque capteur $\mathcal{E}_{n+1}$, borner le calcul de la synergie $S(\mathcal{E}_{n+1}, \mathcal{E}_j)$ à un **voisinage** restreint (k-NN, par ex.). Cela évite la re-parcours complet du graphe et renforce la **scalabilité** de la solution.

En **robotique**, qu'il s'agisse de flottes de drones, de véhicules ou de robots mobiles, l'aspect temps réel prime. Chaque fois qu'un robot détecte une zone inexplorée ou subit une panne partielle, l'**apprentissage continu** permet au SCN de réattribuer (ou dévaluer) des liaisons ω et de forger de nouveaux clusters sensorimoteurs. Les *formules* de mise à jour locales, éventuellement complétées par un terme de "recuit local" (pour sortir d'un minimum local) ou une "inhibition adaptative" (pour limiter la prolifération de liaisons), garantissent un **réseau** plus agile et autonome.

Enfin, pour le **streaming** massif (Big Data, logs, flux de réseaux sociaux), on ne peut reclassifier toutes les données à chaque incrément. L'**approche** DSL continue offre une **auto-organisation** incrémentale : chaque nouvel élément s'intègre selon la synergie S qu'il partage avec des entités proches. Ce fonctionnement "glissant" soulage la mémoire (pas besoin de conserver tout l'historique) et évite le retraining.

B. Conséquences et Intérêt

L'**apprentissage continu** offre d'abord une grande **évolutivité** (il n'y a pas de mur $O(n^2)$ tant qu'on limite la mise à jour au voisinage local). Il assure aussi une **résilience** : un module tombé, ou des capteurs bruyants, peuvent être isolés localement sans détruire la structure globale. Cela se traduit *mathématiquement* par l'extinction progressive de liens $\omega_{i,j}$ qui ne sont plus "alimentés" en synergie.

L'**adaptation** perpétuelle se révèle cruciale dans un flux aux patterns changeants. Les entités en place peuvent **reformuler** leurs liens, les clusters obsolètes se réduisent à néant (ou se scindent), de nouvelles agrégations de nœuds voient le jour. Le DSL n'a pas besoin d'arrêt, illustrant la notion de "**réseau vivant**" dont le SCN est la structure.

Avantages

D'un seul tenant, on constate que l'**apprentissage continu** dans un DSL assure une **mise à jour incrémentale** évitant la re-construction complète, favorise la **scalabilité** (particulièrement pour IoT ou Big Data), et soutient une **résilience** propre à la robotique (où les environnements varient instantanément). Il fluidifie la **dynamique** du SCN : à tout instant, on traite juste ce qui est **nécessaire**. Il en découle un gain en **rapidité**, un confort de déploiement (pas de "temps mort" pour re-entraîner), et une meilleure **continuité** de service.

Limites, Inconvénients

Néanmoins, la forte **plasticité** peut introduire des risques de dérive si la distribution évolue de manière trop brutale (c'est pourquoi un recuit partiel ou des mécanismes d'inhibition/filtrage sont souvent nécessaires). De plus, on doit paramétrer soigneusement la portée locale des mises à jour, sous peine d'accumuler des erreurs ou de rater des synergies globales. Enfin, la suppression de données anciennes pour maintenir la "fraîcheur" peut faire perdre un savoir historique si le flux repasse par un contexte déjà rencontré, mais qui a été "effacé".

Conclusion (9.10.3.1)

La **dimension** *"apprentissage continu"* constitue un **fondement** crucial pour aborder IoT, robotique coopérative et streaming massifs, où la **nature** même des données évolue sans cesse. Les principes du DSL, tels que la **mise à jour** incrémentale, la régulation locale via l'inhibition ou le recuit, et l'éventuelle gestion par "fenêtre glissante", permettent à un **SCN** d'évoluer "en direct", de *s'auto-organiser* face à des nouveautés ou des changements sans requérir un long entraînement batch ni un "reset" permanent de la structure. Cette **philosophie** s'avère décisive pour les applications en temps réel, se prêtant à un déploiement dans des environnements hautement **dynamiques**, massifs, et exigeant un fonctionnement **ininterrompu**.

9.10.3.2. Importance de perfectionner les heuristiques (inhibition adaptative, recuit local) pour conserver un SCN cohérent face à l'évolution constante

Lorsqu'un **SCN** (*Synergistic Connection Network*) évolue dans un contexte de flux continu (ou d'environnement changeant), l'**auto-organisation** risque de se retrouver constamment déstabilisée, car l'arrivée de nouvelles entités ou la modification de la synergie S imposent des remaniements successifs. Les **heuristiques** d'**inhibition adaptative** et de **recuit local** (parmi d'autres) se révèlent alors incontournables pour assurer une **cohérence** persistante : elles régulent la **plasticité** de la structure $\{\omega_{i,j}\}$ tout en évitant soit une dérive perpétuelle, soit un enlisement rigide.

A. Contrainte : Évolution Constante et Cohérence Long Terme

Dans un **DSL** (Deep Synergy Learning), on manipule les pondérations $\omega_{i,j}$ par des règles locales. Si l'environnement est statique, la convergence vers un certain état stable ou un ensemble d'attracteurs est concevable. En revanche, quand le SCN fait face à un **flux** ininterrompu d'entités (arrivées et retraits) ou à un changement continu de la fonction de synergie S, cette convergence peut être constamment mise en péril. La principale difficulté est donc de conserver une **cohérence globale** tout en restant *ouvert* à des réagencements locaux :

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

Si cette équation reste "nue", on risque soit un basculement incessant de ω dès qu'une nouvelle entité apparaît, soit un gel prématuré si τ domine ou si l'apprentissage est trop faible.

B. Inhibition Adaptative : Contrôle Dynamique de la Concurrence

L'**inhibition latérale** ou la "concurrence hebbienne" offre un mécanisme permettant de **limiter** le nombre total de liaisons significatives pour chaque entité $\mathcal{E}_i$. Dans un contexte flux, si trop de liens moyens se créent au fil du temps, l'architecture se "pollue" : la lecture des clusters devient confuse. On introduit alors un **terme** dans la mise à jour :

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

où $\gamma(t)$ peut **varier** au cours du temps, permettant une **inhibition adaptative**. Par exemple, si la densité moyenne de liens devient trop forte, on **augmente** γ pour accentuer la compétition, forçant les entités à sélectionner des connexions vraiment pertinentes plutôt que de conserver de nombreux liens "tièdes". Cela garantit une meilleure **lisibilité** des clusters.

C. Recuit Local : Mini-chauffe contre l'Enlisement

Le **recuit simulé** global (cf. chap. 7.3) consiste en un bruit $\xi_{i,j}$ d'amplitude $\sigma(t)$, injecté dans la mise à jour ω. Mais sur un SCN en flux, on ne veut pas nécessairement recuire *l'ensemble* du réseau en continu, ce qui pourrait s'avérer coûteux ou potentiellement trop instable. On pratique alors un **recuit local**, c'est-à-dire qu'on "shakes" (brasse stochastiquement) seulement un **sous-ensemble** de liens $\omega_{i,j}$ lors d'une phase critique (arrivée d'un bloc de nouvelles entités, suspicion d'enlisement local). Mathématiquement, on ajoute :

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

pour (i,j) dans un voisinage $\mathcal{V}_{\text{cible}}$. Cela permet de **sortir** d'un puits local, de briser des configurations sous-optimales, tout en évitant un bruit global permanent.

D. Rôle Clé pour la Cohérence dans la Durée

Lorsque de **nouvelles entités** arrivent régulièrement, ou que la **synergie** subit des changements contextuels, le SCN est en "équilibre dynamique" plus qu'en convergence stricte. Les heuristiques — *inhibition adaptative* et *mini recuits* localisés — s'avèrent décisives pour :

- **Éviter la prolifération** d'un trop grand nombre de liens moyens,
- **Réactiver** de l'exploration (via le recuit) lorsque des configurations sclérosées ne correspondent plus à la réalité.

Les **formules** d'ajustement $\gamma(t)$ ou $\sigma(t)$ peuvent dépendre de **mesures** comme la densité de liens, la détection d'un "score" de cohérence en baisse, ou la détection de stagnation (voir chap. 9.9).

Avantages

Ces deux mécanismes — inhibition adaptative et recuit local — préservent la **flexibilité** et la **cohérence** du SCN à long terme. L'inhibition adaptative empêche la surcharge de liens superflus, tandis que le recuit local offre la possibilité de rompre un **enlisement** local, tout en évitant un recuit global coûteux. La complémentarité de ces heuristiques, appliquées de façon ciblée, stabilise l'**auto-organisation** en flux : on gagne en **robustesse** face aux perturbations incessantes, tout en maintenant la capacité de réorganiser les clusters quand la situation l'impose.

Limites, Inconvénients

Le calibrage de l'inhibition et du recuit local n'est pas trivial : un excès d'inhibition rend le réseau trop fragmenté, un défaut d'inhibition provoque une confusion généralisée. Un recuit trop fréquent ou trop intense aboutit à une instabilité chronique, alors qu'une rareté d'injection de bruit risque de laisser le SCN coincé dans un état sous-optimal. De plus, la mise en œuvre d'une **inhibition dynamique** ou d'un recuit local exige la définition de critères déclencheurs, augmentant la complexité algorithmique.

Conclusion

Dans un **DSL** évoluant de manière continue, les **heuristiques** — notamment l'**inhibition adaptative** et le **recuit local** — demeurent indispensables pour **préserver** un SCN **cohérent**. L'inhibition adaptative contrôle la concurrence entre liens et clusters, évite la profusion de connexions moyennes, tandis que le recuit local (mini-chauffe ciblée) prévient l'enlisement et encourage une réorganisation si le flux induit un changement majeur. Avec ces mécanismes, le SCN parvient à maintenir une **plasticité** suffisante pour absorber les évolutions constantes, sans sombrer dans le désordre ou la rigidité.