# Métaheuristiques pour l'optimisation difficile

Johann Dréo - Alain Pétrowski
Patrick Siarry - Éric Taillard
Ouvrage coordonné par Patrick Siarry


ISBN : 2-212-11368-4
EYROLLES

# Chapitre 4

# Les algorithmes de colonies de fourmis

## 4.1 Introduction

Les algorithmes de colonies de fourmis forment une classe des métaheuristiques récemment proposée pour des problèmes d'optimisation difficile. Ces algorithmes s'inspirent des comportements collectifs de dépôt et de suivi de piste observés dans les colonies de fourmis. Une colonie d'agents simples (les *fourmis*) communiquent indirectement via des modifications dynamiques de leur environnement (les *pistes de phéromones*) et construisent ainsi une solution à un problème en s'appuyant sur leur expérience collective.

Le premier algorithme de ce type (le "Ant System[1]") a été conçu pour le problème du voyageur de commerce, mais n'a pas permis de produire des résultats compétitifs. Cependant, l'intérêt pour la métaphore était lancé et de nombreux algorithmes s'en inspirant ont depuis été proposés, certains atteignant des résultats très convaincants.

Ce chapitre insiste dans un premier temps (section 4.2) sur l'aspect biologique sous-tendant ces algorithmes, car il nous semble intéressant de mettre en perspective la conception et l'utilisation de cette métaheuristique avec les théories biologiques qui l'ont inspirée. La section 4.3 décrit en détail le premier algorithme de colonies de fourmis et quelques-unes de ses principales variantes, puis nous présentons quelques pistes pour découvrir la grande variété d'adaptations possibles de ces algorithmes (section 4.4). Nous étudions ensuite les principes de fonctionnement de la métaheuristique dans la section 4.5, et les perspectives de recherche dans le domaine (section 4.6). Après une conclusion sur l'ensemble du chapitre (section 4.7) nous proposons une bibliographie commentée pour approfondir le sujet (section 4.8).

[1] traduisible simplement par "Système de Fourmis"

## 4.2 Comportements collectifs des insectes sociaux

### 4.2.1 Auto-organisation et comportement

#### 4.2.1.1 Auto-organisation

L'*auto-organisation* est un phénomène décrit dans plusieurs disciplines, notamment en physique et en biologie. Une définition claire a été proposée [Camazine *et al.* 00, p.8] :

> L'auto-organisation est un processus dans lequel un *modèle* de niveau global *émerge* uniquement d'un grand nombre d'interactions entre les composants de *bas niveau* du système. De plus, les règles spécifiant les interactions entre composants du système sont suivies en utilisant uniquement des informations locales, sans référence au *modèle* global.

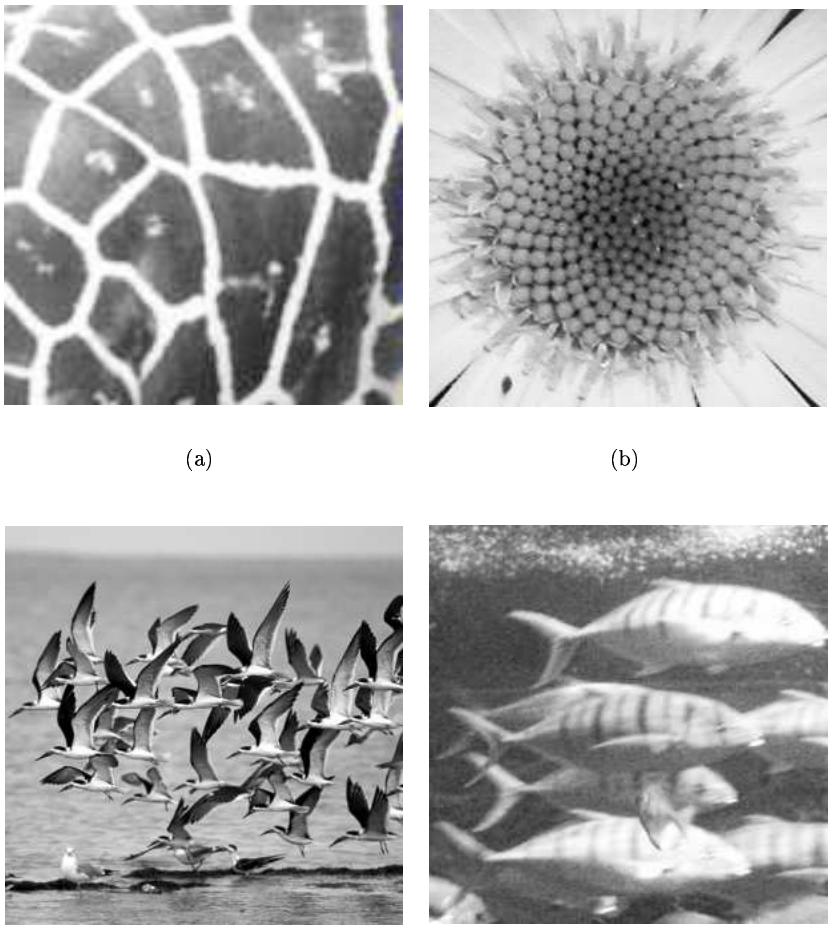
Deux termes sont à préciser pour une bonne compréhension, "modèle" et "émerger". Le mot *modèle* est une traduction approximative du mot anglais "pattern", qui déborde la notion de structure, et peut signifier aussi configuration générale, forme, schéma, type. D'une manière générale, il s'applique à un "arrangement organisé d'objets dans l'espace ou le temps" (figure 4.1). Une propriété *émergente* d'un système est quant à elle une caractéristique qui apparaît à l'imprévu (sans avoir été *explicitement* déterminée), de par les interactions entre les composants de ce système.

La question cruciale est donc de comprendre comment les composants d'un système interagissent entre eux pour produire un modèle complexe (au sens relatif du terme, i.e. *plus* complexe que les composants eux-mêmes). Un certain nombre de phénomènes nécessaires ont été identifiés : ce sont les processus de *rétroaction* et la gestion des *flux d'informations*.

Les *rétroactions positives* sont des processus dont le résultat renforce l'action, par exemple par amplification, facilitation, auto-catalyse, etc. Les rétroactions positives sont capables d'amplifier les *fluctuations* du système, permettant la mise à jour d'informations peu apparentes. De tels processus peuvent facilement entraîner une explosion du système, s'ils ne sont pas maintenus sous contrôle par des *rétroactions négatives*, qui jouent ainsi le rôle de stabilisateurs du système. Lorsqu'ils sont couplés, de tels processus de rétroaction sont de puissants générateurs de modèles.

Dans le cadre de la biologie du comportement, il est aisé de comprendre que les interactions entre les composants d'un système vont très souvent mettre en jeu des processus de *communication*, de transfert d'informations entre individus. D'une manière générale, les individus peuvent communiquer, soit par le biais de signaux, c'est-à-dire en utilisant un moyen spécifique pour porter une information, soit par le biais d'indices, où l'information est portée accidentellement. De même, l'information peut provenir directement d'autres individus, ou bien passer par le biais de l'état d'un travail en cours. Cette deuxième possibilité d'échanger des informations, par le biais de modifications de l'environnement, se nomme la *stigmergie*.

D'une manière générale, tous ces processus sont plus ou moins inter-connectés, permettant à un système constitué d'un grand nombre d'individus agissant de concert de résoudre des problèmes trop complexes pour un individu unique.



(a) (b)

(c) (d)

FIG. 4.1 – *Exemples de* modèles *observables dans des systèmes biologiques. (a) motifs de la robe d'une girafe réticulée (U.S. Fish and Wildlife Service, Gary M. Stolz), (b) double spirale de Fibonacci au coeur d'une pâquerette, (c) groupe d'oiseaux en vol, (d) poissons assemblés en banc.*

Certaines caractéristiques des systèmes auto-organisés sont particulièrement intéressantes, en particulier leur *dynamisme*, ou encore leur capacité à produire des modèles *stables*. Dans le cadre de l'étude du comportement des insectes sociaux, certains concepts liés au principe de l'auto-organisation méritent d'être soulignés : la *décentralisation* intrinsèque de ces systèmes, leur organisation en *hétérarchie dense* et l'utilisation récurrente de la *stigmergie*. En effet, ces concepts sont parfois utilisés comme autant d'angles de vue différents sur un même problème et recouvrent une partie des principes de l'auto-organisation.

#### 4.2.1.2 Stigmergie

La stigmergie est un des concepts à la base de la création des métaheuristiques de colonies de fourmis. Elle est précisément définie comme une "forme de communication passant par le biais de modifications de l'environnement", mais on peut rencontrer le terme "interactions sociales indirectes" pour décrire le même phénomène. Les biologistes différencient la stigmergie "quantitative" de celle "qualitative", mais le processus en lui-même est identique. Un exemple d'utilisation de la stigmergie est décrit dans la section 4.2.2. La grande force de la stigmergie est que les individus échangent des informations par le biais du travail en cours, de l'*état* d'avancement de la tâche globale à accomplir.

#### 4.2.1.3 Contrôle décentralisé

Dans un système auto-organisé, il n'y a pas de prise de décision à un niveau donné, suivie d'ordres et d'actions pré-déterminées. En effet, dans un système décentralisé, chaque individu dispose d'une vision *locale* de son environnement, et ne connaît donc pas le problème dans son ensemble. La littérature des systèmes multi-agents (voir [Ferber 97] pour une première approche) emploie souvent ce terme ou celui "d'intelligence artificielle distribuée" [Jennings 96], bien que, d'une manière générale, cette discipline tende à utiliser des modèles de comportements plus complexes, fondés notamment sur les sciences de la cognition. Les avantages d'un contrôle décentralisé sont notamment la *robustesse* et la *flexibilité* [Bonabeau *et al.* 99]. Systèmes robustes, car capables de continuer à fonctionner en cas de panne d'une de leurs parties ; flexibles, car efficaces sur des problèmes dynamiques.

#### 4.2.1.4 Hétérarchie dense

L'hétérarchie dense est un concept issu directement de la biologie [Wilson *et al.* 88], utilisé pour décrire l'organisation des insectes sociaux, et plus particulièrement des colonies de fourmis. Le concept d'hétérarchie décrit un système où les propriétés des niveaux globaux agissent plus ou moins sur les propriétés des niveaux locaux, mais également un système où une activité dans les unités locales influence en retour les niveaux globaux. L'hétérarchie est dite *dense* dans le sens où un tel système forme un réseau hautement connecté, où chaque individu peut échanger des informations avec n'importe quel autre. Ce concept est en quelque sorte opposé à celui de *hiérarchie* où, dans une vision populaire mais erronée, la reine gouvernerait ses sujets en

faisant passer des ordres dans une structure *verticale*, alors que, dans une *hétérarchie*, la structure est plutôt *horizontale* (figure 4.2).

(a) (b)

FIG. 4.2 – *Hiérarchie (a) et hétérarchie dense (b) : deux concepts opposés.*

On constate que ce concept recoupe celui de contrôle décentralisé, mais aussi celui de stigmergie, en ce sens que l'hétérarchie décrit la *manière* dont le flux d'information parcourt le système. Cependant, dans une hétérarchie dense, tout type de communication doit être pris en compte, tant la stigmergie que les échanges directs entre individus.

### 4.2.2 Optimisation naturelle : pistes de phéromones

Les algorithmes de colonies de fourmis sont nés à la suite d'une constatation : les insectes sociaux en général, et les colonies de fourmis en particulier, résolvent naturellement des problèmes relativement complexes. Les biologistes ont étudié comment les fourmis arrivent à résoudre collectivement des problèmes trop complexes pour un seul individu, notamment les problèmes de choix lors de l'exploitation de sources de nourriture.

Les fourmis ont la particularité d'employer pour communiquer des substances volatiles appelées *phéromones*. Elles sont très sensibles à ces substances, qu'elles perçoivent grâce à des récepteurs situés dans leurs antennes. Ces substances sont nombreuses et varient selon les espèces. Les fourmis peuvent déposer des phéromones au sol, grâce à une glande située dans leur abdomen, et former ainsi des pistes odorantes, qui pourront être suivies par leurs congénères (figure 4.3).

FIG. 4.3 – *Des fourmis suivant une piste de phéromone.*

Les fourmis utilisent les pistes de phéromones pour marquer leur trajet, par exemple entre le nid et une source de nourriture. Une colonie est ainsi capable de choisir (sous certains conditions) le plus court chemin vers une source à exploiter [Goss *et al.* 89, Beckers *et al.* 92], sans que les individus aient une vision *globale* du trajet.


FIG. 4.4 – *Expérience de sélection des branches les plus courtes par une colonie de fourmis : (a) au début de l'expérience, (b) à la fin de l'expérience.*

En effet, comme illustré sur la figure 4.4, les fourmis le plus rapidement arrivées au nid, après avoir visité la source de nourriture, sont celles qui empruntent les deux branches les plus courtes. Ainsi, la *quantité* de phéromone présente sur le plus court trajet est légèrement plus importante que celle présente sur le chemin le plus long. Or, une piste présentant une plus grande concentration en phéromones est plus attirante pour les fourmis, elle a une *probabilité* plus grande d'être empruntée. La piste courte va alors être plus renforcée que la longue, et, à terme, sera choisie par la grande majorité des fourmis.

On constate qu'ici le choix s'opère par un mécanisme d'*amplification* d'une fluctuation initiale. Cependant, il est possible qu'en cas d'une plus grande quantité de phéromone déposée sur les grandes branches, au début de l'exploitation, la colonie choisisse le plus long parcours.

D'autres expériences [Beckers *et al.* 92], avec une autre espèce de fourmis, ont montré que si les fourmis sont capables d'effectuer des demi-tours sur la base d'un trop grand écart par rapport à la direction de la source de nourriture, alors la colonie est plus flexible et le risque d'être piégé sur le chemin long est plus faible.

Il est difficile de connaître avec précision les propriétés physico-chimiques des pistes de phéromone, qui varient en fonction des espèces et d'un grand nombre de paramètres. Cependant, les métaheuristiques d'optimisation de colonies de fourmis s'appuient en grande partie sur le phénomène d'*évaporation* des pistes de phéromone. Or, on constate dans la nature que les pistes s'évaporent plus lentement que ne le prévoient les modèles. Les fourmis réelles disposent en effet "d'heuristiques" leur apportant un peu plus d'informations sur le problème (par exemple une information sur la direction). En effet, il faut garder à l'esprit que l'intérêt immédiat de la colonie (trouver le plus court chemin vers une source de nourriture) peut être en concurrence avec l'intérêt *adaptatif* de tels comportements. Si l'on prend en compte l'ensemble des contraintes que subit une colonie de fourmis (prédation, compétition avec d'autres colonies, etc.), un choix rapide et stable peut être meilleur, et un changement de site exploité peut entraîner des coûts trop forts pour permettre la sélection naturelle d'une telle option.

## 4.3 Optimisation par colonies de fourmis et problème du voyageur de commerce

Le problème du voyageur de commerce ("Travelling Salesman Problem", *TSP*) a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis : le "Ant System" (*AS*) [Colorni *et al.* 92]. Le passage de la métaphore à l'algorithme est ici relativement facile à faire et le problème du voyageur de commerce est bien connu et étudié. Il est intéressant d'approfondir le principe de ce premier algorithme pour bien comprendre le mode de fonctionnement des algorithmes de colonies de fourmis. Il y a deux façons d'aborder ces algorithmes. La première, la plus évidente au premier abord, est celle qui a historiquement mené au "Ant System" original ; nous avons choisi de la décrire dans cette section. La seconde est une description plus formelle

des mécanismes communs aux algorithmes de colonies de fourmis, elle sera décrite dans la section 4.5.

Le problème du voyageur de commerce consiste à trouver le trajet le plus court reliant $n$ villes données, chaque ville ne devant être visitée qu'une seule fois. Le problème est plus généralement défini comme un graphe complètement connecté $(N, A)$, où les villes sont les noeuds $N$ et les trajets entre ces villes, les arêtes $A$.

### 4.3.1 Algorithme de base

Dans l'algorithme *AS*, à chaque itération $t$ ($1 \leq t \leq t_{max}$), chaque fourmi $k$ ($k = 1, \ldots, m$) parcourt le graphe et construit un trajet complet de $n = |N|$ étapes (on note $|N|$ le cardinal de l'ensemble $N$). Pour chaque fourmi, le trajet entre une ville $i$ et une ville $j$ dépend de :

1. la liste des villes déjà visitées, qui définit les mouvements possibles à chaque pas, quand la fourmi $k$ est sur la ville $i$ : $J_i^k$ ;
2. l'inverse de la distance entre les villes : $\eta_{ij} = \frac{1}{d_{ij}}$, appelée *visibilité*. Cette information statique est utilisée pour diriger le choix des fourmis vers des villes proches, et éviter les villes trop lointaines ;
3. la quantité de phéromone déposée sur l'arête reliant les deux villes, appelée l'*intensité de la piste*. Ce paramètre définit l'attractivité d'une partie du trajet global et change à chaque passage d'une fourmi. C'est en quelque sorte une mémoire globale du système, qui évolue par apprentissage.

La règle de déplacement (appelée "règle aléatoire de transition proportionnelle" par les auteurs [Bonabeau *et al.* 99]) est la suivante :

$$p_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha \cdot (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t))^\alpha \cdot (\eta_{ij})^\beta} & \text{si } j \in J_i^k \\ 0 & \text{si } j \notin J_i^k \end{cases} \tag{4.1}$$

où $\alpha$ et $\beta$ sont deux paramètres contrôlant l'importance relative de l'*intensité* de la piste, $\tau_{ij}(t)$, et de la *visibilité*, $\eta_{ij}$. Avec $\alpha = 0$, seule la visibilité de la ville est prise en compte ; la ville la plus proche est donc choisie à chaque pas. Au contraire, avec $\beta = 0$, seules les pistes de phéromone jouent. Pour éviter une sélection trop rapide d'un trajet, un compromis entre ces deux paramètres, jouant sur les comportements de *diversification* et d'*intensification* (voir section 4.5.3 de ce chapitre), est nécessaire.

Après un tour complet, chaque fourmi laisse une certaine quantité de phéromones $\Delta\tau_{ij}^k(t)$ sur l'ensemble de son parcours, quantité qui dépend de la *qualité* de la solution trouvée :

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L^k(t)} & \text{si } (i,j) \in T^k(t) \\ 0 & \text{si } (i,j) \notin T^k(t) \end{cases} \tag{4.2}$$

où $T^k(t)$ est le trajet effectué par la fourmi $k$ à l'itération $t$, $L^k(t)$ la longueur du tour et $Q$ un paramètre fixé.

L'algorithme ne serait pas complet sans le processus d'*évaporation* des pistes de phéromone. En effet, pour éviter d'être piégé dans des solutions sous-optimales, il est nécessaire de permettre au système "d'oublier" les mauvaises solutions. On contrebalance donc l'additivité des pistes par une décroissance constante des valeurs des arêtes à chaque itération. La règle de mise à jour des pistes est donc :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{4.3}$$

où $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^{k}(t)$ et $m$ est le nombre de fourmis. La quantité initiale de phéromone sur les arêtes est une distribution uniforme d'une petite quantité $\tau_0 \geq 0$.

La figure 4.5 présente un exemple simplifié de problème du voyageur de commerce optimisé par un algorithme AS dont le pseudo-code est présenté sur l'algorithme 4.1.

**Algorithme 4.1** Algorithme de colonies de fourmis de base : le "Ant System".

**Pour** $t = 1, \ldots, t_{max}$

  **Pour** chaque fourmi $k = 1, \ldots, m$

    **Choisir** une ville au hasard

    **Pour** chaque ville non visitée $i$

      **Choisir** une ville $j$, dans la liste $J_i^k$ des villes restantes, selon la formule 4.1

    **Fin** Pour

    **Déposer** une piste $\Delta\tau_{ij}^{k}(t)$ sur le trajet $T^k(t)$ conformément à l'équation 4.2

  **Fin** Pour

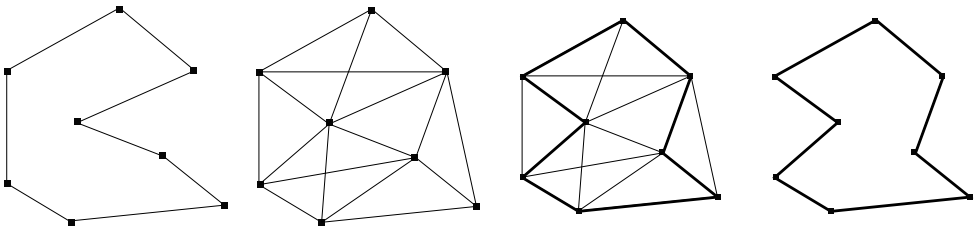
  **Évaporer** les pistes selon la formule 4.3

**Fin** Pour

## 4.3.2 Variantes

### 4.3.2.1 *Ant System* & élitisme

Une première variante du "Système de Fourmis" a été proposée dans [Dorigo *et al.* 96] : l'introduction de fourmis "élitistes". Dans cette version, la meilleure fourmi (celle qui a effectué le trajet le plus court) dépose une quantité de phéromone plus grande, dans l'optique d'accroître la probabilité des autres fourmis d'explorer la solution la plus prometteuse.



(a) (b) (c) (d)

FIG. 4.5 – *Le problème du voyageur de commerce optimisé par l'algorithme AS, les points représentent les villes et l'épaisseur des arêtes la quantité de phéromone déposée. (a) exemple de trajet construit par une fourmi, (b) au début du calcul, tous les chemins sont explorés, (c) le chemin le plus court est plus renforcé que les autres, (d) l'évaporation permet d'éliminer les moins bonnes solutions.*

#### 4.3.2.2 *Ant-Q*

Dans cette variante de $AS$, la règle de mise à jour locale est inspirée du "Q-learning[2]" [Gambardella *et al.* 95]. Cependant, aucune amélioration par rapport à l'algorithme $AS$ n'a pu être démontrée. Cet algorithme n'est d'ailleurs, de l'aveu même des auteurs, qu'une pré-version du "Ant Colony System".

#### 4.3.2.3 *Ant Colony System*

L'algorithme "Ant Colony System"[3] ($ACS$) a été introduit pour améliorer les performances du premier algorithme sur des problèmes de grandes tailles [Dorigo *et al.* 97b, Dorigo *et al.* 97a]. $ACS$ est fondé sur des modifications du $AS$ :

1. $ACS$ introduit une règle de transition dépendant d'un paramètre $q_0$ ($0 \leq q_0 \leq 1$), qui définit une balance *diversification/intensification*. Une fourmi $k$ sur une ville $i$ choisira une ville $j$ par la règle :

$$j = \begin{cases} argmax_{u \in J_i^k} \left[ (\tau_{iu}(t)) \cdot (\eta_{iJ})^\beta \right] & \text{si } q \leq q_0 \\ J & \text{si } q > q_0 \end{cases}$$

où $q$ est une variable aléatoire uniformément distribuée sur $[0,1]$ et $J \in J_i^k$ une ville sélectionnée aléatoirement selon la probabilité :

$$p_{iJ}^k(t) = \frac{(\tau_{iJ}(t)) \cdot (\eta_{iJ})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t)) \cdot (\eta_{il})^\beta} \tag{4.4}$$

En fonction du paramètre $q_0$, il y a donc deux comportements possibles : si $q > q_0$ le choix se fait de la même façon que pour l'algorithme $AS$, et le système

[2] un algorithme d'apprentissage par renforcement
[3] "Système de Colonie de Fourmis"

tend à effectuer une *diversification* ; si $q \leq q_0$, le système tend au contraire vers une *intensification*. En effet, pour $q \leq q_0$, l'algorithme exploite plus l'information récoltée par le système, il ne peut pas choisir de trajet non exploré.

2. La gestion des pistes est séparée en deux niveaux : une mise à jour locale et une mise à jour globale. Chaque fourmi dépose une piste lors de la mise à jour locale :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \tau_0$$

où $\tau_0$ est la valeur initiale de la piste. À chaque passage, les arêtes visitées voient leur quantité de phéromone diminuer, ce qui favorise la diversification par la prise en compte des trajets non explorés. À chaque itération, la mise à jour globale s'effectue comme ceci :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t)$$

où les arêtes $(i,j)$ appartiennent au meilleur tour $T^+$de longueur $L^+$ et où $\Delta\tau_{ij}(t) = \frac{1}{L^+}$. Ici, seule la meilleure piste est donc mise à jour, ce qui participe à une intensification par sélection de la meilleure solution.

3. Le système utilise une liste de candidats. Cette liste stocke pour chaque ville les $v$ plus proches voisines, classées par distances croissantes. Une fourmi ne prendra en compte une arête vers une ville en dehors de la liste que si celle-ci a déjà été explorée. Concrètement, si toutes les arêtes ont déjà été visitées dans la liste de candidats, le choix se fera en fonction de la règle 4.4, sinon c'est la plus proche des villes non visitées qui sera choisie.

#### 4.3.2.4 ACS & 3-opt

Cette variante est une hybridation entre le ACS et une recherche locale de type 3-opt [Dorigo *et al.* 97b]. Ici, la recherche locale est lancée pour améliorer les solutions trouvées par les fourmis (et donc les ramener à l'optimum local le plus proche).

#### 4.3.2.5 Max-Min Ant System

Cette variante (notée *MMAS*) est fondée sur l'algorithme AS et présente quelques différences notables [Stützle *et al.* 97, Stützle *et al.* 00] :

1. Seule la meilleure fourmi met à jour une piste de phéromone ;
2. Les valeurs des pistes sont bornées par $\tau_{min}$ et $\tau_{max}$ ;
3. Les pistes sont initialisées à la valeur maximum $\tau_{max}$.
4. La mise à jour des pistes se fait de façon proportionnelle, les pistes les plus fortes étant moins renforcées que les plus faibles ;
5. Une ré-initialisation des pistes peut être effectuée.

Les meilleurs résultats sont obtenus en mettant à jour la meilleure solution avec une *fréquence* de plus en plus forte au cours de l'exécution de l'algorithme.

### 4.3.3 Choix des paramètres

Pour l'algorithme AS, les auteurs préconisent que, bien que la valeur de $Q$ ait peu d'influence sur le résultat final, cette valeur soit du même ordre de grandeur qu'une estimation de la longueur du meilleur trajet trouvé. D'autre part, la ville de départ de chaque fourmi est typiquement choisie par un tirage aléatoire uniforme, aucune influence significative du placement de départ n'ayant pu être démontrée.

En ce qui concerne l'algorithme *ACS*, les auteurs conseillent d'utiliser $\tau_0 = (n \cdot L_{nn})^{-1}$, où $n$ est le nombre de villes et $L_{nn}$ la longueur d'un tour trouvé par la méthode du plus proche voisin.

Le nombre de fourmis $m$ est un paramètre important, puisqu'il participe à la rétroaction positive principale du système. Les auteurs suggèrent d'utiliser autant de fourmis que de villes (i.e. $m = n$) pour de bonnes performances sur le problème du voyageur de commerce. Il est possible de n'utiliser qu'une seule fourmi, mais l'effet d'amplification des longueurs différentes est alors perdu, de même que le parallélisme naturel de l'algorithme, ce qui peut s'avérer néfaste pour certains problèmes. En règle générale, les algorithmes de colonies de fourmis semblent assez peu sensibles à un réglage fin du nombre de fourmis.

## 4.4 Autres problèmes combinatoires

Les algorithmes de colonies de fourmis sont beaucoup étudiés depuis quelques années et il serait trop long de faire ici une liste exhaustive de toutes les applications et variantes qui ont été produites. Dans les deux principaux champs d'application (problèmes $NP$-difficiles et problèmes dynamiques), certains algorithmes ont cependant donné de très bons résultats. On peut notamment retenir des performances particulièrement intéressantes dans le cas de l'affectation quadratique [Stützle *et al.* 00], de problèmes de planification [Merkle *et al.* 00], de l'ordonnancement séquentiel [Gambardella *et al.* 00], du routage de véhicule [Gambardella *et al.* 99b], ou du routage sur réseau [Di Caro *et al.* 98] (voir aussi pour cette application la section 4.6.2 de ce chapitre). Il existe une littérature importante sur toutes sortes de problèmes : voyageur de commerce, coloriage de graphes, affectation de fréquence, affectation généralisée, sac à dos multi-dimensionnel, satisfaction de contraintes, etc.

## 4.5 Formalisation et propriétés d'un algorithme de colonie de fourmis

Une description élégante a été proposée [Dorigo *et al.* 03], qui s'applique aux problèmes (combinatoires) où une construction partielle de la solution est possible. Ce cas, bien que restrictif, permet de dégager les apports originaux de ces métaheuristiques (dénommées *ACO,* pour "Ant Colony Optimization", par les auteurs).

> Une métaheuristique de colonie de fourmis est un processus stochastique construisant une solution, en ajoutant des composants aux solutions partielles. Ce processus prend en compte (i) une heuristique sur l'instance

du problème (ii) des pistes de phéromone changeant dynamiquement pour refléter l'expérience acquise par les agents.

Une formalisation plus précise existe [Dorigo *et al.* 03]. Elle passe par une *représentation* du problème, un *comportement* de base des fourmis et une *organisation* générale de la métaheuristique. Plusieurs concepts sont également à mettre en valeur pour comprendre les principes de ces algorithmes, notamment la définition des pistes de phéromone en tant que *mémoire adaptative*, la nécessité d'un réglage *intensification/diversification* et enfin l'utilisation d'une *recherche locale*. Nous traitons ci-après ces différents sujets.

## 4.5.1 Formalisation

### 4.5.1.1 Représentation du problème

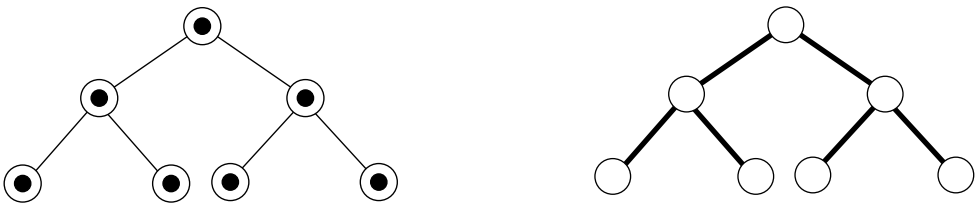
Le problème est représenté par un *jeu de solutions*, une *fonction objectif* assignant une valeur à chaque solution et un *jeu de contraintes*. L'objectif est de trouver l'optimum global satisfaisant les contraintes. Les différents états du problème sont caractérisés comme une séquence de composants. On peut noter que, dans certains cas, un coût peut être associé à des états autres que des solutions.

Dans cette représentation, les fourmis construisent des solutions en se déplaçant sur un graphe $G = (C, L)$, où les noeuds sont les composants de $C$ et où l'ensemble $L$ connecte les composants de $C$. Les contraintes du problème sont implémentées directement dans les règles de déplacement des fourmis (soit en empêchant les mouvements qui violent les contraintes, soit en pénalisant de telles solutions).

### 4.5.1.2 Comportement des fourmis

Les fourmis peuvent être caractérisées comme une procédure de construction stochastique *construisant* des solutions sur le graphe $G = (C, L)$. En général, les fourmis tentent d'élaborer des solutions faisables, mais si nécessaire, elles peuvent produire des solutions infaisables. Les composants et les connexions peuvent être associés à des pistes de phéromone $\tau$ (mettant en place une mémoire adaptative décrivant l'état du système) et à une valeur heuristique $\eta$ (représentant une information a priori sur le problème, ou venant d'une source autre que celle des fourmis ; c'est bien souvent le coût de l'état en cours). Les pistes de phéromone et la valeur de l'heuristique peuvent être associées soit aux composants, soit aux connexions (figure 4.6).

Chaque fourmi dispose d'une mémoire utilisée pour stocker le trajet effectué, d'un état initial et de conditions d'arrêt. Les fourmis se déplacent d'après une règle de décision probabiliste fonction des *pistes* de phéromone locales, de l'*état* de la fourmi et des *contraintes* du problème. Lors de l'ajout d'un composant à la solution en cours, les fourmis peuvent mettre à jour la piste associée au composant ou à la connexion correspondante. Une fois la solution construite, elles peuvent mettre à jour la piste de phéromone des composants ou des connexions utilisées. Enfin, une fourmi dispose au minimum de la capacité de construire une solution au problème.



(a) (b)

FIG. 4.6 – *Dans un algorithme de colonie de fourmis, les pistes de phéromone peuvent être associées aux composants (a) ou aux connexions (b) du graphe représentant le problème à résoudre.*

#### 4.5.1.3 Organisation de la métaheuristique

En plus des règles régissant le comportement des fourmis, un autre processus majeur a cours : l'*évaporation* des pistes de phéromone. En effet, à chaque itération, la valeur des pistes de phéromone est *diminuée*. Le but de cette diminution est d'éviter une convergence trop rapide et le piégeage de l'algorithme dans des minimums locaux, par une forme d'oubli favorisant l'exploration de nouvelles régions.

Selon les auteurs du formalisme *ACO*, il est possible d'implémenter d'autres processus nécessitant un contrôle *centralisé* (et donc ne pouvant être directement pris en charge par des fourmis), sous la forme de processus annexes. Ce n'est, à notre sens, que peu souhaitable ; en effet, on perd alors la caractéristique décentralisée du système. De plus, l'implémentation de processus *annexes* entre difficilement dans une formalisation rigoureuse, car on peut y voir n'importe quel processus.

### 4.5.2 Phéromones et mémoire

L'utilisation de la *stigmergie* est cruciale pour les algorithmes de colonies de fourmis. Le choix de la méthode d'implémentation des pistes de phéromone est donc important pour obtenir les meilleurs résultats. Ce choix est en grande partie lié aux possibilités de *représentation* de l'espace de recherche, chaque représentation pouvant apporter une façon différente d'implémenter les pistes. Par exemple, pour le problème du voyageur de commerce, une implémentation efficace consiste à utiliser une piste $\tau_{ij}$ entre deux villes $i$ et $j$ comme une représentation de l'*intérêt* de visiter la ville $j$ après la ville $i$. Une autre représentation possible, moins efficace en pratique, consiste à considérer $\tau_{ij}$ comme une représentation de l'intérêt de visiter $i$ en tant que $j$ème ville.

En effet, les pistes de phéromone décrivent à chaque pas l'état de la recherche de la solution par le système, les agents modifient la façon dont le problème va être *représenté* et perçu par les autres agents. Cette information est partagée par le biais des modifications de l'*environnement* des fourmis, par une forme de communication indirecte : la stigmergie. L'information est donc stockée un certain temps dans le

système, ce qui a amené certains auteurs à considérer ce processus comme une forme de *mémoire adaptative* [Taillard 98, Taillard *et al.* 98], où la dynamique de stockage et de partage de l'information va être cruciale pour le système.

### 4.5.3 Intensification/diversification

Le problème de l'emploi relatif de processus de *diversification* et d'*intensification* est un problème extrêmement courant dans la conception et l'utilisation de métaheuristiques. Par intensification, on entend l'*exploitation* de l'information rassemblée par le système à un moment donné. La diversification est au contraire l'*exploration* de régions de l'espace de recherche imparfaitement prises en compte. Bien souvent, il va s'agir de choisir où et quand "injecter de l'aléatoire" dans le système (*diversification*) et/ou améliorer une solution (*intensification*).

Dans les algorithmes de type ACO, comme dans la plupart des cas, il existe plusieurs façons de gérer l'emploi de ces deux facettes des métaheuristiques d'optimisation. La plus évidente passe par le réglage via les deux paramètres $\alpha$ et $\beta$, qui déterminent l'influence relative des pistes de phéromone et de l'information heuristique. Plus la valeur de $\alpha$ sera élevée, plus l'*intensification* sera importante, car plus les pistes auront une influence sur le choix des fourmis. À l'inverse, plus $\alpha$ sera faible, plus la *diversification* sera forte, car les fourmis éviteront les pistes. Le paramètre $\beta$ agit de façon similaire. On doit donc gérer à la fois les deux paramètres pour régler ces aspects.

On peut également introduire des modifications de la gestion des pistes de phéromone. Par exemple, l'emploi de stratégies *élitistes* (les meilleures solutions contribuent plus aux pistes, voir section 4.3.2.1 : l'algorithme AS avec élitisme) favorise l'intensification, alors qu'une ré-initialisation de l'ensemble des pistes favorisera l'exploration (section 4.3.2.5, algorithme MMAS).

Ce choix diversification/intensification peut s'effectuer de manière statique avant le lancement de l'algorithme, en utilisant une connaissance a priori du problème, ou de manière dynamique, en laissant le système décider du meilleur réglage. Deux approches sont possibles : un réglage par les paramètres ou l'introduction de nouveaux processus. Dans ces algorithmes fondés en grande partie sur l'utilisation de l'auto-organisation, ces deux approches peuvent être équivalentes, un changement de paramètre pouvant induire un comportement complètement différent du système, au niveau global.

### 4.5.4 Recherche locale et heuristiques

Les métaheuristiques de colonies de fourmis sont souvent plus efficaces quand elles sont *hybridées* avec des algorithmes de recherche locale. Ceux-ci optimisent les solutions trouvées par les fourmis avant que celles-ci ne soient utilisées pour la mise à jour des pistes de phéromone. Du point de vue de la recherche locale, utiliser des algorithmes de colonies de fourmis pour générer une solution initiale est un avantage indéniable. Ce qui différencie une métaheuristique de type ACO *intéressante* d'un algorithme réellement *efficace* est bien souvent l'hybridation avec une recherche locale.

Une autre possibilité pour améliorer les performances est d'injecter une information heuristique plus pertinente. Cet ajout a généralement un coût élevé en terme de calculs supplémentaires.

Il faut noter que ces deux approches sont similaires de par l'emploi qu'elles font des informations de coût pour améliorer une solution. La recherche locale le fait de façon plus directe que l'heuristique, cependant que cette dernière est peut-être plus naturelle pour utiliser des informations a priori sur le problème.

### 4.5.5 Parallélisme

La structure même des métaheuristiques de colonies de fourmis comporte un parallélisme *intrinsèque*. D'une manière générale, les solutions de bonne qualité émergent du résultat des *interactions* indirectes ayant cours dans le système, pas d'un codage explicite d'échanges. En effet, chaque fourmi ne prend en compte que des informations locales de son environnement (les pistes de phéromones) ; il est donc très facile de paralléliser un tel algorithme. Il est intéressant de noter que les différents processus en cours dans la métaheuristique (i.e. le comportement des fourmis, l'évaporation et les processus annexes) peuvent également être implémentés de manière indépendante, l'utilisateur étant libre de décider de la manière dont ils vont interagir.

### 4.5.6 Convergence

Les métaheuristiques peuvent être vues comme des modifications d'un algorithme de base : une recherche aléatoire. Cet algorithme possède l'intéressante propriété de *garantir* que la solution optimale sera trouvée tôt ou tard, on parle alors de convergence. Cependant, puisque cet algorithme de base est *biaisé*, la garantie de convergence n'existe plus.

Si, dans certains cas, on peut facilement être certain de la convergence d'un algorithme de colonies de fourmis (*MMAS* par exemple, voir section 4.3.2.5), le problème reste entier en ce qui concerne la convergence d'un algorithme *ACO* quelconque. Cependant, il existe une variante dont la convergence a été prouvée [Gutjahr 00, Gutjahr 02] : le "Graph-Based Ant System" (*GBAS*). La différence entre *GBAS* et l'algorithme AS se situe au niveau de la mise à jour des pistes de phéromone, qui n'est permise que si une meilleure solution est trouvée. Pour certaines valeurs de paramètres, et étant donné $\epsilon > 0$ une faible valeur, l'algorithme trouvera la solution optimale avec une probabilité $P_t \geq 1 - \epsilon$, après un temps $t \geq t_0$ (où $t_0$ est fonction de $\epsilon$).

## 4.6 Perspectives

Devant le succès rencontré par les algorithmes de colonies de fourmis, de nombreuses pistes autres que celle de l'optimisation combinatoire commencent à être explorées : par exemple, l'utilisation de ces algorithmes dans des problèmes *continus* et/ou *dynamiques*, ou encore la mise en relation de ce type d'algorithmes dans un cadre d'*intelligence en essaim* et avec d'autres métaheuristiques.

### 4.6.1 Optimisation continue

#### 4.6.1.1 Problèmes d'adaptation

Les métaheuristiques sont bien souvent élaborées pour des problèmes combinatoires, mais il existe une classe de problèmes souvent rencontrée en ingénierie, où la fonction objectif est *continue* et pour lesquels les métaheuristiques peuvent être d'un grand secours (fonction non dérivable, multiples minimums locaux, grand nombre de variables, non-convexité, etc ; voir section 6.2). Plusieurs tentatives pour adapter les métaheuristiques de colonies de fourmis au domaine continu sont apparues.

En plus des problèmes classiques d'adaptation d'une métaheuristique, les algorithmes de colonies de fourmis posent un certain nombre de problèmes spécifiques. Ainsi, le principal problème vient si l'on se place dans le formalisme ACO avec une construction de la solution composant par composant. En effet, un problème continu peut — selon la perspective choisie — présenter une infinité de composants, le problème de la construction est difficilement soluble dans ce cas. La plupart des algorithmes s'inspirent donc des caractéristiques d'auto-organisation et de mémoire externe des colonies de fourmis, laissant de côté la construction itérative de la solution.

Nous avons recensé quatre algorithmes de colonies de fourmis pour l'optimisation continue : CACO, un algorithme hybride non baptisé, CIAC et API.

#### 4.6.1.2 L'algorithme CACO

Le premier de ces algorithmes, tout naturellement nommé *CACO* ("Continuous Ant Colony Algorithm") [Bilchev *et al.* 95, Wodrich *et al.* 97, Mathur *et al.* 00], utilise deux approches : un algorithme de type *évolutionnaire* sélectionne et croise des régions d'intérêt, que des *fourmis* explorent et évaluent. Une fourmi sélectionne une région avec une probabilité proportionnelle à la concentration en phéromone de cette région, de la même manière que — dans le "Ant System" —, une fourmi sélectionnerait une piste allant d'une ville à une autre :

$$p_i(t) = \frac{\tau_i^{\alpha}(t) \cdot \eta_i^{\beta}(t)}{\sum_{j=1}^{N} \tau_j^{\alpha}(t) \cdot \eta_j^{\beta}(t)}$$

où $N$ est le nombre de régions et $\eta_i^{\beta}(t)$ est utilisé pour inclure une heuristique spécifique au problème. Les fourmis partent alors du centre de la région et se déplacent selon une direction choisie aléatoirement, tant qu'une amélioration de la fonction objectif est trouvée. Le pas de déplacement utilisé par la fourmi entre chaque évaluation est donné par :

$$\delta r(t, R) = R \cdot \left(1 - u^{\left(1 - \frac{t}{T}\right)^c}\right)$$

où $R$ est le diamètre de la région explorée, $u \in [0, 1]$ un nombre aléatoire, $T$ le nombre total d'itérations de l'algorithme et $c$ un paramètre de refroidissement. Si la fourmi a trouvé une meilleure solution, la région est déplacée de façon à ce que son centre coïncide avec cette solution, et la fourmi augmente la quantité de phéromone de la

région proportionnellement à l'amélioration trouvée. L'évaporation des "pistes" se fait classiquement en fonction d'un coefficient $\rho$.

Des modifications ont été apportées par Wodrich et al. [Wodrich *et al.* 97] pour améliorer les performances de l'algorithme original. Ainsi, en plus des fourmis "locales" de CACO, des fourmis "globales" vont explorer l'espace de recherche (figure 4.7) pour éventuellement remplacer les régions peu intéressantes par de nouvelles régions non explorées. Les régions sont également affectées d'un âge, qui augmente si aucune amélioration n'est découverte. De plus, le paramètre $t$ dans le pas de recherche des fourmis $\delta r(t, R)$ est défini par l'âge de la région explorée.

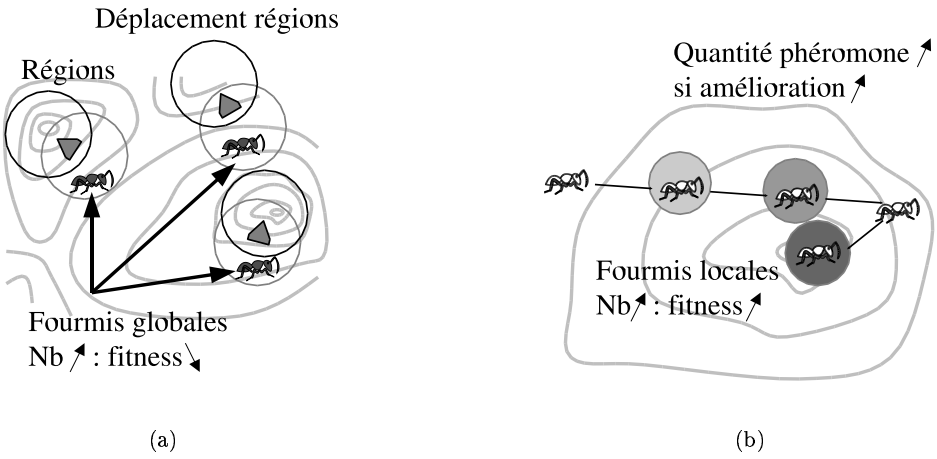



FIG. 4.7 – *L'algorithme CACO : les fourmis globales (a) participent au déplacement des régions que les fourmis locales (b) évaluent.*

Une refonte de l'algorithme [Mathur *et al.* 00] a été opérée en vue de relier plus finement CACO avec le paradigme des colonies de fourmis et d'abandonner la liaison avec l'algorithme évolutionnaire. On parle ainsi par exemple de diffusion pour définir la création de nouvelles régions.

Cet algorithme a été comparé à certains algorithmes classiques et a fait montre de performances moyennes dans ses premières versions et meilleures dans ses dernières versions.

#### 4.6.1.3 Une méthode hybride

Une approche semblable — utilisant à la fois une approche de colonies de fourmis et d'algorithme évolutionnaire — a été proposée par Ling et al. [Ling *et al.* 02], mais peu de résultats sont disponibles au moment où nous écrivons ce livre. L'idée principale de cette méthode est de considérer les écarts entre deux individus sur chaque dimension comme autant de parties d'un chemin où les phéromones sont déposées, l'évolution des individus étant prise en charge par des opérateurs de mutation et de croisement.

D'un certain point de vue, cette méthode tente donc de reproduire le mécanisme de construction de la solution composant par composant.

La méthode procède précisément comme décrit dans l'algorithme 4.2. Chaque fourmi $x_i$ de la population contenant $m$ individus est considérée comme un vecteur à $n$ dimensions. Chaque élément $x_{i,e}$ de ce vecteur peut donc être considéré comme un candidat à l'élément $x^*_{i,e}$ de la solution optimale. L'idée est d'utiliser le chemin entre les éléments $x_{i,e}$ et $x_{j,e}$ — noté $(i,j)$ — pour déposer une piste de phéromone dont la concentration est notée $\tau_{ij}(t)$ au pas de temps $t$.

Les auteurs ont proposé une version "adaptative" où les probabilités de mutation et de croisement sont variables. Malheureusement cet algorithme n'a pas encore été complètement testé, ses performances sont donc sujettes à caution.

**Algorithme 4.2** Un algorithme de colonies de fourmis hybride pour le cas continu.

1. À chaque itération, chaque fourmi sélectionne une valeur initiale dans le groupe de valeurs candidates avec la probabilité :

$$p^k_{ij}(t) = \frac{\tau_{ij}(t)}{\sum \tau_{ir}(t)}$$

2. Utiliser des opérateurs de mutation et de croisement sur les $m$ valeurs afin d'obtenir $m$ nouvelles valeurs ;
3. Ajouter ces nouvelles valeurs au groupe de valeurs candidates pour le composant $x_{i,e}$ ;
4. Les utiliser pour former $m$ solutions de la nouvelle génération ;
5. Calculer la "fitness" de ces solutions ;
6. Quand $m$ fourmis ont parcouru toutes les arêtes, mettre à jour les pistes de phéromone des valeurs candidates de chaque composant par :

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ir}(t) + \sum \tau^k_{ij}$$

7. Si la $k^{me}$ fourmi choisit la $j^{me}$ valeur candidate du groupe de composants, alors $\Delta\tau^k_{ij}(t+1) = Wf_k$, sinon $\Delta\tau^k_{ij} = 0$. Avec $W$ une constante et $f_k$ la "fitness" de la solution trouvée par la $k^{me}$ fourmi ;
8. Effacer les $m$ valeurs ayant les plus basses intensités de phéromone dans chaque groupe de candidats.

#### 4.6.1.4 L'algorithme CIAC

Un autre algorithme, se focalisant sur les principes de *communication* des colonies de fourmis a été élaboré par deux des coauteurs de cet ouvrage. Il propose d'ajouter aux processus stigmergiques des échanges *directs* d'informations [Dréo *et al.* 02], en s'inspirant pour cela de l'approche "hétérarchique" décrite précédemment dans la

section 4.2.1.4. Ainsi, une formalisation des échanges d'informations est proposée autour de la notion de canaux de communication. En effet, il existe plusieurs façons possibles de faire passer de l'information entre deux groupes d'individus, par exemple soit par dépôts de pistes de phéromone soit par échanges directs. On peut définir de la sorte différents *canaux de communication* représentant l'ensemble des caractéristiques du transport de l'information. Du point de vue des métaheuristiques, il y a trois caractéristiques principales (voir figure 4.8) :

**Portée :** le nombre d'individus mis en cause dans l'échange d'information. L'information peut par exemple être émise par un individu et être perçue par plusieurs autres, et inversement.

**Mémoire :** la persistance de l'information dans le système. L'information peut rester un certain temps dans le système ou n'être que transitoire.

**Intégrité :** les modifications engendrées par l'utilisation du canal de communication. L'information peut varier dans le temps ou être biaisée lors de sa transmission.

De plus, l'information passant par un canal de communication peut bien sûr être n'importe quelle information d'intérêt, comme par exemple la valeur et/ou la position d'un point de l'espace de recherche.

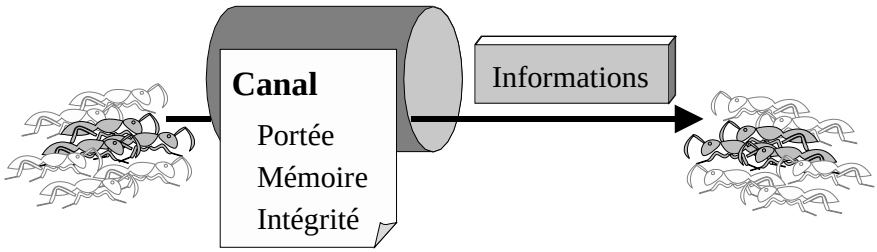



FIG. 4.8 – *Un canal de communication structure les caractéristiques de la transmission de l'information : portée, mémoire et intégrité.*

L'algorithme *CIAC* (acronyme pour "Continuous Interacting Ant Colony") utilise deux canaux de communication :

1. Le canal stigmergique fait appel à des spots de phéromone, déposés sur l'espace de recherche, qui vont être plus ou moins attractifs pour les fourmis artificielles, selon leurs concentrations et leurs distances. Les caractéristiques du canal stigmergique sont donc les suivantes : la portée est à son maximum, toutes les fourmis peuvent potentiellement prendre en compte l'information, il y a utilisation de mémoire puisque les spots persistent sur l'espace de recherche, enfin l'information évolue avec le temps puisque les spots s'évaporent. L'information portée par un spot contient implicitement la position d'un point et explicitement la valeur de l'amélioration trouvée par la fourmi ayant déposé le spot.
2. Le canal direct est implémenté sous la forme d'échange de messages entre deux individus. Une fourmi artificielle possède une pile de messages reçus et peut en

envoyer à une autre fourmi. La portée de ce canal est de un puisque seule une fourmi reçoit le message, la mémoire est implémentée dans la pile de messages que la fourmi mémorise, enfin l'information (ici un couple position/valeur d'un point) n'est pas altérée au cours du temps.

L'algorithme a montré certaines caractéristiques intéressantes, tirant parti des propriétés auto-organisées des algorithmes de colonies de fourmis, notamment une certaine capacité à osciller entre un processus d'intensification et un processus de diversification quand les deux canaux de communication (stigmergique et direct) sont utilisés en synergie. La figure 4.9 illustre ce comportement d'oscillations : en ordonnée est reporté l'écart-type de la distribution des valeurs de la fonction objectif, un grand écart-type correspond à une grande dispersion des fourmis sur l'axe des valeurs (*diversification*) alors qu'une faible valeur correspond au contraire à un regroupement (*intensification*). Il faut noter que ce comportement n'est pas observé lors de l'utilisation d'un seul canal, il y a donc synergie entre les deux canaux.

Cependant, les résultats ne sont comparables qu'à ceux produits par les autres algorithmes de colonies de fourmis pour le domaine continu, donc en deçà des meilleurs résultats procurés par d'autres métaheuristiques adaptées au cas continu.

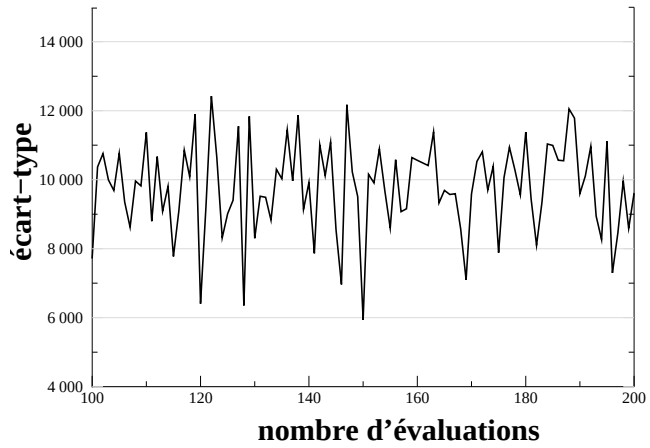



FIG. 4.9 – *Oscillations observées lors de l'utilisation conjointe de deux canaux de communication dans l'algorithme CIAC.*

Cette approche a donné lieu à une hybridation avec l'algorithme de recherche locale de Nelder-Mead [Dréo *et al.* 03]. Cette modification de l'algorithme *CIAC* original, appelée *HCIAC*, utilise donc deux canaux de communication, ajoute une recherche locale et des processus décisionnels stochastiques. Ce dernier point est implémenté par l'utilisation de fonctions de type stimulus/réponse qui permettent de définir un seuil de choix pour une action. Concrètement on utilise une fonction sigmoïde $p(x) = \frac{1}{1+e^{\delta\omega-\omega x}}$ pour tester un choix fonction d'un état $x$ d'une fourmi, d'un

seuil $\delta$ déterminant la position du point d'inflexion et d'une puissance $\omega$ caractérisant l'inflexion de la sigmoïde. On tire un nombre aléatoire $r$ dans une distribution uniforme, on a donc deux choix possibles : $r < p(x)$ ou $r > p(x)$. Pour $\delta = 0.5$ et $\omega = +\infty$ on obtient par exemple un simple choix binaire. L'utilisation de ce type de fonction permet de s'affranchir d'un paramétrage délicat, par exemple en distribuant les seuils selon une loi normale sur toute la population. De même, on peut mettre en place par ce biais un système très simple d'apprentissage en faisant varier les seuils.

L'algorithme HCIAC est décrit sur la figure 4.10. L'hybridation a — comme souvent avec les algorithmes de colonies de fourmis — permis d'atteindre des résultats comparables à ceux procurés par d'autres métaheuristiques concurrentes pour les problèmes continus.

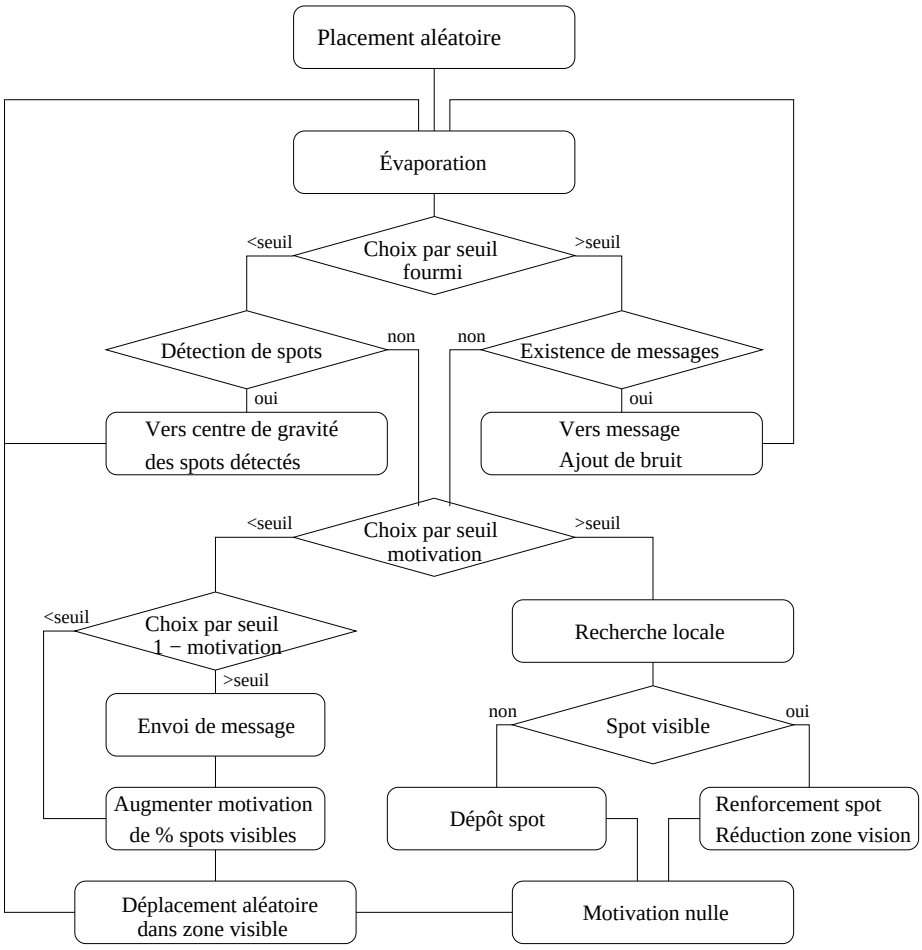



FIG. 4.10 – *L'algorithme HCIAC.*

#### 4.6.1.5 L'algorithme API

Dans tous ces algorithmes adaptés à des problèmes continus, le terme "colonies de fourmis" s'entend de par l'utilisation de la stigmergie comme processus d'échange d'information.

Il existe cependant un algorithme adaptable au cas continu [Monmarché *et al.* 00] qui s'inspire du comportement de fourmis primitives (ce qui ne veut pas dire *inadaptées*) de l'espèce *Pachycondyla apicalis*, et qui ne fait *pas* usage de la communication indirecte par pistes de phéromone : l'algorithme *API*.

Dans cette méthode, on commence par positionner un nid aléatoirement sur l'espace de recherche, puis des fourmis sont envoyées aléatoirement dans un périmètre donné. Ces fourmis vont alors explorer localement leur "site de chasse" en évaluant plusieurs points dans un périmètre donné (voir figure 4.11). La fourmi mémorise le meilleur point trouvé. Si lors de l'exploration de son site de chasse elle trouve un meilleur point, alors elle reviendra sur ce site, sinon après un certain nombre d'explorations, elle choisira un autre site. Une fois les explorations des sites de chasse terminées, des fourmis tirées au hasard comparent deux à deux (comme peuvent le faire les fourmis réelles durant le comportement de "tandem-running[4]") leurs meilleurs résultats, puis mémorisent le meilleur des deux sites de chasse. Le nid est finalement réinitialisé sur le meilleur point trouvé après un temps donné, la mémoire des sites des fourmis est remise à zéro et l'algorithme effectue une nouvelle itération.

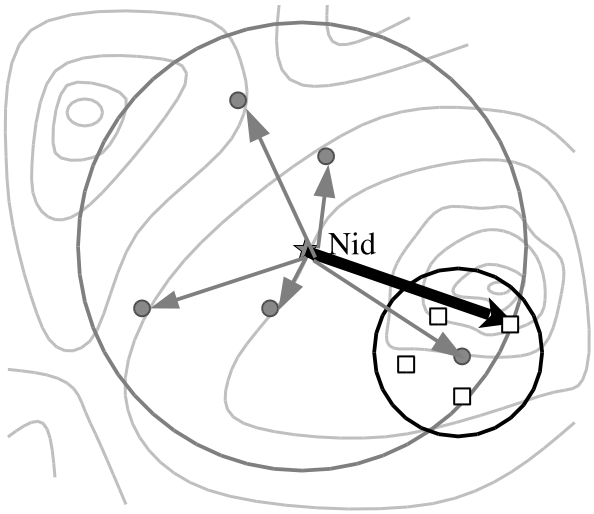



FIG. 4.11 – *L'algorithme API : une méthode à démarrage multiple inspirée par une espèce de fourmi primitive. Les fourmis (cercles pleins) explorent des sites de chasse (petits carrés) dans un périmètre (grand cercle) autour du nid. Le nid est déplacé sur le meilleur point au moment de la ré-initialisation (flèche en trait gras).*

[4]qu'on pourrait traduire par "course en couple"

#### 4.6.1.6 Conclusion dans le domaine continu

On a pu voir que deux types d'algorithmes sur quatre étaient en fait plus ou moins hybridés avec un algorithme de type évolutionnaire, et qu'un troisième n'utilisait pas la métaphore "classique" des colonies de fourmis. D'une manière générale, la recherche en est encore à ses débuts et les algorithmes produits n'ont pas atteint leur pleine maturité, et ne sont donc pas encore vraiment compétitifs par rapport à d'autres classes de métaheuristiques plus élaborées sur les problèmes continus.

### 4.6.2 Problèmes dynamiques

Un problème est dit dynamique s'il *varie* dans le temps, i.e. si la solution optimale ne possède pas les mêmes caractéristiques durant le temps de l'optimisation. Ces problèmes soulèvent des difficultés spécifiques, du fait qu'il faut approcher au mieux la meilleure solution à *chaque* pas de temps.

La première application des algorithmes de colonies de fourmis sur des problèmes dynamiques concernait l'optimisation du routage sur des réseaux de type téléphonique [Schoonderwoerd *et al.* 96], mais l'algorithme proposé n'ayant pas fait l'objet d'études et de comparaisons approfondies, il est difficile d'en tirer des enseignements. Une autre application a également été proposée par White et al. [White *et al.* 98, Bieszczad *et al.* 99] sur des problèmes similaires. Une application a été décrite pour des problèmes de routage sur des réseaux de type Internet (voir figure 4.12) : l'algorithme AntNet [Di Caro *et al.* 97]. Cette métaheuristique a fait l'objet de plusieurs études (voir notamment [Di Caro *et al.* 98]) et semble avoir prouvé son efficacité sur plusieurs problèmes tests.

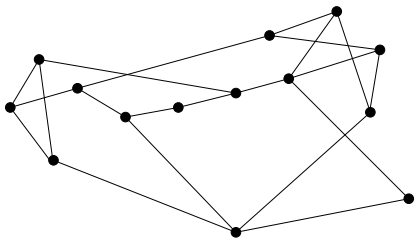


FIG. 4.12 – *Exemple de réseau utilisé pour tester l'algorithme AntNet :* NFSNET *(chaque arête représente une liaison orientée).*

Chacun de ces algorithmes utilise, pour mettre à jour des tables de routage probabilistes, des fourmis pour explorer le réseau de façon à ce que l'information pertinente soit la fréquence de passage des fourmis sur chaque noeud. D'une manière générale, l'aspect distribué et flexible des algorithmes de colonies de fourmis semble très bien adapté à des problèmes dynamiques.

### 4.6.3 Métaheuristiques et éthologie

Les métaheuristiques sont la plupart du temps issues de *métaphores* provenant de la nature, et notamment de la biologie. Les algorithmes de colonies de fourmis sont inspirés par le comportement d'insectes sociaux, mais ce ne sont pas les seuls algorithmes à être issus de l'étude du comportement animal (*éthologie*). En effet, l'optimisation par essaims particulaires ("Particle Swarm Optimization" [Eberhart *et al.* 01], voir section 5.6) est issue d'une analogie avec les comportements collectifs de déplacements d'animaux, tels qu'observés dans les bancs de poissons ou les vols d'oiseaux ; il existe également des algorithmes inspirés des comportements des abeilles [Choo 00, Panta 02]. On trouve encore des algorithmes s'inspirant de certains aspects du comportement des insectes sociaux, bien que ne faisant pas usage des caractéristiques *classiques* des algorithmes de colonies de fourmis (voir par exemple [De Wolf *et al.* 02, Nouyan 02] ainsi que la section 5.12 de ce livre).

Tout porte à croire que l'éthologie peut être une source d'inspiration intéressante pour la conception de nouvelles métaheuristiques.

### 4.6.4 Liens avec d'autres métaheuristiques

Les métaheuristiques forment une classe d'algorithmes foisonnante, où beaucoup de concepts se retrouvent d'une catégorie à l'autre. De plus, les nombreuses variantes d'un type d'algorithmes font que les frontières entre différentes métaheuristiques sont floues.

Un exemple de recoupement entre deux métaheuristiques est donné par le terme d'"intelligence en essaim", qui est utilisé pour décrire le mode de fonctionnement des algorithmes de colonies de fourmis [Bonabeau *et al.* 99], mais aussi d'autres algorithmes comme les "essaims particulaires" [Eberhart *et al.* 01] (voir section 5.6 pour une description détaillée). D'une manière générale, ce terme se réfère à tout système (généralement artificiel toutefois) possédant des propriétés d'auto-organisation — similaires à celles décrites dans la section 4.2.1.1 — capables de résoudre un problème donné par la seule force des interactions au niveau individuel.

Une tentative plus large de présentation unifiée est apparue : le cadre de la "programmation à mémoire adaptative" [Taillard *et al.* 98] (voir section 7.5), incluant notamment les colonies de fourmis avec la recherche tabou et les algorithmes évolutionnaires. Ce cadre insiste sur l'utilisation d'une forme de *mémoire* dans ces algorithmes, ainsi que sur l'utilisation de phases d'*intensification* et de *diversification* (voir section 4.5.3 pour cet aspect des colonies de fourmis artificielles).

On peut donc rapprocher plusieurs métaheuristiques des algorithmes de colonies de fourmis, et inversement. Le premier rapprochement venant à l'esprit provient du fait que les algorithmes de colonies de fourmis ne sont très souvent efficaces qu'avec un processus de recherche locale (voir section 4.5.4). Ainsi, d'un certain point de vue, un algorithme de colonie de fourmis ressemble fortement à l'algorithme *GRASP* [Feo *et al.* 95, Resende 00] ("Greedy Randomized Adaptive Search Procedures[5]", voir section 5.8) avec une phase de *construction* particulière.

[5] traduisible par "procédures de recherche adaptative aléatoire gloutonne"

De la même manière, la méthode "Cross-Entropy[6]" [Rubinstein 97, Rubinstein 01] (voir section 5.9) dispose de deux phases : d'abord la génération d'un jeu de données *aléatoire*, puis le changement des paramètres *générant* ce jeu de données pour obtenir un meilleur jeu pour la prochaine itération. Là encore, on peut considérer que cette méthode est proche de celle des colonies de fourmis [Rubinstein 01]. Certains travaux visent même à utiliser conjointement ces deux méthodes [Wittner *et al.* 02].

On peut également effectuer un rapprochement avec l'optimisation par essaim particulaire [Kennedy *et al.* 95, Eberhart *et al.* 01] (décrite section 5.6), qui elle aussi utilise les capacités de systèmes fortement distribués, ici de larges groupes de particules parcourant l'espace de recherche, avec une dynamique de déplacement qui tend à les faire se rapprocher les unes des autres.

Un autre rapprochement très intéressant peut se faire avec les algorithmes à estimation de distribution ("Estimation of Distribution Algorithms", *EDA*, [Larranaga *et al.* 02], décrits section 5.7). En effet, ces algorithmes — dérivés d'algorithmes évolutionnaires à l'origine — reposent sur le fait qu'à chaque itération, les individus sur l'espace de recherche sont tirés aléatoirement selon une *distribution* construite à partir des états des individus précédents. Schématiquement, meilleur est un individu, plus grande est la probabilité de création d'autres individus aux alentours. La ressemblance avec les algorithmes de type $ACO$ est frappante [Monmarché *et al.* 99].

On peut donc également faire un parallèle entre algorithmes évolutionnaires (voir chapitre 3) et colonies de fourmis, de par l'exploitation de populations d'"agents", les processus mémoriels ou encore probabilistes qu'ils utilisent. On peut également mettre l'accent sur l'idée, défendue par certains biologistes, selon laquelle les phénomènes d'auto-organisation auraient un rôle à jouer dans les processus évolutifs [Camazine *et al.* 00]... processus dont s'inspirent les algorithmes évolutionnaires !

Une nouvelle approche — moins liée aux métaheuristiques — consiste à considérer une classe particulière d'algorithmes de colonies de fourmis (classe appelée "Ant Programming[7]") et à la replacer entre les théories du contrôle optimal et de l'apprentissage par renforcement [Birattari *et al.* 02].

On voit bien que de nombreuses interactions et recouvrements existent, les relations entre algorithmes évolutionnaires, algorithmes à évolution de distribution et colonies de fourmis sont en ce sens suffisamment parlantes, chacune révélant finalement les caractéristiques des autres. Il est donc difficile d'étudier la métaheuristique de colonies de fourmis comme un ensemble homogène, et délicat de la considérer comme une classe bien séparée des autres. Cependant, la puissance de la métaphore et le regroupement d'un ensemble de caractéristiques maintenant relativement bien connues (voir section 4.5) permettent d'en clarifier la définition.

[6] Entropie croisée

[7] Programmation par fourmis

## 4.7 Conclusion

La métaheuristique s'inspirant des colonies de fourmis commence à être bien décrite et formalisée. L'ensemble des propriétés la décrivant est connu : construction probabiliste d'une solution (par ajout de composants dans le formalisme $ACO$), heuristique sur l'instance du problème, utilisation d'une forme de mémoire indirecte et d'une structure comparables à celles d'un système auto-organisé. Les idées directrices sous-tendant les algorithmes de colonies de fourmis sont puissantes ; on pourrait décrire cette métaheuristique comme un système *distribué* où les *interactions* entre composants de base, par le biais de processus *stigmergiques*, permettent de faire émerger un comportement global cohérent rendant le système capable de résoudre des problèmes d'optimisation difficiles.

Les colonies de fourmis ont été appliquées avec succès à de nombreux problèmes combinatoires et commencent à être adaptées à des problèmes continus. L'importance du choix d'une recherche locale est à mettre en valeur pour produire des algorithmes compétitifs avec d'autres métaheuristiques plus anciennes et souvent plus spécialisées. Les problèmes dynamiques semblent être la cible de choix pour ces algorithmes fondés sur une structure auto-organisée, spécialement quand seule une information locale est disponible.

## 4.8 Bibliographie commentée

[Hölldobler *et al.* 90] : Ce livre recense une somme impressionnante de connaissances sur la biologie des fourmis. Une bible en la matière, qui a reçu le prix Pullitzer.

[Camazine *et al.* 00] : On trouvera ici une description très complète des principes de l'auto-organisation dans les systèmes biologiques, accompagnée de nombreux exemples. Des descriptions de modèles permettent de comprendre les fondements théoriques de l'auto-organisation.

[Bonabeau *et al.* 99] : Cet ouvrage traite des algorithmes de colonies de fourmis comme des systèmes faisant preuve d'intelligence en essaim. Le livre s'articule autour de concepts biologiques et algorithmiques, notamment autour des métaheuristiques de colonies de fourmis. Une référence sur les algorithmes de type ACO.

[Dorigo *et al.* 03] : Un chapitre spécifiquement dédié aux algorithmes des colonies de fourmis dans un livre généraliste sur les métaheuristiques. Moins riche que le précédent, mais plus récent.

[Dorigo *et al.* 02] : Les actes du dernier congrès *ANTS* sur les "algorithmes de fourmis", une vision rapide des recherches les plus récentes dans le domaine. Le congrès se tient tous les deux ans depuis l'année 1998.