Optimisation par colonies de fourmis

COSTANZO Andrea LUONG Thé Van MARILL Guillaume

19 mai 2006

Table des matières

Chapitre 1

Introduction

1.1 Pourquoi les fourmis ?

Nous présenterons dans notre travail d'étude des nouvelles méta-heuristiques permettant de résoudre des problèmes tels que le voyageur de commerce, en s'inspirant du comportement social des fourmis.

Le comportement des fourmis est un comportement collectif. Chaque fourmi a pour priorité le bien être de la communauté.

Chaque individu de la colonie est à priori indépendant et n'est pas supervisé d'une manière ou d'une autre. Ce concept est appellé Hétérarchie (s'opposant à la Hiérarchie)[Dréo 04], chaque individu est aidé par la communauté dans son évolution et en retour il aide au bon fonctionnement de celle-ci.

La colonie est donc auto-controlé par le biais de mécanismes relativement simples à étudier.

1.2 Relation avec l'informatique

En observant une colonie de fourmis à la recherche de nourriture dans les environs du nid, on s'aperçoit qu'elle résoud des problèmes tels que celui de la recherche du plus court chemin.

Les fourmis résolvent des problèmes complexes par des mécanismes assez simples a modéliser.

Il est ainsi assez simple de simuler leur comportement par des algorithmes.

1.3 Comportement de la fourmi

Les pistes de phéromones

En marchant du nid à la source de nourriture et vice-versa (ce qui dans un premier temps se fait essentiellement de façon aléatoire), les fourmis déposent au passage sur le sol une substance odorante appelée **phéromones**. Cette substance permet ainsi donc de créer une piste chimique, sur laquelle les fourmis s'y retrouvent. En effet, d'autres fourmis peuvent détecter les phéromones grâce à des capteurs sur leurs antennes.

Les phéromones ont un rôle de marqueur de chemin : quand les fourmis choisissent leur chemin, elles ont tendance à choisir la piste qui porte la plus forte concentration de phéromones. Cela leur permet de retrouver le chemin vers leur nid lors du retour. D'autre part, les odeurs peuvent etre utilisées par les autres fourmis pour retrouver les sources de nourritures trouvées par leurs congénères.

Ce comportement permet de trouver le chemin le plus court vers la nourriture lorsque les pistes de phéromones sont utilisées par la colonie entière. Autrement dit, lorsque plusieurs chemins marqués sont à la disposition d'une fourmi, cette dernière peut connaitre le chemin le plus court vers sa destination. Cette constatation essentielle est la base de toutes les méthodes que l'on va développer plus loin.

On va d'abord étudier le comportement naturel de ces individus afin d'en prouver la validité, avant de l'extraire pour le simuler informatiquement.

Chapitre 2

Optimisation par colonie de fourmis

2.1 Historique

Dans les sections qui viendront plus tard, nous présenterons la méta-heuristique ACO, pour le "*Ant Colony Optimization*". Toutes ces idées abstraites sont inspirées des travaux de Deneubourg sur les fourmis.

Cette méta-heuristique est relativement récente. Elle a été introduite en 1991 par Colorni, Dorigo et Maniezzo pour résoudre le problème du Voyageur de commerce. Elle s'est popularisée, puis a été l'objet d'améliorations dès 1995 et a été appliquée avec succès à d'autres problèmes d'optimisation combinatoire dès 1994.

Nous allons tout d'abord exposer les différences et les points communs entre les fourmis virtuelles et les fourmis réelles [Dorigo 03], avant d'exposer en termes plus abstraits la méta-heuristique proprement dite. Ceci expliquera comment les fourmis virtuelles peuvent être exploitées pour résoudre un problème d'optimisation combinatoire.

2.2 Similarités et différences avec les fourmis réelles [Dréo 04]

Les fourmis virtuelles ont une double nature. D'une part, elles modélisent les comportements abstraits de fourmis réelles, et d'autre part, elles peuvent être enrichies par des capacités que ne possèdent pas les fourmis réelles, afin de les rendre plus efficaces que ces dernières. Nous allons maintenant synthétiser ces ressemblances et différences.

2.2.1 Points communs

Colonie d'individus coopérants. Comme pour les fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisés, qui se rassemblent ensemble pour trouver une "bonne" solution au problème considéré. Chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.

Pistes de phéromones. Ces entités communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.

Évaporation des phéromones. La méta-heuristique ACO comprend aussi la possibilité d'évaporation des phéromones. Ce mécanisme permet d'oublier lentement ce qui s'est passé avant. C'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte

par ses anciennes décisions.

Recherche du plus petit chemin. Les fourmis réelles et virtuelles partagent un but commun : recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).

Deplacement locaux Les vraies fourmis ne sautent pas des cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre sites adjacents du terrain.

Choix aléatoire lors des transitions. Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi du contexte de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle).

2.2.2 Différences

Les fourmis virtuelles possèdent certaines caractéristiques que ne possèdent pas les fourmis réelles :

Elle vivent dans un monde non-continu. Leurs déplacements consistent en des transitions d'état.

Mémoire (état interne) de la fourmi. Les fourmis réelles ont une mémoire très limitée. Tandis que nos fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.

Nature des phéromones déposées. Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent, là où les fourmis virtuelles modifient des informations dans les variables d'états associées au problème. Ainsi, l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération.

Qualité de la solution. Les fourmis virtuelles déposent une quantité de phéromone proportionnelle à la qualité de la solution qu'elles ont découvert.

Retard dans le dépôt de phéromone. Les fourmis virtuelles peuvent mettre à jour les pistes de phéromones de façon non immédiate : souvent elles attendent d'avoir terminé la construction de leur solution. Ce choix dépend du problème considéré bien évidemment.

Capacités supplémentaires. Les fourmis virtuelles peuvent être pourvues de capacités artificielles afin d'améliorer les performances du système. Ces possibilités sont liées au problème et peuvent être :

1. l'anticipation : la fourmi étudie les états suivants pour faire son choix et non seulement l'état local.
2. le retour en arrière : une fourmi peut revenir à un état déjà parcouru car la decision qu'elle avait prise à cet état a été mauvaise.

2.3 Expériences

2.3.1 Pont binaire de Deneubourg [Dréo 04]

L'expérience montre un nid d'une colonie de fourmis, qui est séparé d'une source de nourriture par un pont à deux voies de même longueur. On laisse évoluer les fourmis sur le pont, on trace ainsi

en fonction du temps, le graphe du nombre de fourmis empruntant chaque branche. Le résultat de l'expérience est exposé à la figure 2.1.

L'illustration (a) représente la configuration physique de l'expérience. Le graphique (b) indique l'évolution de ce système en fonction du temps : on constate que les fourmis ont tendance à emprunter le même chemin (par exemple celui du haut) après une dizaine de minutes.

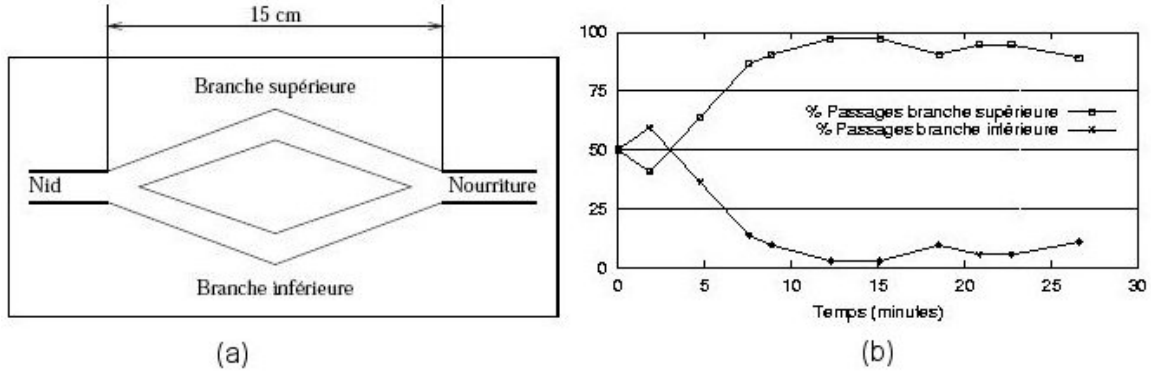


FIG. 2.1 – Pont binaire de Deneubourg.

Explication

Au départ, il n'y a pas de phéromone sur le pont. Donc, chaque branche peut être choisie par une fourmi avec la même probabilité. Néanmoins, dans notre exemple, après une certaine période, des variations aléatoires ont fait qu'un peu plus de fourmis ont choisi le chemin du haut plutôt que celui du bas.

Puisque les fourmis déposent des phéromones en avançant et puisqu'elles sont plus nombreuses en haut qu'en bas, le chemin du haut comportera plus de phéromones. Cette quantité supérieure de phéromone incite plus de fourmis à choisir la branche du haut, donc la quantité de phéromone déposée augmentera encore plus.

On en déduit que plus les fourmis suivent un chemin, plus ce chemin devient interessant à suivre. Ainsi la probabilité avec laquelle une fourmi choisit un chemin, augmente avec le nombre de fourmis qui ont pris ce chemin précédemment.

2.3.2 Expérience du double pont binaire [Dorigo 03]

On peut se demander à présent quel serait l'effet de l'augmentation de la longueur d'une des deux branches du pont. L'effet produit sera que la branche la plus courte sera sélectionnée.

En effet, les premières fourmis qui reviennent au nid avec de la nourriture sont celles qui ont emprunté le chemin le plus court dans les deux sens. Ce chemin, marqué deux fois par les phéromones, attire plus les autres fourmis que le long chemin, qui lui est marqué une seule fois dans le sens de l'aller. Cet effet se renforce au fur du temps, jusqu'à ce que toutes les fourmis choisissent le chemin le plus court.

C'est ainsi que dans cette expérience, on voit que les variations aléatoires sont réduites, puisque les deux chemins n'ont plus la même longueur. Contrairement à la première expérience, le comportement des fourmis qui consistait à suivre les pistes de phéromones n'est plus le seul mécanisme

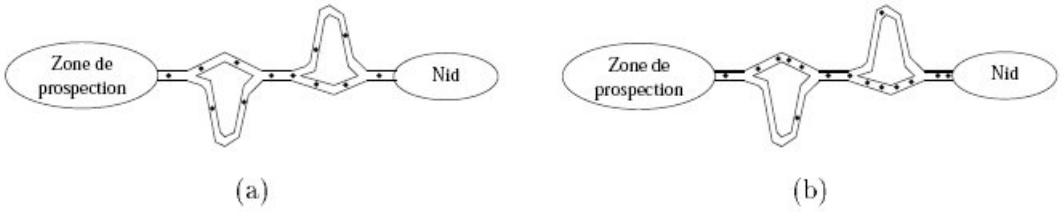


FIG. 2.2 – Expérience du double pont binaire.

présent : maintenant on associe ce mécanisme à une notion de *distance*.

Toutefois, quand le chemin le plus court n'est ouvert qu'après le chemin plus long (par exemple, quand le chemin était au départ bloqué par un obstacle), les fourmis continuent de parcourir le chemin le plus long car c'est le seul à être recouvert de phéromone. C'est ainsi que , l'évaporation des phéromones joue un rôle capital : les pistes de phéromones sont moins présentes sur les chemins les plus longs, car le réapprovisionnement en phéromone demande plus de temps que sur les chemins plus courts. Donc il suffit alors qu'une poignée de fourmis choisissent le chemin auparavant bloqué pour favoriser celui-ci. Ainsi, même quand des voies plus courtes ont été découvertes après, les fourmis finissent par les emprunter.

On peut généraliser cela à plus de deux chemins possibles : dans la figure 2.2 (a), on a utilisé un double pont avec quatre chemins possibles de différentes longueurs. On voit dans le dessin (b) que la plupart des fourmis finissent par choisir le chemin le plus court. Les expériences montrent que quand environ cent fourmis ont déjà emprunté le pont, plus de 90 pourcents d'entre elles sélectionnent le chemin le plus court : les fourmis convergent donc assez rapidement.

2.3.3 Effet de la coupure d'une piste de phéromone [Dréo 04]

Cette fois, les fourmis sont en train de suivre une piste de phéromones, comme présenté à la figure 2.3 (a). À un moment donné, on a un obstacle qui barre la route des fourmis. Les fourmis qui arrivent à côté de l'obstacle doivent choisir soit d'aller à gauche soit d'aller à droite (b). Puisqu'aucune phéromone n'est déposée le long de l'obstacle, il y a autant de fourmis qui partent à gauche qu'à droite.

Néanmoins, puisque le chemin de droite est plus court que celui de gauche, les fourmis qui l'empruntent, vont retrouver plus vite la piste de phéromone de départ. Pour chaque fourmi allant du nid à la nourriture, on associe également une fourmi qui va de la nourriture au nid (en fait elles ont été séparées par l'apparition brutale de l'obstacle). Les phéromones de ces fourmis vont se superposer à droite. Donc quand elles vont rejoindre le chemin initial, le chemin de droite sera deux fois plus imprégnée de phéromone que la piste de gauche, où les deux fourmis n'ont pas encore pu rejoindre la piste initiale (ce chemin étant plus long).
Les fourmis qui arrivent à l'obstacle à partir de ce moment, préféreront suivre la piste de droite. Le nombre de fourmis qui passent par la droite va augmenter, ce qui augmentera encore la concentration de phéromones. De plus, l'évaporation des phéromones sera plus forte sur la piste de gauche du fait que sa longueur est supérieure. La piste de gauche sera dont rapidement abandonnée, parce qu'elle en est beaucoup moins imprégnée : les fourmis passeront toutes très rapidement par la piste la plus courte.

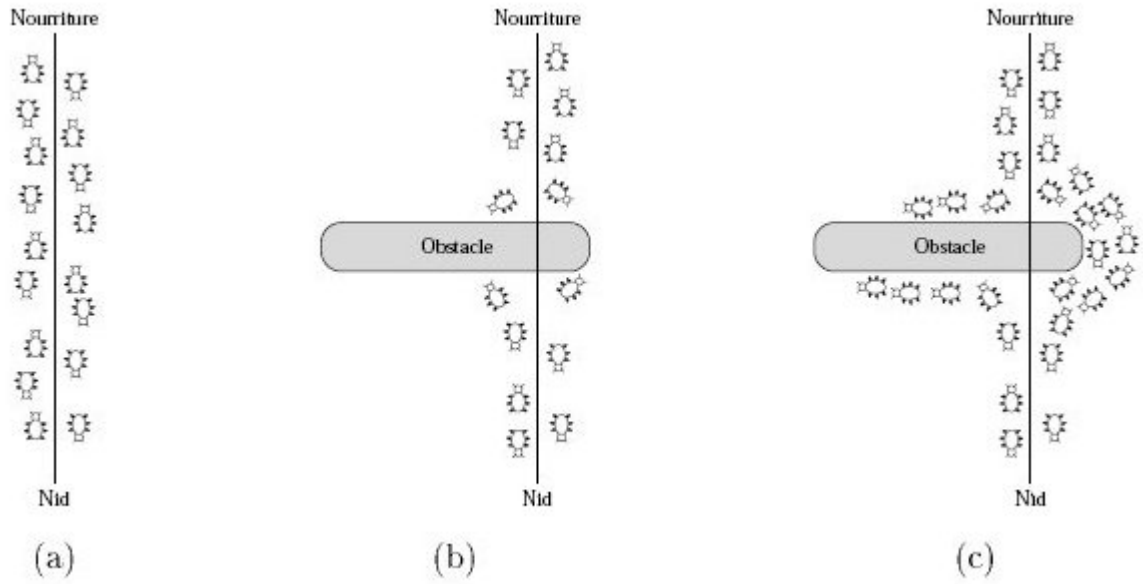


FIG. 2.3 – Effet de la coupure d'une piste de phéromone.

2.3.4 Conclusions

Il est intéressant de remarquer que bien qu'une seule fourmi soit capable de construire une solution (i.e. de trouver un chemin du nid à la nourriture), c'est seulement le comportement de l'*ensemble* de la colonie qui crée le chemin le plus court [Dréo 04].

Chapitre 3

Voyageur de commerce : Algorithme Ant System (AS)

3.1 Introduction

3.1.1 Définitions

Données fournies [Dorigo 02] :

- un ensemble fini X de noeuds représentant des villes
- un ensemble fini $U = \{(i, j)|i, j \in X\}$ d'arcs reliant les noeuds de X
- un ensemble de constantes d_{ij} représentant la longueur de chaque arc (i,j) ∈ U , c'est-à-dire la distance séparant la ville i de la ville j (avec i,j ∈ X).

On imagine alors un voyageur de commerce qui doit réaliser sa tournée en visitant une et une seule fois l'ensemble X des villes. Il souhaite déterminer la suite de villes qui minimisera la distance qu'il a à parcourir.

Formalisation [Dorigo 02] :

- **circuit hamiltonien** est un circuit qui passe exactement une fois par tous les sommets du graphe.

- la **longueur d'un circuit** μ est la somme des longueurs des arcs qui le composent, soit :

$$L(\mu) = d_{u_q,u_1} + \sum_{i=1}^{q-1} d_{u_1,u_{i+1}}$$

- le **TSP** (Traveling Salesman Problem) est le problème consistant à trouver un circuit hamiltonien de longueur minimale sur le graphe G = (X,U).

Le voyageur de commerce est un problème NP-complet. La méthaphore de la colonisation de fourmis s'y applique particulièrement bien.

3.2 Ant System

3.2.1 Conventions

Ant System sera par la suite appelé AS. Les variables que nous devons définir pour comprendre la suite sont les suivantes [Dorigo 02] :

- $b_i(t)(oui \in X)$ le nombre de fourmis dans la ville i à l'instant t
- $m = \sum_{i \in X} b_i$ leur nombre total, invariant dans le temps
- $\tau_{ij}(t)$ la valeur de τ_{ij} , à l'instant t
- $n = |X|$, le nombre de villes
- $\eta_{ij} = \frac{1}{d_{ij}}$, la *visibilité* d'une ville j quand on est placé sur la ville i, invariante dans le temps.

Précisons maintenant le comportement de l'ensemble de la colonie. A tout instant t, chaque fourmi choisit une ville de destination selon un choix défini. Toutes les fourmis se placent a l'instant $t+1$ dans une ville de leur choix. On appelle une itération de l'algo AS, l'ensemble de deplacements de l'ensemble de la colonie entre l'intant t et l'instant $t+1$. Ainsi après n itérations, l'ensemble de la colonie aura effectué un circuit hamiltonien sur le graphe. De cette manière toutes les fourmis commenceront et finiront leur tour en meme temps.

3.2.2 Choix d'implementation

On précise que chaque fourmi a une mémoire implémentée par une liste de villes déja visitées. Cela permet de garantir qu'aucune fourmi ne visitera deux fois une même ville au cour de sa recherche.

La mémoire de chaque fourmi est vidée lorsqu'elles ont terminé leur cycle.

Choix des transitions [Dorigo 02]

Une fourmi k placée sur la ville i à l'instant t va choisir sa ville j de destination en fonction de la visibilité η_{ij} de cette ville et de la quantité de phéromones $\tau_{ij}(t)$ déposée sur l'arc reliant ces deux villes. Ce choix sera réalisé de manière aléatoire, avec une probabilité de choisir la ville j donnée par :

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in N_i^k(t)} [\tau_{il}(t)]^\alpha \cdot \eta_{il}^\beta} & si\ j \in N_i^k \\ 0 & sinon \end{cases} \quad (1)$$

où l'on défini l'ensemble N_i^k comme étant l'ensemble des villes que la fourmi k, placée sur la ville i, n'a pas encore visité à l'instant t dans le cycle courant. α et β sont deux paramètres qui contrôlent l'importance relative entre phéromones et visibilité. Ainsi si α est égal à 0, le choix se fera uniquement en fonction de la visibilité (si β est différent de zero).

Mise à jour des phéromones [Dorigo 02]

A la fin de chaque cycle (chaque fourmi a parcouru les n sommets qui composent le graphe), les variables des phéromones sont mises à jour selon la formule :

$$\tau_{ij}(t+n) = \rho \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \quad (2)$$

où $\rho \in [0, 1[$ est un coefficient qui définira la vitesse d'évaporation des phéromones sur les arcs entre l'instant t et l'instant $t+n$, et où $\Delta\tau_{ij}(t)$ représente la quantité de phéromone déposée par

les fourmis dans ce même intervalle de temps sur l'arc (i, j).
Le choix de ρ est important, en effet si ρ se rapproche trop de 1, on observe un effet de stagnation des phéromones sur les arcs, ce qui implique des inconvenients tel que le fait de voir les mauvaises solutions persister.
De même, choisir $\rho \approx 0$ implique une évaporation trop rapide des phéromones, donc amène la fourmi à un choix dépendant uniquement de la visibilité des noeuds.

Quantités de phéromones déposées [Dorigo 02]

Appelons $T_k(t) = (u_{k_1}, \ldots, u_{k_q})$ le tour réalisé par la k-ème fourmi dans l'intervalle de temps $[t, t+, n]$, et $L_k(t)$ sa longueur. $T_k(t)$ (et donc $L_k(t)$) s'obtient en analysant la mémoire de la fourmi. Soit $\Delta\tau_{ij}^k(t)$, la quantité de phéromones déposée par cette fourmi sur l'arc (i, j) dans ce même intervalle de temps. On le définit ainsi :

$$\Delta\tau_{ij}^k(t) = \begin{cases} Q/L_k(t) \; si(u \in T_k(t) \; \wedge \; u \; = \; (i,j)) \\ 0 \qquad\quad sinon \end{cases} \tag{3}$$

où Q est un constante. On voit bien ici que les phéromones sont régulés en fonction de la qualité de la solution obtenue car plus $L_k(t)$ est faible plus l'arc sera mis à jour en phéromones.
On peut maintenant définir le $\Delta\tau_{ij}(t)$ de la formule de mise à jour des phéromones ainsi :

$$\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t) \tag{4}$$

3.3 Fonctionnement de l'algorithme

Nous allons maintenant préciser les différentes étapes de l'algorithme au cours du temps.

Initialisation de l'algorithme. Les élements s'agencent de la manière suivante au début de l'algorithme :

1. les m fourmis sont réparties aléatoirement sur les n villes.
2. Pour chaque fourmi, la liste qui modélise sa mémoire contient sa ville de départ.
3. Les pistes de phéromones sont initialisées comme suit : $t_{ij}(0) = c$, où c est une petite constante positive, qui ne peut être nulle (sinon, il y a un problème lors du calcul de (1))

Fin d'un cycle. Après n itérations, nous sommes à l'instant t, toutes les fourmis ont terminé leur tour, chacune a une liste "mémoire" pleine et est revenue à sa propre ville de départ. À ce moment :

1. Chaque fourmi calcule sa valeur $L_k(t)$.
2. Les variables $\tau_{ij}^k(t)$ sont calculées conformément à la formule (1).
3. Les variables de phéromone $\tau_{ij}(t)$ sont mises à jour suivant la formule (2). En d'autres termes, la fourmi refait son tour en sens inverse tout en déposant des phéromones.
4. On observe quelle fourmi a trouvé le tour de longueur minimum (i.e. on recherche la fourmi k telle que $k = min_{k=1}^m L_k(t)$). Si ce tour est meilleur que le meilleur tour jusqu'ici, on le mémorise.
5. Les mémoires des fourmis (liste des villes visitées) sont effacées.

6. Les fourmis recommencent un nouveau tour, toujours au départ de la ville sur laquelle elles avaient été placées au début de l'algorithme.

Fin de l'algorithme. On arrête l'algorithme après un nombre de cycles égal à une constante NC_{max}. Si à partir d'un instant, toutes les fourmis font le même tour, l'algorithme s'interrompt : on est dans une situation de stagnation où le programme arrête de chercher des alternatives. L'algorithme donne en retour le meilleur tour mémorisé.

Les paramètres permettant de caractériser complètement une instance de AS sont repris dans les tuplet : $< m, p, \alpha, \beta, Q, NC_{max} >$,où $0 \leq p < 1$,$\alpha \geq 0$, $\beta \geq 0$ et $c > 0$.

3.3.1 Complexité

Afin de se faire une idée, nous divisons l'algorithme en 5 sections pour calculer la complexité [Dréo 04] :

1. initialisation.
2. un cycle de l'algorithme.
3. fin du cycle et calcul des dépôts de phéromone.
4. Evaporation des phéromones.
5. boucle de l'algorithme.

Procédons étape par étape :

1. On a une complexité $O(|L| + m) = O(n^2 + m)$, puisque l'on a supposé une interconnexion totale entre les villes.
2. La complexité est $O(n^2 \cdot m)$, puisque les opérations de calcul de la ville suivante nécessite un balayage de l'intégralité des villes.
3. La complexité est $O(|L| + m \cdot |L|) = O(m \cdot |L|) = O(n^2 \cdot m)$.
4. La complexité est $O(|L|) = O(n^2)$.
5. Le test de stagnation est de complexité $O(n \cdot m)$ (on doit comparer les tours de m fourmis, chaque tour ayant une longueur de n éléments).

La complexité globale est obtenue en additionnant la complexité de l'étape 1, au produit du nombre total de cycle (soit NC_{max}) par la complexité globale des étapes 2 à 5. La complexité globale étant celle maximale, soit $O(n^2 * m)$. La complexité générale de l'algorithme devient donc : $O(n^2 + m + NC_{max} \cdot n^2 \cdot m)$

soit : AScomplexity $= O(NC_{max} \cdot n^2 \cdot m)$

Il faut noter cependant que cette formule ne nous dit rien sur le nombre d'étapes qui sont effectivement nécessaires pour atteindre l'optimum : on pourrait atteindre celui-ci bien avant les NC_{max} cycles.

3.3.2 Variantes [Dorigo 03]

L'algorithme que nous venons de présenter s'appelle en fait AS-ant-cycle. Historiquement, il y a eu deux autres versions de AS, qui étaient différentes par la manière dont les pistes de phéromones étaient mises à jour. Dans ces deux modèles, le dépôt de phéromones était entrelacé avec la progression des fourmis et ne se déroulait pas à la fin de chaque cycle. Il s'agit en fait du fonctionnement des fourmis réelles : à chaque pas, la fourmi dépose un peu de phéromone. Ces deux autres algorithmes diffèrent entre eux par la quantité de phéromones déposée à chaque pas :

AS - ant-density : $\Delta\tau_{ij}^k(t) = \begin{cases} Q \ si \ la \ fourmi \ va \ de \ i \ a \ j \ dans \ l'intervalle \ [t, t+1] \\ 0 \ sinon \end{cases}$

AS - ant-quantity : $\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{d_{ij}} \ si \ la \ fourmi \ va \ de \ i \ a \ j \ dans \ l'intervalle \ [t, t+1] \\ 0 \ sinon \end{cases}$

On peut remarquer que, du fait de la présence du coefficient d_{ij}, ant-quantity rend plus désirables les transitions plus courtes. La justification de ant-density est qu'il n'y a pas besoin de faire intervenir une telle distinction, puisqu'elle est déjà apparente dans le calcul des probabilités de transitions (cf la formule (1)). Cette dernière idée a d'ailleurs été reprise dans ant-cycle.

Ces algorithmes ont vite été abandonnés, car leurs performances étaient nettement en dessous de celles de ant-cycle. Cette chute de performances s'explique intuitivement par l'absence de statistiques sur la compétitivité du tour qui est en train d'être construit : dans ant-cycle, on connaît L_k qui est cet indicateur de compétitivité. Dans la suite, quand nous parlerons de AS, nous sous-entendrons AS-ant-cycle.

3.4 Choix des paramètres

3.4.1 Considérations générales [Dréo 04]

Une grande faiblesse de AS est le nombre élevé de paramètres en jeu. De ce fait, il est très difficile de les doser subtilement.

Ceci est sans doute aussi une raison à l'absence de modèle mathématique permettant de justifier AS. On pourrait d'ailleurs s'attendre à ce que si un tel modèle existait il ne permettrait sans doute pas d'ajuster les paramètres de manière optimale.

En l'absence d'argumentation rigoureuse, les seules justifications à la disposition des chercheurs pour fixer les paramètres de AS sont des résultats expérimentaux. La synthèse de ces résultats est presentée dans la section suivante.

3.4.2 Résultats expérimentaux

Les expériences ont été réalisées sur des ensembles de villes particuliers. Certaines conclusions en ont été tirées et nous proposons de les reprendre ici :

1. On pose $p = 0,5$ et $c \approx 0$, seule la stratégie gloutonne est a l'oeuvre au debut de l'algorithme : ceci est souhaitable, car les pistes de phéromones ont été généré artificiellement et n'ont donc aucune signification physique. Ensuite, grâce au paramètre $p > 0$, les fourmis commencent progressivement à exploiter les valeurs τ_{ij} au fur et à mesure que les pistes de phéromones se renforcent. Afin de ne pas être esclave des choix anciens (i.e. de ne plus tenir compte que des τ_{ij}), il faut que la visibilité ait encore un impact. Ceci explique le choix selon lequel $p = 0,5$.
2. La recherche est plus efficace quand les fourmis communiquent entre elles, c'est-à-dire quand les quantités de phéromones ont un grand impact sur la politique de décision de la fourmi. Ceci correspond à prendre $\alpha \approx 1$ (cf formule (1)).
3. Il existe un optimum quand $m \approx n$, où la coordination des fourmis semble atteindre son maximum.
4. Lors de l'initialisation de l'algorithme, il semble toujours plus intéressant de distribuer les fourmis sur toutes les villes plutôt que de toutes les faire démarrer d'une même ville. On entend de ce fait de placer sur chaque ville le même nombre de fourmis. Ceci impose que m

soit multiple de n. Du fait du point précédent, on choisira donc toujours $m = n$. Il a aussi été constaté qu'une distribution aléatoire des fourmis n'a qu'un impact négligeable sur la convergence. On place donc une fourmi par ville initialement.

5. Le paramètre Q n'a qu'une influence négligeable. Cela se comprend, puisque dans la politique de choix des fourmis, on utilise un rapport entre quantités de phéromones et non pas directement ces quantités. On pose généralement $Q = 100$.

Pour résumer, on a observé que les résultats suivants donnent en pratique les meilleurs résultats pour un problème à 30 villes : $n = 30$, $NC_{max} = 5000$, $\alpha = 1$, $\beta = 1$, $p = 0,5$ et $Q = 100$.

3.4.3 Les fourmis élitistes

Nous présentons ici une amélioration qui donne de bons résultats en pratique. Elle consiste à renforcer de manière artificielle à chaque étape les quantités de phéromone présentes sur les arcs appartenant au meilleur tour trouvé jusqu'à present.

Intuitivement, cette méthode consiste à faire reparcourir le meilleur tour par certaines fourmis artificielles, dites fourmis élitistes. Ces fourmis sont conceptuellement identiques aux autres fourmis, si ce n'est qu'elles choisissent leur chemin de manière déterministe et qu'elle ne se mettent en route que quand toutes les autres fourmis ont terminé un tour.

Si l'on désigne par T^* le meilleur tour et par L^* sa longueur, à chaque fin de cycle on réalise l'opération suivante :

$$\tau_{ij}(t) \leftarrow \tau_{ij}(t) + \begin{cases} e \cdot \frac{Q}{L^*} \ si \ (i,j) \in T^* \\ 0 \ sinon \end{cases}$$

où e est le nombre de fourmis élitistes.

L'effet des fourmis élitistes est de grandement accroître la convergence de la méthode (typiquement d'un facteur 10), au risque que celle-ci converge vers une solution non-optimale. Cette amélioration de la convergence s'explique très bien intuitivement, puisque cela revient à favoriser un chemin parmi ceux déjà explorés. Elle est précieuse dans l'AS qui reste un algorithme très gourmand en ressources de calcul.

Il y a donc une valeur optimale à e. Quand nous sommes sous cette valeur et que nous augmentons e, le meilleur tour peut être découvert plus tôt. Quand nous sommes au-delà de cette valeur, les fourmis élitistes forcent l'exploration autour de circuits non-optimaux dès les premières étapes de la recherche, ce qui amoindrit les performances.

Notons aussi que la présence de fourmis élitistes peut même permettre l'émergence de meilleurs tours. En effet, les fourmis élitistes sont un mécanisme qui permet de focaliser la recherche sur des zones prometteuses de l'espace de recherche en augmentant l'attrait des fourmis pour des pistes que l'on sait faire partie d'un bon tour. Or, quand un bon tour a été découvert, il y a de forte chance que le tour optimal n'en est pas très éloigné.

Exemple Dans le problème à 30 villes cité plus haut, la valeur optimale de e est 8 (bien que des valeurs allant de 2 à 16 donnent aussi de bons résultats) : on passe de 2500 cycles à 250 seulement pour trouver une solution qui est même légèrement meilleure en termes de distance à la solution optimale.

On pourrait supposer qu'accroître e en fonction de la progression de l'algorithme évite de favoriser au début de la recherche des tours trop long. À notre connaissance, cette possibilité n'a pas encore été étudiée par les chercheurs.

3.4.4 Résumé

Un paramétrage qui donne de bons résultats en pratique est le suivant [Dréo 04 mesures] :

paramètres	**valeurs**
m	n
p	$0,5$
α	1
β	5
Q	100
c	Petite valeur positive non nulle
Distribution initiale	Uniforme

Si l'on choisit d'utiliser les fourmis élitistes pour améliorer la convergence, on pose $e = 8$. Le NC_{max} est fixé en fonction des ressources de calcul qui sont allouées à la résolution du problème. Rien n'interdit d'interrompre la recherche selon un critère probabiliste sur la distribution de la population des $L_k(t)$. Puisque $m = n$, la complexité de l'algorithme donnée par la formule (3.4) devient :
AScomplexity = $O(NC_{max} \cdot n^3)$ (3.5)
Au final, AS est un algorithme cubique, car NCmax est une constante.
Attirons une nouvelle fois l'attention sur le fait que ces paramètres ont été ajustés expérimentalement par les chercheurs et qu'ils ne possèdent aucune justification théorique. Ce qui fonctionne bien pour les "problèmes-étalons" peut très bien ne pas fonctionner dans le cas général. Les études à ce sujet restent plutôt floues.

Chapitre 4

Performances de l'algorithme Ant System

Les résultats expérimentaux fournis dans les premières implémentations étaient intéressants d'un point algorithmique mais peu performants. Il s'agissait de tester AS sur des petites instances du TSP (entre 30 et 75 villes).

AS (sans fourmis élitistes) est capable de trouver la solution optimale (exacte) pour le problème de 30 villes. Cette solution est atteinte en moins de 400 cycles et des solutions très proches de la solution optimales sont atteintes en environ 100 cycles.

Malheureusement, pour des problèmes de dimensions croissantes, AS n'a jamais pu trouver une solution optimale en moins de 3000 cycles, bien qu'il présente une rapide convergence vers de bonnes solutions.

Il s'agit donc de résultats encourageants, mais non exceptionnels. C'est la raison pour laquelle, de nombreuses améliorations sur la méta-heuristique ACO ont été faites.

4.1 Présentation générale

Nous allons tenter de concrétiser le fonctionnement de AS. Dans le cadre de ce travail, on s'est inspiré des travaux de Dorigo sur l'implémentation de AS [Dorigo 03 mesures].

Le programme qui a été testé est un programme écrit en C qui se nomme ACOTSP [Dorigo internet]. Les tests réalisés se basent sur un problème à 48 villes, le problème "Att48", qui représente une carte des grandes villes des États-Unis [Dréo 04 mesures]. La solution optimale pour ce problème est présentée à la figure 4.1. Elle a une longueur de 10628 km.

Un test de stagnation a été introduit dans le programme. En fait, on observe l'écart-type de la population de la longueur des tours. On divise ce nombre par la plus petite distance entre deux villes dans le problème, ce qui donne une bonne mesure du degré de stagnation pour le problème considéré. Si ce nombre descend sous un seuil fixé par l'utilisateur (nous avons fixé ici un seuil de 2), on considère que l'on est dans une phase de stagnation, donc on arrête l'algorithme. Dans les tests qui suivent, on a pas activé ce test, l'algorithme progresse jusqu'à atteintre NC_{max} cycles.

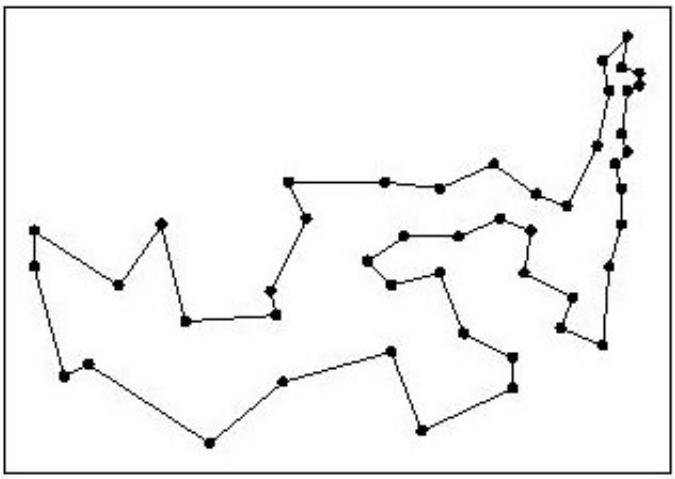


FIG. 4.1 – Le problème “Att48”

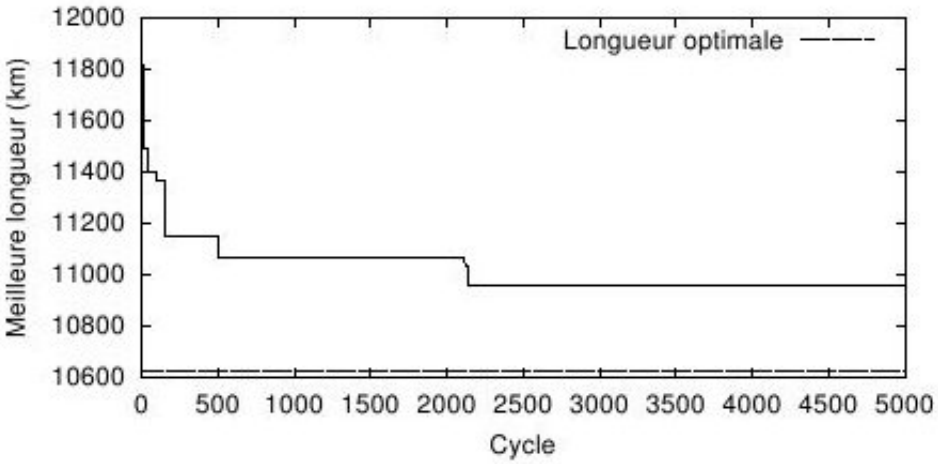


FIG. 4.2 – Évolution de la longueur du meilleur tour.

4.2 Résultats sans fourmis élitistes

Nous avons employé pour cette expérience les paramètres de la section Choix des paramètres, sans fourmis élitistes. Nous avons laissé tourné l'algorithme durant 5000 cycles. Le temps d'exécution total est de 4 minutes et 23 secondes sur un Pentium II 233 Mhz (ce test comprend les tests de stagnation, l'affichage des statistiques et l'exportation des résultats).

Le meilleur tour a été trouvé après 2139 cycles et une très bonne approximation est déjà disponible après environ 500 cycles [Dréo 04 mesures]. On peut observer l'évolution de la longueur des tours découverts à la figure 4.2. Ce tour avait une longueur avait une longueur de 10.957 km : sa distance relative à l'optimum était donc de $(10957 - 10628)/10628 = 3,09\%$. On remarque que AS découvre dans sa première phase de bons tours qui seront par la suite affinés.

Nous avons mesuré la concentration en phéromone sur chacune des pistes reliant les différentes

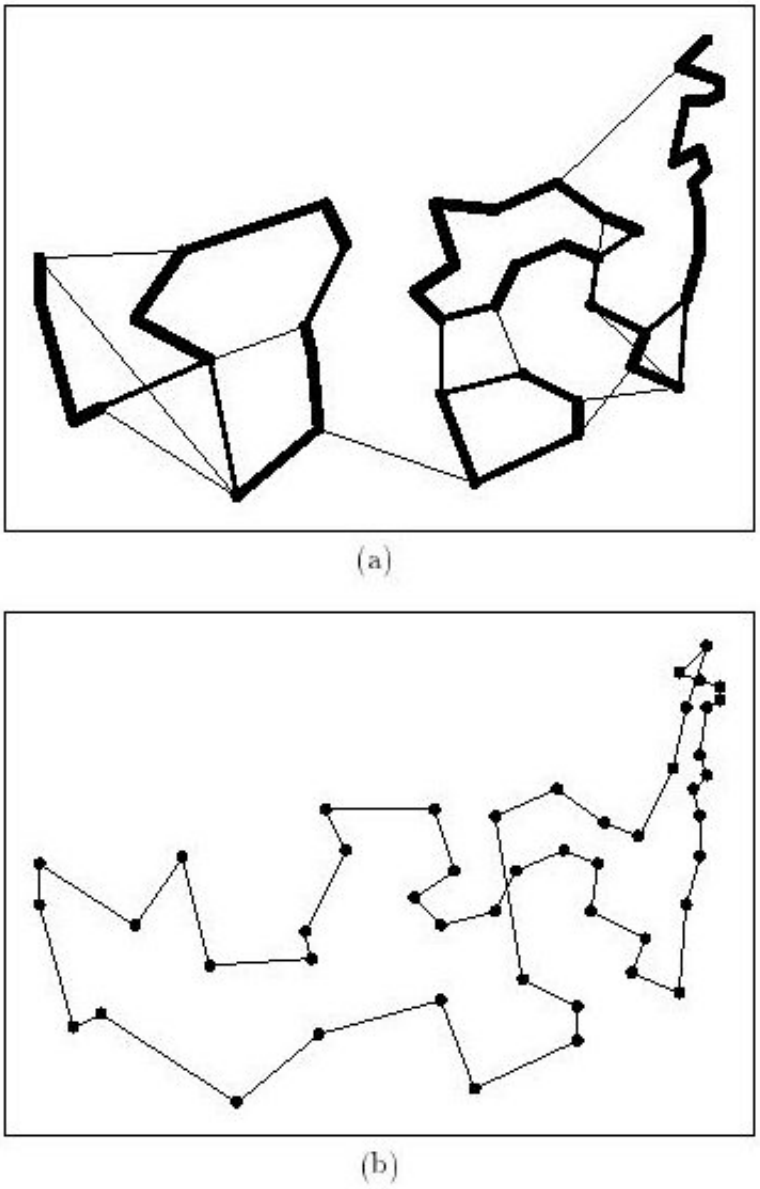


FIG. 4.3 – La meilleure solution trouvée et les quantités de phéromones correspondantes.

villes après les 5000 cycles. Nous avons porté ces quantités sur un graphique à la figure 4.3 (b). Ces deux dessins nous donnent une bonne idée intuitive du fonctionnement de l'algorithme.

L'évolution de l'écart-type des longueurs des tours parcours en fonction du temps est porté à la figure 4.4. On voit que cette valeur ne tombe jamais à zéro, ce qui assure que l'algorithme cherche toujours de nouvelles solutions, qui peuvent être meilleures que celles trouvées précédemment.

On essaye de chercher les meilleures solutions sur les quantités de phéromones déposées lors des cycles précédents. On peut vérifier cela en observant la figure 4.5, où l'axe verticial représente le nombre moyen d'arcs potentiellement empruntables par les fourmis à chaque noeud. Ce nombre correspond au nombre moyen d'arcs pour lesquels τ_{ij} n'est pas encore tombé sous une petite valeur ϵ : on considère que de tels arcs auraient une probabilité quasiment nulle d'être choisis par

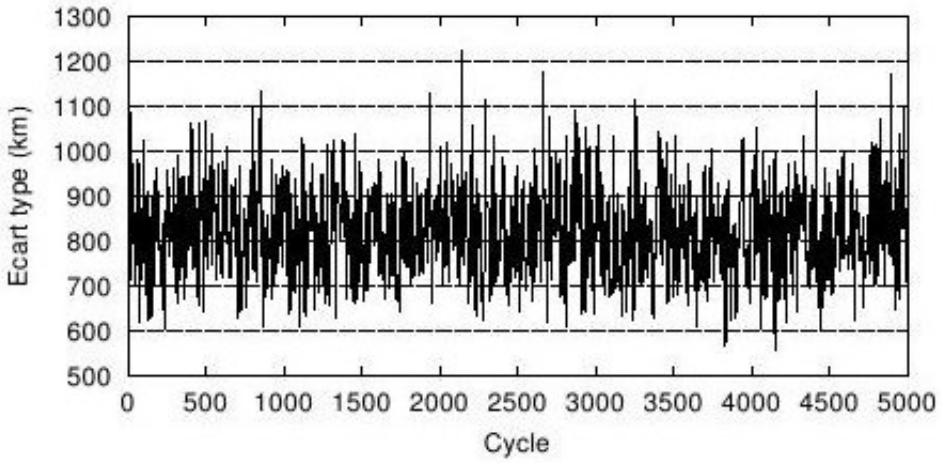


FIG. 4.4 – Écart-type de la population de la longueur des solutions.

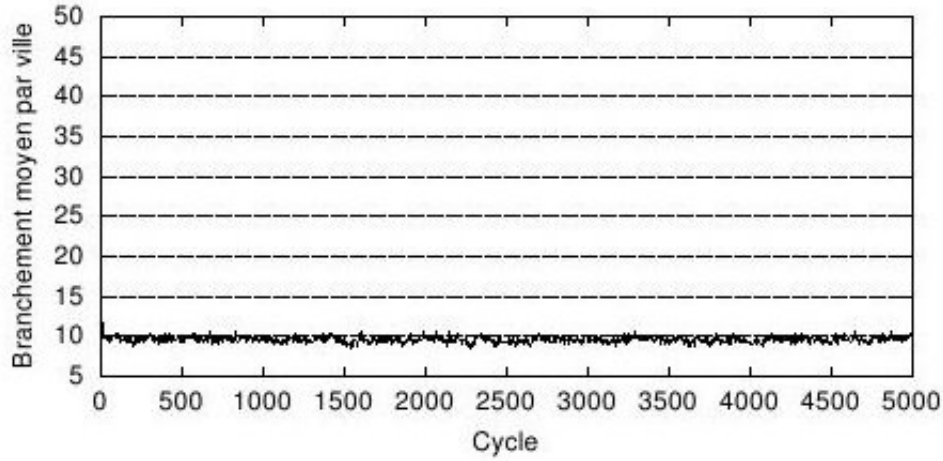


FIG. 4.5 – Nombre moyen de branchements possibles par noeud.

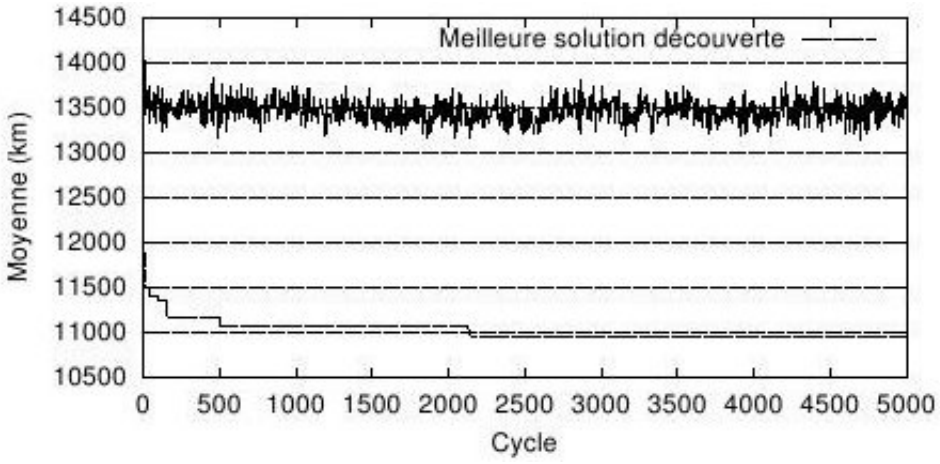


FIG. 4.6 – Comparaison de la longueur moyenne des tours avec la longueur du meilleur tour.

les fourmis.

Au début de l'algorithme, les fourmis sont totalement libres d'aller d'un noeud quelconque à nimporte quel autre (sauf ceux déjà visités). Mais très vite au cours de l'exécution, les choix possibles sont considérablement réduits pour en arrive à en moyenne 10 possibilités par ville. Si cette valeur tombait à deux, cela voudrait dire que l'algorithme stagne : arrivée à une ville, la voie de sortie s'impose à elle (le deux vient du fait qu'il faut une voie de sortie pour les deux sens de circulation des fourmis). Ce graphe prouve ainsi que l'algorithme ne cesse de rechercher de nouvelles solutions.

Ce dernier graphe met en évidence un défaut de AS : même si l'algorithme cherche activement de nouvelles solutions, la découverte d'un meilleur tour n'est pas suffisamment exploitée pour que la longueur moyenne des tours diminue pour converger vers l'optimum découvert, comme le montre la figure 4.6. En fait, AS améliore les sous-circuits déjà faits mais a des difficultés à les combiner entre eux.

4.3 Résultats avec fourmis élitistes

La même expérience a été menée avec huit fourmis élitistes. Le temps d'exécution était identique (on a aussi laissé courir l'algorithme sur 5000 cycles). Le meilleur tour a été trouvé au cycle 2579 [Dréo 04 mesures] : ce tour a une longueur de 10730 km, soit une une distance qui se rapproche du tour optimal ($0,96\%$ au lieu de $3,09\%$ évoqués plus haut).

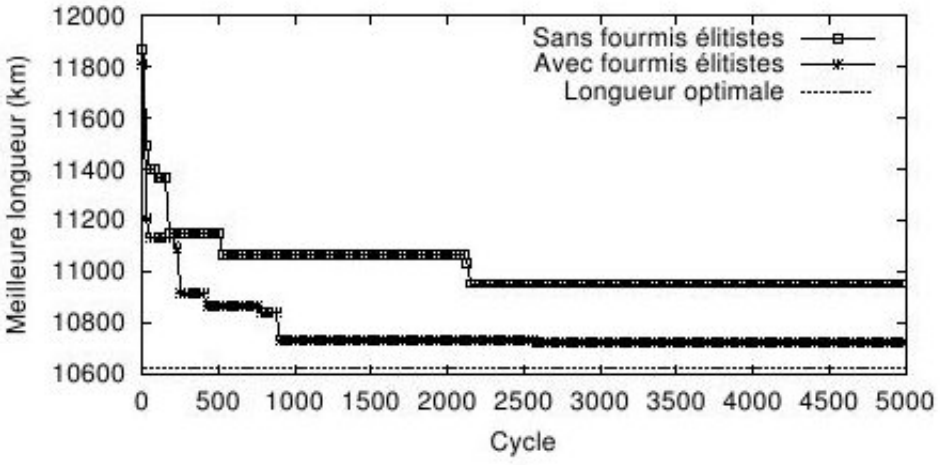


FIG. 4.7 – Comparaison de la longueur du meilleur tour avec ou sans fourmis élististes.

La figure 4.7 montre une comparaison de l'évolution de la longueur du meilleur tour au cours du temps selon que l'on utilise ou pas les fourmis élitistes. Grâce aux fourmis élitistes, AS converge plus vite vers une solution qui est plus proche de l'optimal .

La figure 4.8 présente cette meilleur solution. Il faut remarquer que les sous-zones de la figure 4.3 (page 20) sont nettement moins apparentes et que leur contour épouse cette fois le meilleur circuit hamiltonien pour cet ensemble de villes (voir dessin 4.1 à la page 19), ce qui indique donc un meilleur fonctionnement. Le graphe de l'écart-type est quant à lui identique à celui de la figure 4.4

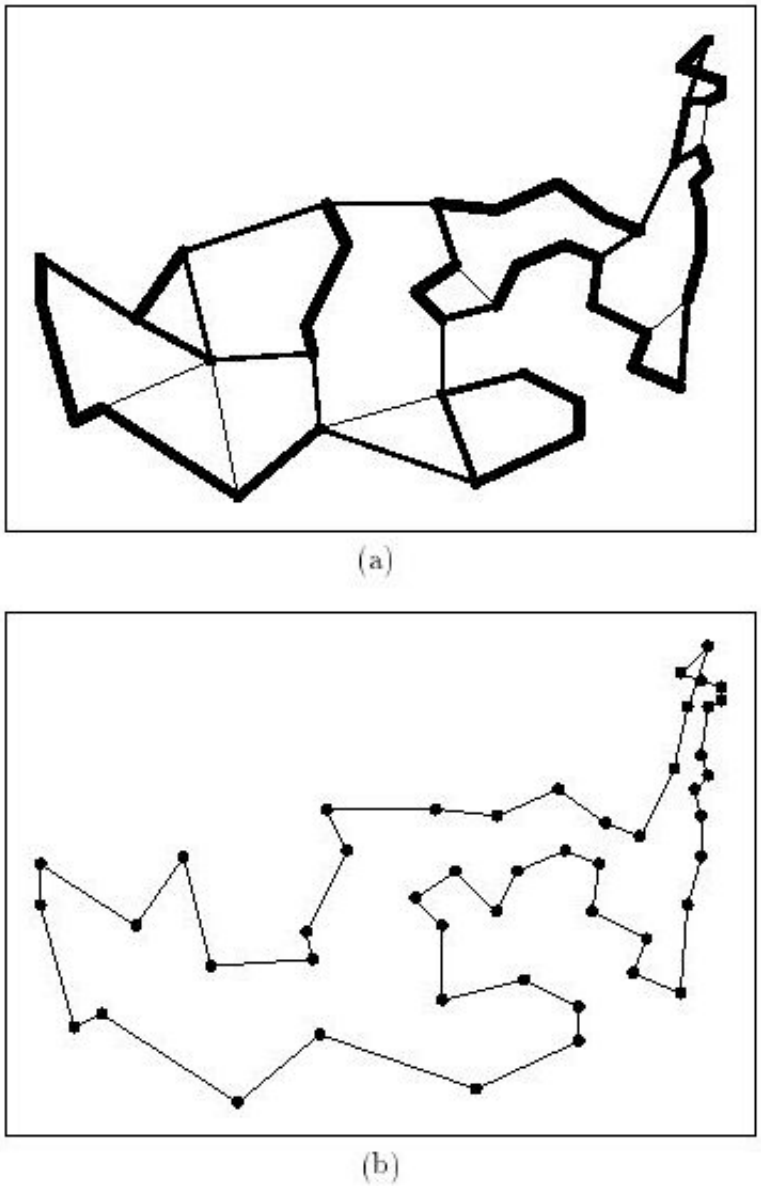


FIG. 4.8 – La meilleure solution trouvée avec fourmis élitistes et les quantités de phéromones correspondantes.

Le graphe de branchement est reproduit à la figure 4.10. On y découvre que le nombre de choix de transition est nettement réduit (on passe d'une moyenne de 10 à une moyenne de 8.5) : la méthode converge mieux en abandonnant les transitions qui sont trop éloignées d'un bon tour déjà découvert.

La figure 4.10 montre toutefois que la déficience déjà évoquée de AS reste présente : la longueur moyenne des tours ne converge pas vers celle du meilleur circuit découvert. Toutefois, la longueur moyenne des circuits est nettement inférieure à celle de la figure 4.6 (page 21). On en conclut que AS avec fourmis élitistes donne des résultats supérieurs à ceux de AS sans fourmis élitistes.

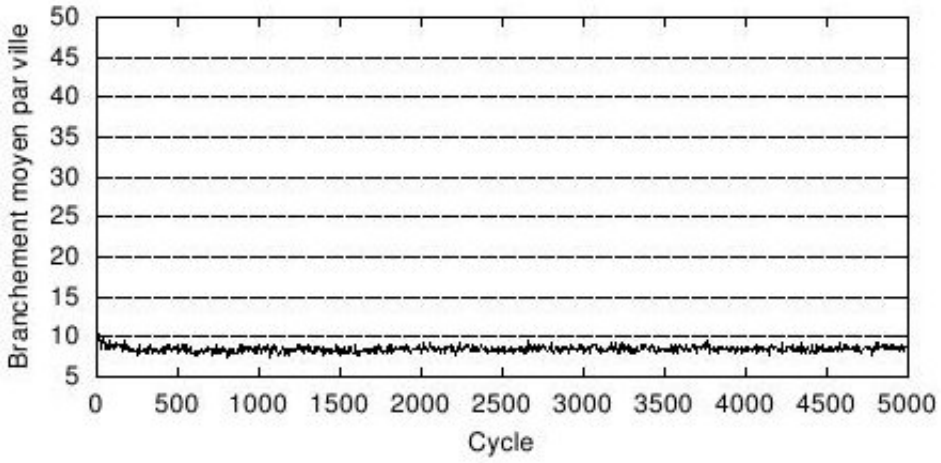


FIG. 4.9 – Nombre moyen de branchements possibles par noeud avec fourmis élitistes.

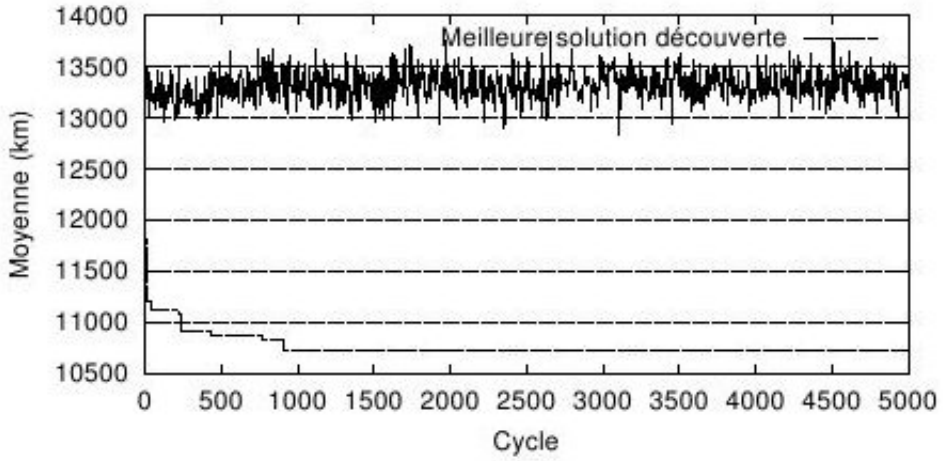


FIG. 4.10 – Comparaison de la longueur moyenne des tours avec la longueur du meilleur tour dans le cas des fourmis élitistes.

4.4 Applications sur d'autres tailles de données

4.4.1 Un plus petit problème ...

Nous avons appliqué AS avec fourmis élitistes à un problème de taille plus petit c'est à dire 16 villes (figure 4.11 à la page 26). Durant nos essais [Dréo 04 mesures], l'algorithme a toujours découvert le tour optimal en environ mille cycles.

L'exécution du programme a pris seulement 29 secondes pour calculer 2500 cycles et la solution optimale a dans ce cas précis été trouvée après 1154 cycles.

Remarquons que contrairement au problème à 48 villes, la longueur moyenne des tours converge vers l'optimum et que le facteur de branchement est réduit (de l'ordre de 4,5 transitions par ville).

L'impact de la déficience de AS semble donc réduit sur de petits problèmes. Toutefois, l'écart-type des longueurs des tours reste élevé (646 km pour un tour optimal de 6867 km de longueur, 1300 cycles après avoir atteint l'optimum).

4.4.2 ... et un plus grand

Notre dernière expérimentation [Dréo 04 mesures] a été effectuée sur un problème à cent villes (figure 4.12 à la page 27). Ce problème et la meilleur solution trouvée par AS avec fourmis élitistes sont présentés à la figure 3.13. Cette dernière solution est à $4,68\%$ de la longueur du tour optimal et a découverte dès le cycle 372.

Le temps total de calcul fut de 12 minutes 34 secondes. Ainsi, le meilleur tour fut découvert après seulement 2 minutes 20 secondes. AS traite donc assez rapidement le problème. L'écart de $4,65\%$ est assez bon, mais pas exceptionnel. D'autres statistiques peuvent être trouvées à la figure 4.13.

4.4.3 Conclusion des expériences

En guise de conclusion, il semble y avoir intérêt à exploiter les fourmis élitistes. Elles permettent d'arrivent plus vite à une solution plus proche de la solution optimale. Donc de ce fait à appliquer AS à de plus grand problèmes.

Il faut toutefois faire attention de ne pas donner à ces problèmes trop poids, afin de laisser à l'algorithme la possibilité de trouver des solutions meilleures que les précédentes.

AS présente l'avantage de continuellement rechercher de nouvelles solutions, comme le prouve les graphes de branchement. D'un autre côté, AS souffre d'une déficience : l'algorithme ne fait pas converger la moyenne de longueur des tours vers l'iptimum qui a été trouvé jusqu'à présent.

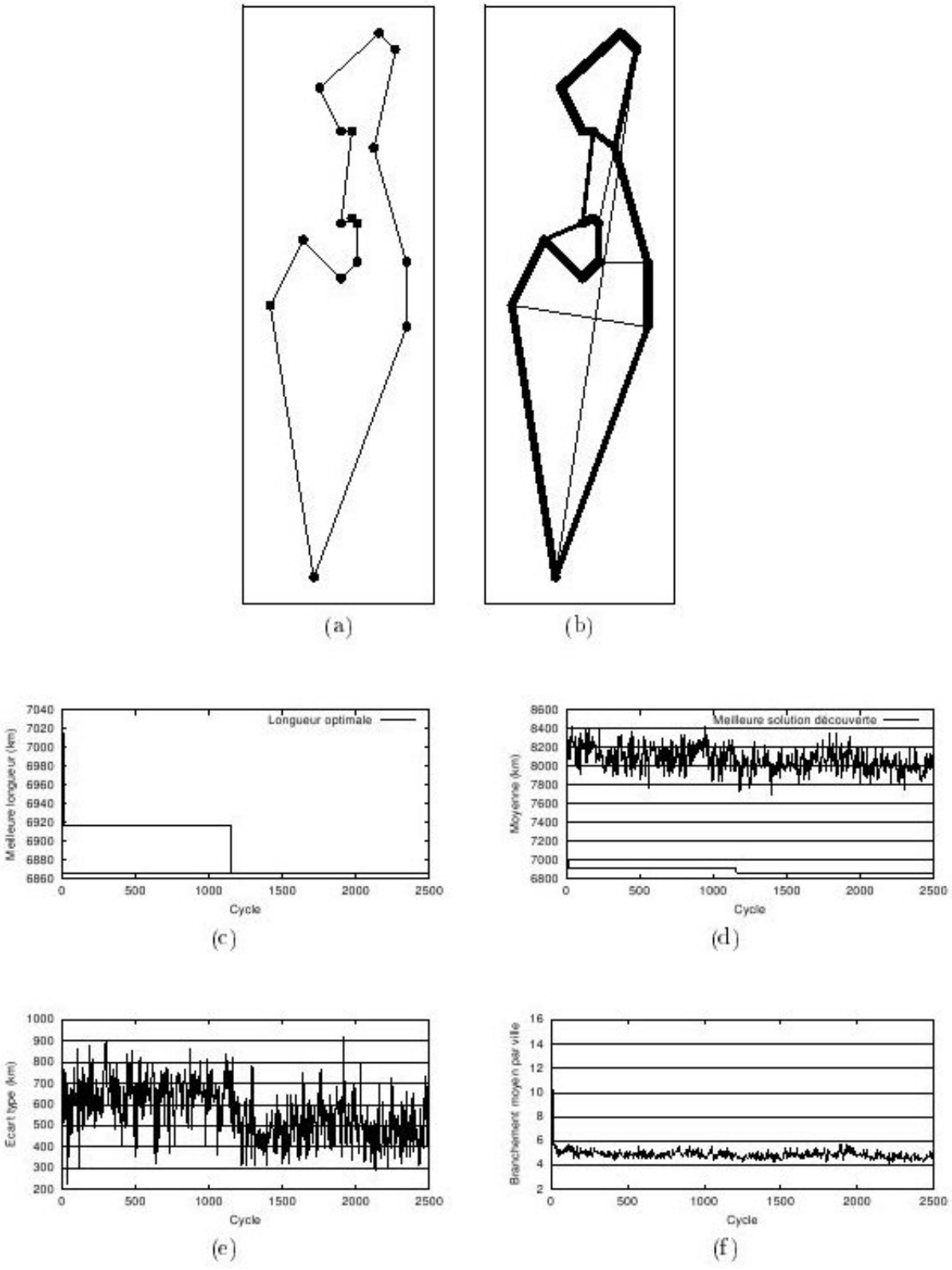


FIG. 4.11 – Statistiques pour le problème à 16 villes "Ulysses16".

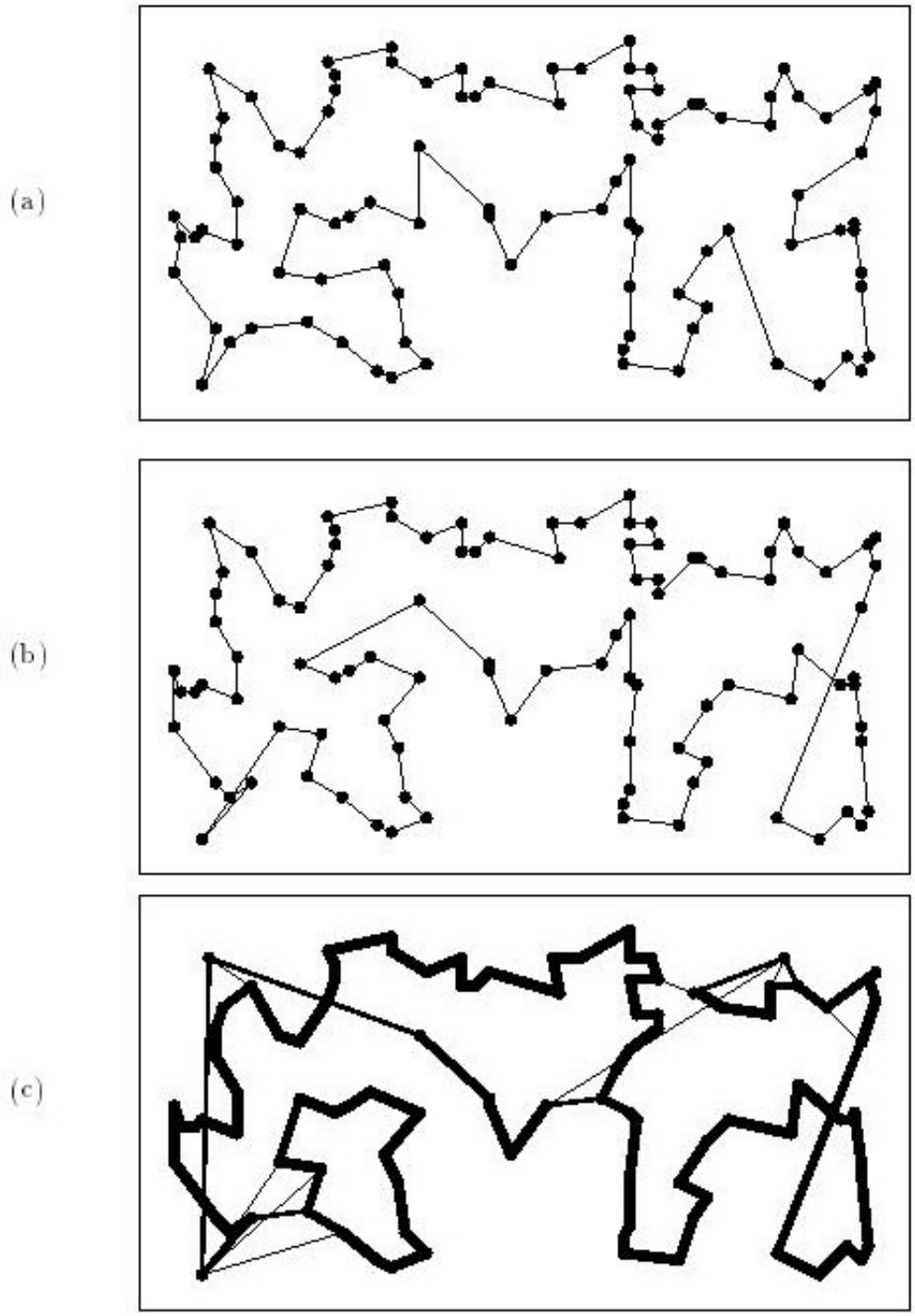
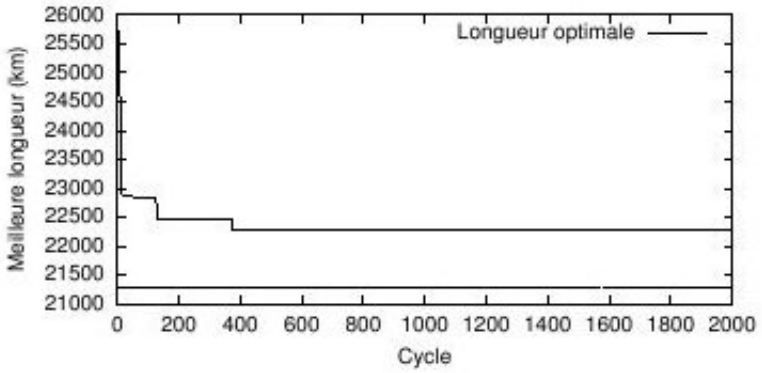
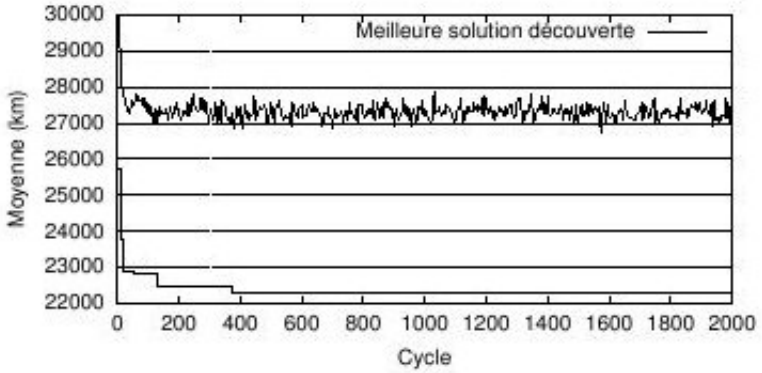


FIG. 4.12 – Le problème "kroA-100" : (a) circuit hamiltonien optimal, (b) meilleur tour trouvé par AS avec huit fourmis élitistes et (c) concentrations de phéromones correspondates.

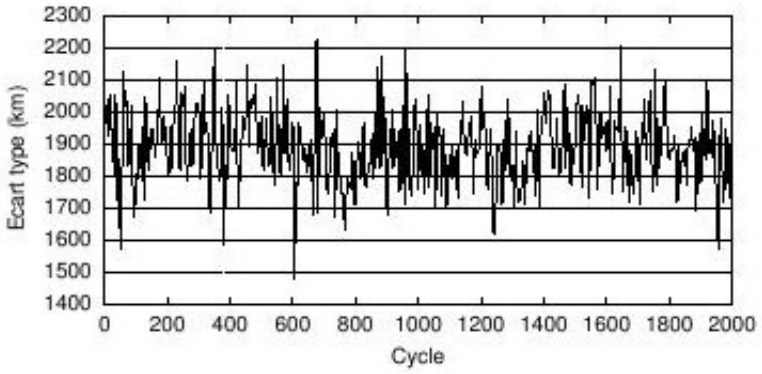
(a)



(b)



(c)



(d)

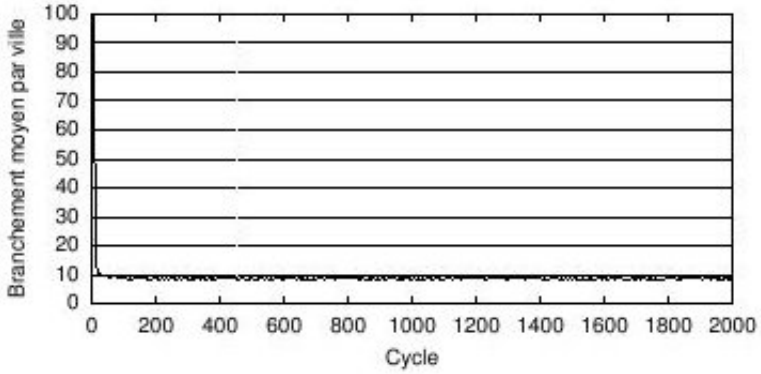


FIG. 4.13 – Statistiques pour le problème "kroA-100" : (a) meilleur longueur, (b) longueur moyenne des tours, (c) écart-type des tours et (d) nombre moyen de branchements possibles.

Chapitre 5

Améliorations plus récentes de Ant System

5.1 Évolution des algos TSP : Nouvelles approches

5.1.1 Introduction

Nous allons maintenant décrire l'évolution de AS . AS fut trouvé en 1991 et a subi depuis de évolutions qui ont abouti à Ant-Q , ACS , MMAS que l'ont va analyser par la suite [Dorigo 03] [Dorigo 02].

5.1.2 Ant-Q [Dorigo 02]

Différences

Ant-Q date de 1995 Cet algorithme reprend le meme fonctionnement que AS. La seule différence est qu'en plus d'une modification globale des phéromones aprés chaque cycle, les fourmis modifient pas-à-pas les phéromones sur l'arc quelle choisissent.

Formule

On met donc a jour l'arc que la fourmi vient de choisir. On suppose que la fourmi k choisit a l'instant t de passer de la ville i à la ville j (selon les règles de décision de AS), on met à jour la quantité de phéromones sur l'arc (i, j) ainsi :

$$\tau_{t+1} = \rho \cdot \tau_{ij}(t) + (1 - \rho)\gamma \cdot max_{l \in N_j^k(t)} \tau_{jl}$$

où γ est une constante paramètrable (typiquement 0,5). À la fin du cycle on ajoute à τ_{ij} le terme (1 - ρ).$\Delta\tau_{ij}(t)$, où $\Delta\tau_{ij}(t)$ est défini comme pour AS.

Interet

L'idée est de mettre à jour les pistes de phéromones avec une valeur qui est une prévision des quantités de phéromones que la fourmi va trouver dans la ville suivante. Ainsi une fourmi se déplaçant d'un noeud i à j pourra signaler aux suivantes fourmis qui arriveront en i si le déplacement en j est un déplacement justifié .

Cet algorithme présente des bons résultats, mais donne des performances semblables à ACS (voir plus loin), alors que ACS allège le calcul.

5.1.3 Ant Colony System (ACS) [Dorigo 02]

ACS date de 1996/97

On ne parle pour ACS que des principales modifications par rapport à AS

Premiere amélioration

Sur des grands parcours on peut reprocher à AS de ne pas assez se focaliser sur les meilleures solutions trouvées précédemment pour influer sur les décisions des fourmis. Pour remédier à cela ACS fait rentrer en jeu une variable constante paramètrable ψ_0 de valeur situé entre 0 et 1. À chaque fois que l'on doit décider de la prochaine ville où placer la fourmi, on attribue une valeur aléatoire uniformément distribué entre 0 et 1 à la variable ψ. Lorsque $\psi \leq \psi_0$ on effectue un choix uniquement basé sur les villes deja visitées (de manière efficace). Dans le cas où $\psi > \psi_0$ on effectue un choix de la même manière que pour AS.

En ajustant ψ_0 on peut donc pousser les fourmis a se focaliser sur les meilleures solutions trouvées ou bien à les pousser a chercher d'autres chemins.

Seconde amélioration

ACS met en place une politique de mise a jour des phéromones visant a intensifier la solution optimale. Lors de la mise a jour locale (la mise a jour pas à pas effectué sur un arc lorsque une fourmi l'emprunte), ACS a tendance diminuer la teneur en phéromones. A l'inverse de AS, ACS augmente lors de la mise à jour globale (après chaque cycle) uniquement le chemin appartenant au meilleur tour (celui de longueur minimale).

5.1.4 Max-Min Ant System : MMAS [Dorigo 03]

MMAS date de 1997

Différences avec AS

1. Uniquement le parcours le plus court est mis-à-jour en phéromones
2. Les valeurs des phéromones sur chaque arc sont bornées par τ_{min} et τ_{max}
3. Les valeurs des phéromones sur chaque arc sont initialisées a la valeur maximum τ_{max}
4. La quantité de phéromones que l'on fait évaporer est proportionelle à sa valeur au moment de la modification, plus les pistes sont fortes plus ses phéromones seront diminués

Intérêt

- On empêche ainsi la monopolisation de certains arcs qui ont été tellement imprégnés au début du processus de recherche qu'ils sont systématiquement parcourus par les fourmis.
- On permet grâce à cette façon de gérer l'évaporation de tirer vers le bas, les arcs chargés fortement en phéromones afin de vérifier si leur importance est pertinente. De ce fait si ce n'est pas le cas, les pistes plus faiblement chargés pourront leur prendre le pas.

Chapitre 6

Optimisation des tables de routage

6.1 Introduction

La flexibilité indégniable proposé par l'optimisation par colonie de fourmis a permis aux chercheurs d'adapter ces algorithmes a des données dynamiques. A l'heure actuelle, où les réseaux de communication et le nombre d'utilisateurs augmentent exponentiellement, il est nécéssaire de gérer le réseau en temps réel pour éviter les encombrements ou la saturations des lignes.

Le réseau est en grande partie lié à la notion de routage. En effet un réseau informatique peut être vu comme un graphe dont les arètes sont les différentes lignes de communication (qui elles mêmes peuvent être vu comme des sous réseaux plus petits) et les sommets représentent les routeurs. Lorsqu'un routeur reçoit un paquet de la machine A qui doit arriver à la machine B, celui-ci a pour rôle d'envoyer le paquet vers le prochain noeud de son parcours. Les routeurs étant souvent interconnectés plusieurs routes sont possibles pour un même paquet, il doit donc essayer de choisir un trajet minimisant le délai d'acheminement en fonction de l'encombrement du réseau.

On utilise principalement deux grandeurs pour exprimer la qualité d'un réseau :
- la bande passante (bit/s) qui représente le taux de transfert (soit la place sur une ligne).
- le délai moyen (s) qui est le temps d'acheminement moyen d'un paquet.

Un routage de bonne qualité permet d'augmenter la bande passante quand le réseau est chargé et de baisser le délai de transfert d'un paquet quand la charge est faible. La charge du réseau variant continuellement et de façon brutale, les algorithmes d'optimisation par colonie de fourmis semblent parfaitement indiqués pour les gestions des tables de routage. Nous détaillerons dans ce chapitre le fonctionnement de l'algorithme **antNet**.

6.2 Principe de l'algorithme [Dréo 04]

Les développeurs d'antNet on dû faire quelques adaptations aux algorithmes méta-heuristique déjà existants, que se soit la modélisation des fourmis ou le fonctionnement de l'algorithme.

6.2.1 Les fourmis du net

Les fourmis utilisées pour la gestion des tables de routage sont de deux types différents. Ainsi, il existe une fourmi "test" qui va suivre les branches du réseau comme si elle était un paquet et une fourmi "compte-rendu" qui a pour rôle de mettre a jour les tables (fourmis respectivement nommées F-ant et B-ant).

La première fourmi est un peu plus intelligente qu'un insecte lambda car elle a la possibilité de mémoriser l'état du réseau des noeuds qu'elle traverse. En pratique, elle garde en mémoire le temps de parcours et le chemin suivi. Le comportement de cette fourmi est dicté par une loi probabiliste que nous verrons plus loin et elle est également capable d'engendrer une fourmi B-ant à la fin de son parcours avant de mourir.

Les fourmis B-ant quant à elles ont un chemin pré-determiné puisqu'elles parcourent uniquement les noeuds dans le sens inverse de la F-ant l'ayant créée, mettant à jour les tables de routage de ceux-ci. En effet, grâce aux informations de la F-ant, la B-ant est au courant de l'encombrement du réseau permettant ainsi au routeur de s'adapter aux fluctuations.

6.2.2 Mecanisme de antNet

L'algorithme d'antNet est très proche de ceux utilisé pour le TSP, les routeurs pouvant être associés aux villes, ceci dit il y a quelques différences notables :

- il n'y a aucune contrainte sur les villes à visiter, c'est-à-dire qu'une fourmi n'est pas obligé de visiter tous les noeuds.
- il y a une piste de phéromone différente pour chaque noeud de destination.
- la dimension temporelle du problème rend plus délicate son évaluation.

algorithme :

1. Chaque noeud lance une fourmi F-ant avec une destination aléatoire [Dréo 04].
2. Chaque fourmi est routée grace à la fonction stochastique f prenant en paramètre $\tau_{ijd}(t)$, le taux de phéromones sur la piste et $\mu_{ij}(t)$, l'information spécifique au problème (ici μ_{ij} est proportionnel à la liste d'attente entre les noeud i et j).

$$p^k_{ijd}(t) = f(\tau_{ijd}(t), \mu_{ij}(t))$$

3. Chaque fourmi mémorise son parcour et le délai entre chaque noeud. Elle incrémente également le taux de phéromone des pistes traversées.
4. une fois arrivé à destination la F-ant génère une B-ant, celle-ci hérite des informations de la première et emprunte le même chemin en sens inverse.
5. les F-ant sont effacées... On met a jour les $\tau_{ijd}(t)$ pour simuler l'évaporation.
6. la B-ant met à jour les tables de routage.

Bien entendu pour être efficace ce processus doit être lancé périodiquement de façon à pouvoir suivre les fluctuations temporelles de l'encombrement du réseau.

6.3 Comparaison aux algorithmes classiques

Pour juger des performences de cet algorithme, il a été comparé avec les methodes déja connues(SPF,adaptive Bellman-Ford,...).

6.3.1 Expériences

Les expériences ont été réalisé sur simulateur, en utilisant plusieurs topologies et des modèlisations différentes de trafic.
Nous avons décidé de détailler une des experimentations : le trafic est simulé de façon aléatoire en condition de forte charge. A un moment précis, on simule un accroissement soudain menant à la saturation du trafic pendant 120 secondes. Une courbe-étalon nommé Daemon a été faite simulant un algorithme idéal offrant la meilleure performance possible afin d'avoir une borne maximale.

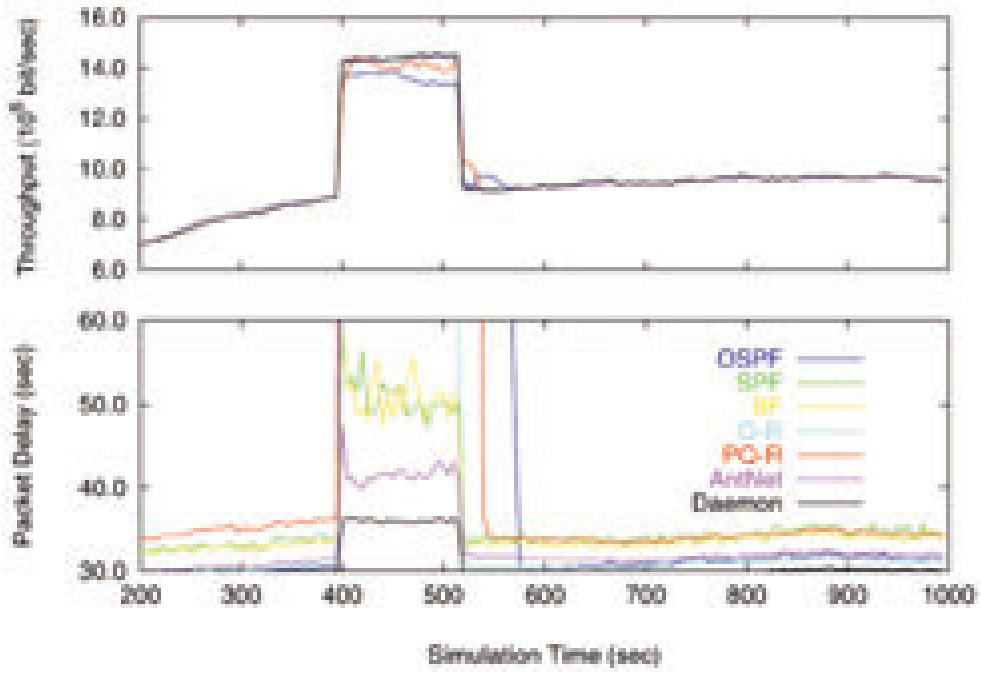


FIG. 6.1 – resultat d'un test de performance

6.3.2 Resultats

Le graphe du haut de la figure 6.1 montre les flux du réseau appliqué a chacun des algorithmes testés, tandis que celui du bas permet de voir les délais moyens des paquets envoyés. On peut constater que antNet (courbe violette) permet de conservé un taux de transfert aussi élevé que les meilleurs algorithmes utilisés aujourd'hui pour le routage et qu'en plus il a un délai moyen bien plus faible.

Sur 23 problèmes différents l'antNet s'est montré 14 fois plus performant que les autres, 6 fois aussi bon et seulement 3 fois en dessous (mais dans des limites raisonnables).

6.4 Conclusion

Cet algorithme a été testé sur l'opérateur téléphonique japonais NTT, ainsi que sur le réseau de la Fondation Nationale Américaine des Sciences : il a montré des résultats plus que prometteurs. Ainsi, un mecanisme biologiquement naturel d'auto-gestion d'agents autonomes permet de gérer des problèmes complexes en temps réel avec beaucoups d'efficacité. Des opérateurs téléphoniques, les acteurs du web et même des sociétés de logistique peuvent être interessés par ces relativement nouveau type d'algorithme.

Table des figures

Bibliographie

[Dorigo 03] *Un chapitre dans son livre qui parle des algorithmes élaborés des colonies de fourmis notamment adaptés pour le TSP et divers problèmes d'optimisation dynamiques.*

[Dorigo 02] *Un chapitre dans son livre est spécifiquement dédié aux algorithmes élémentaires des colonies de fourmis dans un livre généraliste sur les méta heuristiques.*

[Dorigo internet] *On peut trouver sur son site internet, une simulation élaborée du ACOTSP*

[Dréo 04] *Un chapitre dans son livre qui parle de la méta-heuristique ACO, d'une manière simplifiée*

[Dorigo 03 mesures] *Un chapitre dans son livre qui parle de différents tests et mesures du TSP*

[Dréo 04 mesures] *Un chapitre dans son livre qui parle de différentes mesures du TSP d'une manière simplifiée*