Résolution du problème de la patrouille multi-agent en utilisant des colonies compétitives de fourmis

Fabrice Lauri — François Charpillet

LORIA-INRIA Lorraine - Equipe MAIA
Campus Scientifique – B.P. 239 – F-54506 Vandœuvre-Lès-Nancy
{lauri,charp}@loria.fr

RÉSUMÉ. *Patrouiller dans un environnement implique une équipe d'agents dont le but consiste à visiter continuellement et aussi fréquemment que possible les lieux les plus pertinents. Afin d'obtenir des performances optimales, il est alors primordial que les agents coordonnent leurs actions. De nombreux domaines peuvent être concernés par ce problème, comme la robotique, la simulation ou les jeux vidéo. Nous adoptons dans cet article une approche d'optimisation basée sur les colonies de fourmis pour traiter ce problème. Deux algorithmes sont proposés, dans lesquels des colonies de fourmis sont engagées dans une compétition pour découvrir la meilleure stratégie de patrouille multi-agent. Les résultats expérimentaux montrent que, sur quatre des six graphes étudiés, l'une de nos techniques est significativement meilleure que la technique d'apprentissage par renforcement proposée par Santana.*

ABSTRACT. *Patrolling an environment involves a team of agents whose goal usually consists of continuously visiting its most relevant areas as frequently as possible. For such a task, agents have to coordinate their actions in order to achieve optimal performance. A wide range of applications can be dealt with this problem, from computer network management to vehicle routing. The Ant Colony Optimization is adopted here as the solution approach to this problem. Two novel ACO algorithms are proposed here, in which several ants' colonies try to discover the best multi-agent patrolling strategy. Experimental results show that, for four out of the six evaluated graphs, one of our techniques significantly outperforms the reinforcement learning technique proposed by Santana, irrespective of the number of the involved patrolling agents.*

MOTS-CLÉS : *patrouille multi-agent, colonies de fourmis, ACO.*

KEYWORDS: *multi-agent patrolling, ACO.*

1. Introduction

Une patrouille peut être définie comme une mission impliquant une équipe de plusieurs individus, dont le but consiste à visiter continuellement les lieux stratégiques d'un environnement. L'objectif de cette mission concerne généralement le contrôle, la supervision ou la protection d'une région. Une équipe de facteurs effectuant leurs rondes, une esquade de marines sécurisant une zone ou une colonie de fourmis cherchant de la nourriture sont autant d'exemples de patrouilles possibles. Une telle tâche de patrouille nécessite que tous les membres impliqués coordonnent leurs actions, afin d'éviter de visiter plusieurs fois un lieu et permettre ainsi d'atteindre des performances optimales.

Les techniques résolvant le problème de la patrouille multi-agent (ou PPMA) peuvent être utilisées dans de nombreux domaines, comme la gestion de réseaux informatiques (Reuter *et al.*, 2002), le sauvetage par des robots de personnes blessées lors d'une catastrophe naturelle (Kitano, 2000), la détection de menaces ennemies ou la protection de villes dans le domaine des jeux vidéos (Machado *et al.*, 2002a).

Ce problème n'a été traité rigoureusement que très récemment (Machado *et al.*, 2002b ; Almeida *et al.*, 2004 ; Santana *et al.*, 2004 ; Chevaleyre, 2004). Tous ces travaux considèrent la recherche d'une stratégie de parcours multi-agent des nœuds d'un graphe donné, le graphe représentant l'ensemble des lieux de l'environnement à patrouiller.

Dans la plupart des travaux portant sur le domaine de la patrouille, les stratégies de patrouille imaginées et validées utilisent des critères communs d'évaluation proposés dans (Machado *et al.*, 2002b). Les techniques proposées sont basées sur des approches différentes, utilisant soit des lois heuristiques permettant aux agents de choisir le meilleur nœud à visiter prochainement (Machado *et al.*, 2002b), soit des mécanismes de négociation (Almeida *et al.*, 2004), soit des techniques d'apprentissage par renforcement (Santana *et al.*, 2004), soit des techniques inspirées de la théorie des graphes (Chevaleyre, 2004). La majorité des algorithmes proposés fournissent de bons résultats empiriques sur des graphes constitués de moins de cinquante nœuds et d'une centaine d'arcs.

Dans cet article, nous adoptons un algorithme d'optimisation basé sur des colonies de fourmis pour résoudre le problème de la patrouille multi-agent. Cette méthode s'inspire de la capacité des colonies de fourmis naturelles à résoudre collectivement de nombreux problèmes d'optimisation combinatoire (Colorni *et al.*, 1991 ; Dorigo *et al.*, 1996 ; Dorigo *et al.*, 1997). Les deux algorithmes originaux présentés ici (Lauri *et al.*, 2006) étendent le concept habituellement utilisé dans l'algorithme classique selon lequel les fourmis appartiennent à une et une seule colonie. Ici, plusieurs colonies de fourmis seront utilisées : toutes les colonies sont impliquées dans une compétition, chacune d'entre elles essayant de découvrir/construire la meilleure stratégie de patrouille multi-agent. Chaque fourmi appartenant à une même colonie tente de déterminer la meilleure stratégie individuelle qu'un agent patrouilleur devra adopter.

Nous avons choisi d'utiliser une approche à base de colonies de fourmis pour plusieurs raisons. Premièrement, cette classe d'algorithmes est capable de traiter n'importe quelle instance de ce problème, indépendamment de la topologie du graphe ou du nombre d'agents impliqués. Deuxièmement, il est capable de fournir des solutions en particulier lorsque les positions initiales des agents sur les nœuds du graphe sont connues. Dans quelques applications en effet, spécialement celles en rapport avec la robotique – par exemple lorsqu'un groupe de drones doit patrouiller – tous les agents sont placés initialement au même endroit : leurs positions constituent donc une donnée du problème. Sous cette condition, la tâche de patrouille commence toujours par une phase d'exploration collective du graphe, phase pendant laquelle les agents se répartissent les différents nœuds.

La suite de cet article est organisée de la manière suivante. La section 2 décrit le cadre de travail communément adopté et fournit un état de l'art des techniques du domaine de la patrouille multi-agent. L'approche d'apprentissage par renforcement proposée par Santana (c'est-à-dire GBLA (Santana *et al.*, 2004)) est aussi présentée plus en détail dans cette section. GBLA et nos algorithmes d'ACO sont capables de traiter des problèmes où les positions initiales des agents sont connues. Pour cette raison, GBLA servira de technique témoin pour évaluer la performance de nos algorithmes. La section 3 remémore les concepts fondamentaux des algorithmes à base de colonies de fourmis. La section 4 décrit nos deux algorithmes et la section 5 présente les résultats expérimentaux obtenus avec nos techniques basées sur des colonies de fourmis. Finalement, la conclusion et les perspectives de recherche sont donnés dans la section 6.

2. Le problème de la patrouille multi-agent

2.1. *Cadre de travail*

Le problème de la patrouille multi-agent est habituellement formulé comme suit (Machado *et al.*, 2002b ; Chevaleyre, 2004 ; Santana *et al.*, 2004). L'environnement à patrouiller est réduit à un graphe $G = (V, E)$, V représentant les zones stratégiquement pertinentes et E les moyens de transport ou de communication entre eux. Un coût c_{ij}, associé à chaque arc (i, j), mesure le temps nécessaire pour aller d'un nœud i à un nœud j.

Soient r agents destinés à visiter à intervalles réguliers les zones définies dans le graphe G. Chaque agent se trouve sur un des nœuds de V à l'instant initial. Résoudre le problème de la patrouille consiste alors à élaborer une stratégie σ de parcours multi-agent du graphe G. Une telle stratégie doit optimiser un critère de qualité donné. $\sigma = \{\sigma_1 \cdots \sigma_r\}$ est constitué des r stratégies individuelles σ_i de chaque agent i. Une stratégie individuelle σ_i est définie telle que $\sigma_i : \mathbb{N} \rightarrow V$, $\sigma_i(j)$ représentant le j-ème nœud visité par l'agent i, avec $\sigma_i(j+1) = x$ si $(\sigma_i(j), x) \in E$.

Il est communément admis qu'une stratégie de patrouille efficace est une stratégie qui minimise pour chaque nœud le délai entre deux visites.

Plusieurs critères ont été imaginés dans (Machado *et al.*, 2002b) afin d'évaluer la qualité d'une stratégie de patrouille multi-agent après T pas de temps (ou cycles) de simulation. Tous sont basés sur le concept d'*oisiveté instantanée d'un nœud* (OIN). L'OIN $I_n(t)$ d'un nœud n à l'instant t est le nombre de pas de temps pendant lequel ce nœud est resté non visité. Par convention, à l'instant initial, $I_n(0) = 0$, $\forall n = 1, 2, \cdots, |\mathcal{V}|$.

A un instant t donné, AI_t est alors l'*oisiveté moyenne du graphe* (OMG), c'est-à-dire :

$$AI_t = \frac{1}{|\mathcal{V}|} \sum_{n \in V} I_n(t)$$

De manière similaire, la *pire oisiveté* WI_t est la plus grande oisiveté OIN rencontrée pendant les t pas de temps de simulation, c'est-à-dire :

$$WI_t = \max_{s=1,2,\ldots,t} \max_{n \in V} I_n(s)$$

Une stratégie de patrouille multi-agent σ peut être évaluée après T cycles de simulation en utilisant soit le critère de l'*oisiveté moyenne* AI_σ, soit le critère de la *pire oisiveté* WI_σ. L'oisiveté moyenne dénote la moyenne des OMG sur les T cycles de simulation, c'est-à-dire :

$$AI = \frac{\sum_{t=1}^{T} AI_t}{T}$$

tandis que la pire oisiveté est la plus grande OIN observée pendant les T pas de temps de simulation, c'est-à-dire :

$$WI = \max_{t=1,2,\ldots,T} \max_{n \in V} I_n(t)$$

Comme souligné dans (Chevaleyre, 2005), utiliser le critère de la pire oisiveté assure de toujours trouver une stratégie de patrouille optimale σ^*, puisque si σ^* minimise WI_{σ^*}, elle minimise aussi AI_{σ^*}. Le contraire n'est pas vrai.

2.2. *Etat de l'art*

A notre connaissance, (Machado *et al.*, 2002b ; Machado *et al.*, 2002a) sont les travaux pionniers traitant du problème de la patrouille multi-agent. Dans ces articles, les auteurs considèrent respectivement des graphes unitaires et des graphes avec des distances réelles. Plusieurs architectures multi-agents sont proposées, ainsi que les critères décrits dans la section précédente. Chaque architecture est une combinaison de quelques paramètres spécifiques, tels que le type de communication entre agents (permise ou interdite), la perception des agents (locale ou globale), la fonction heuristique de sélection du nœud suivant (aléatoire, en utilisant une oisiveté individuelle ou partagée, en fonction de la longueur du chemin ...), etc.

(Almeida *et al.*, 2004) améliora les meilleures architectures proposées dans (Machado *et al.*, 2002a) en concevant des agents capables d'échanger librement des messages et de conduire des négociations au sujet des nœuds qu'ils doivent visiter. Chaque agent reçoit au départ un ensemble de nœuds aléatoires à visiter, et utilise un système d'enchères pour échanger avec les autres agents les nœuds qu'il considère comme indésirables. Chaque agent tente alors de conserver les nœuds qu'il peut visiter en un temps raisonnable donné. La capacité de négocier permet donc aux agents d'atteindre une entente mutuelle.

Chevaleyre (Chevaleyre, 2005 ; Chevaleyre, 2004) reformula le problème de la patrouille dans les termes d'un problème d'optimisation combinatoire. Il prouva tout d'abord qu'une stratégie de patrouille impliquant un seul agent pouvait être déterminée à l'aide d'un algorithme permettant de résoudre une variante du problème du voyageur de commerce [1]. Il étudia ensuite trois stratégies de patrouille multi-agent possibles, en introduisant des biais, et montra que toutes permettent d'obtenir des stratégies proches de l'optimale.

Dans (Sempe, 2004), l'auteur propose d'utiliser des algorithmes à base de fonctions de potentiels pour permettre à une équipe de robots de patrouiller en présence de perturbations, perturbations pouvant être dues à des problèmes de localisation ou de batteries déchargées. Les modèles CLInG, proposé par Sempé, s'appuient sur la diffusion de l'oisiveté d'un nœud autour de celui-ci. Chaque robot patrouilleur perçoit et suit le gradient de l'oisiveté la plus forte. Cette stigmergie permet alors à l'équipe de robots de se déployer efficacement dans l'environnement, de manière à privilégier les lieux jugés prioritaires. Le principe de stigmergie, employé dans le modèle CLInG ainsi que dans tout algorithme à base de colonies de fourmis, permet à des entités autonomes de communiquer via l'environnement. Sans cette communication indirecte, ces entités seraient incapables de réaliser ensemble des tâches complexes.

Dans (Santana *et al.*, 2004), les agents sont capables d'apprendre à patrouiller en utilisant le cadre de travail de l'apprentissage par renforcement. Chaque agent implémente un processus décisionnel de Markov qui est employé pour savoir quelle action entreprendre dans n'importe quel état de l'environnement graphique. Une action permet à un agent de se rendre sur le nœud adjacent à celui sur lequel il se trouve actuellement. Un état de l'environnement représente l'information minimale nécessaire à un agent pour décider aussi précisément que possible quoi faire. Deux architectures furent proposées : une dans laquelle les agents ne peuvent pas communiquer, une autre dans laquelle ils peuvent communiquer indirectement (c'est-à-dire via l'environement) leur intention sur la prochaine action. Dans la section dédiée aux résultats, nous comparerons notre approche à cette deuxième architecture (appelée GBLA), qui se révéla être la meilleure des deux dans (Santana *et al.*, 2004).

Toutes les appproches décrites précédemment, à l'exception de celles proposées par Sempé, ont été évaluées dans (Almeida *et al.*, 2004) et comparées sur 12 configurations, c'est-à-dire pour six topologies de graphes avec 5 et 15 agents. Il a été montré

1. la variante *graphical-TSP*, pour laquelle le graphe n'est pas nécessairement complet.

que la stratégie à cycle unique (Chevaleyre, 2004) donne les meilleurs résultats sur toutes les configurations sauf une, tandis que les deux meilleures architectures proposées dans (Machado *et al.*, 2002a ; Machado *et al.*, 2002b) produisirent les moins bons résultats. Toutes les autres techniques présentées dans (Almeida *et al.*, 2004) et (Santana *et al.*, 2004) fournirent des performances équivalentes.

Dans ce présent article, nous avons choisi de comparer nos techniques basées sur des colonies de fourmis avec la méthode d'apprentissage par renforcement GBLA proposée par Santana (Santana *et al.*, 2004). Celle-ci est brièvement décrite ci-après.

2.3. *L'approche Gray-Box Learner Agent*

Chaque agent dans GBLA emploie une *politique*, calculée à partir d'un Processus Décisionnel de Markov (ou PDM), pour déterminer quelle est l'action à exécuter dans n'importe laquelle des situations rencontrées.

Rappelons qu'un PDM peut être défini comme un quadruplet $< S, A, T, R >$ où :

– S est l'ensemble fini des états de l'environnement ;

– $A = \bigcup_{s \in S} A(s)$ est l'ensemble fini des actions permettant à un agent d'influencer l'état de l'environnement, $A(s)$ étant le sous-ensemble des actions disponibles dans l'état s ;

– $T : S \times A \rightarrow S$ représente la dynamique de l'environnement, où $T(s, a)$ indique l'état qui est toujours atteint lorsque l'action a est exécutée dans l'état s ;

– enfin, $R : S \times A \rightarrow \mathbb{R}$ est la fonction de récompense.

Les fonctions T et R, représentant le modèle de l'environnement, doivent satisfaire la propriété de Markov.

Pour un PDM donné, une politique π représente alors une fonction $\pi : S \rightarrow A$, où $\pi(s)$ est l'action que l'agent exécute dans l'état s. Lorsque le modèle $< T, R >$ de l'environnement n'est pas disponible, résoudre un PDM consiste à trouver la politique optimale π^* telle que $Q^{\pi^*}(s, a) = max_\pi \; Q^\pi(s, a) \; \forall s \in S$ et $\forall a \in A(s)$. $Q^\pi(s, a)$ est la *fonction action-valeur* qui représente la récompense attendue lorsque l'agent se trouve dans l'état s, en exécutant l'action a puis en suivant la politique π. Si l'on considère un PDM à horizon infini, $Q^\pi(s, a)$ est généralement définie comme une somme pondérée des récompenses obtenues :

$$Q^\pi(s, a) = E_\pi \left\{ \sum_{k=0}^{+\infty} \gamma^k \, r_{t+k+1} \middle| \, s_t = s, a_t = a \right\}$$

γ étant le *facteur de pondération* tel que $0 \leq \gamma < 1$.

Ainsi, une politique optimale π^* peut être construite selon une stratégie gloutonne, en considérant que $\pi^*(s) = \arg\max_{a \in A(s)} \; Q^{\pi^*}(s, a)$. Nous invitons le lecteur à consulter l'ouvrage de référence (Sutton *et al.*, 1998) pour plus de détails sur les concepts et les problématiques de l'apprentissage par renforcement.

Dans GBLA, le PDM de chaque agent est appris à l'aide du *Q-Learning*, qui est l'algorithme d'apprentissage par renforcement le plus utilisé lorsqu'il s'agit de résoudre un PDM dont le modèle de l'environnement n'est pas disponible. Le *Q-Learning* consiste à mettre à jour les valeurs $Q(s,a)$ d'une paire état-action à chaque pas de temps t, en utilisant l'équation suivante :

$$Q_t(s,a) \leftarrow Q_{t-1}(s,a) +$$

$$\alpha\,[R(s,a) + \gamma\, max_{a'}\, Q_t(s',a') \;-\; Q_{t-1}(s,a)]$$

s' étant l'état atteint en exécutant l'action a dans l'état s et α représentant le *taux d'apprentissage*. Il a été montré que cet algorithme converge vers la fonction état-valeur optimale Q^* sous la condition que chaque état est visité un nombre infini de fois.

En considérant que d représente le degré du graphe de patrouille G et que $|\mathcal{V}|$ est le nombre de nœuds du graphe, l'espace d'états S dans GBLA est construit à partir des informations suivantes :

1) le nœud sur lequel l'agent se trouve ($|\mathcal{V}|$ valeurs possibles)

2) l'arc récemment traversé par l'agent (d valeurs possibles)

3) le nœud voisin qui a la plus grande (pire) oisiveté (d valeurs possibles)

4) le nœud voisin qui a la plus petite oisiveté (d valeurs possibles)

5) la liste des nœuds adjacents qui sont susceptibles d'être visités prochainement par les autres agents (2^d valeurs possibles).

La cardinalité de l'espace d'actions est quant à elle égale au degré d du graphe : chaque action permet à un agent d'atteindre un des nœuds adjacents.

3. Optimisation par colonie de fourmis

Avant de décrire dans la section 4 notre algorithme basé sur des colonies de fourmis, nous exposons tout d'abord les concepts généraux de ce type d'algorithmes, ainsi qu'un état de l'art des problèmes voisins qu'ils ont permis de résoudre efficacement.

3.1. *Fourmis naturelles*

Les fourmis sont des insectes relativement simples. Elles sont presque totalement aveugles, ont une capacité mnésique très limitée et semblent se mouvoir de manière aléatoire (Colorni *et al.*, 1991). Malgré leurs faibles caractéristiques individuelles, les fourmis constituent l'une des seules espèces qui peut être trouvée n'importe où dans le monde, dans n'importe quel type d'environnement, ce qui témoigne de leur remarquable faculté d'adaptation. Cette faculté d'adaptation leur permet de coopérer afin de réaliser des tâches complexes, telles que découvrir des lieux pourvus en nourriture, déterminer les plus courts chemins vers ces lieux, construire et protéger leur nid ou

former des ponts (Schoonderwoerd *et al.*, 1997). Pour accomplir ces tâches, les fourmis communiquent indirectement via des stimuli environnementaux. Cette forme de communication est plus connue sous le nom de *stigmergie*. Les fourmis déposent sur le sol une hormone volatile (la *phéromone*) pour prévenir les autres que cette position devrait être atteinte une seconde fois. La phéromone constitue ainsi un système signalétique, agissant comme un moyen de communication et menant à des comportements coopératifs comme le suivi de trace (Schoonderwoerd *et al.*, 1997).

3.2. *Fourmis artificielles*

Le remarquable pouvoir de collaboration dont font preuve les fourmis a été utilisé par Dorigo pour définir une métaheuristique[2] : l'*Ant Colony Optimization (ou ACO)*. Cette métaheuristique permet de résoudre les problèmes d'optimisation combinatoire les plus difficiles (Colorni *et al.*, 1991 ; Dorigo *et al.*, 1996). D'après leurs auteurs, la métaheuristique ACO possèdent plusieurs qualités. Elle est versatile, dans le sens où un algorithme conçu à partir d'ACO peut être utilisé tel quel pour des variantes différentes d'un problème donné, ou avec des changements minimaux pour traiter des problèmes d'optimisation combinatoire différents. ACO engendre des algorithmes robustes, car les fourmis artificielles sont capables de poursuivre leurs activités même si certaines d'entre elles deviennent défaillantes ou périssent. ACO se base sur une population de solutions, ce qui permet d'avoir recours au principe de rétroaction positive, un mécanisme efficace permettant de trouver rapidement des solutions acceptables. Enfin, ACO est flexible, dans le sens où la colonie de fourmis est capable d'adapter son comportement face aux changements de l'environnement.

Pour que ACO puisse être appliquée efficacement à un problème donné (Colorni *et al.*, 1992 ; Dorigo *et al.*, 1996), celui-ci doit être formulé dans les termes d'un parcours des nœuds d'un graphe par des agents. Sous cette condition, les fourmis artificielles peuvent construire une solution au problème posé, en échangeant des informations par le biais de phéromones déposées sur les arcs du graphe. En traversant les arcs, les fourmis modifient la représentation du problème en ajoutant des informations dans la structure même du graphe (Dorigo *et al.*, 1997).

Puisque ACO n'est pas supposée être utilisée pour simuler de la manière la plus réaliste possible des fourmis dans la nature, mais bien comme un outil d'optimisation combinatoire, les fourmis artificielles qu'elle utilise possèdent des différences notoires par rapport aux fourmis réelles. En effet, les fourmis artificielles dans ACO peuvent mémoriser les actions qu'elles ont réalisées et les lieux qu'elles ont visités. Par ailleurs, elles peuvent évaluer la distance entre deux lieux, percevoir plusieurs niveaux de phéromone, et leur environnement évolue en temps discret.

2. Nous entendons par *métaheuristique* une approche générale permettant de définir une certaine classe d'algorithmes.

3.3. *Etat de l'art de l'ACO sur des problèmes apparentés au PPMA*

Depuis ses premiers développements, ACO a été appliquée à un large spectre de problèmes. Nous exposons ici uniquement l'état de l'art de ACO pour les domaines proches du problème de la patrouille multi-agent. Sont ainsi concernés le problème du voyageur de commerce[3] (Colorni *et al.*, 1991 ; Colorni *et al.*, 1992 ; Dorigo *et al.*, 1996 ; Dorigo *et al.*, 1997), la couverture distribuée de régions par des robots-fourmis[4] (Wagner *et al.*, 1998 ; Wagner *et al.*, 1999 ; Yanowski *et al.*, 2003) et le problème de détermination des itinéraires d'un ensemble de véhicules[5] (Bell *et al.*, 2004 ; Gambardella *et al.*, 1999).

Deux des plus récents travaux sur ACO proviennent de Colorni (Colorni *et al.*, 1991 ; Colorni *et al.*, 1992). Leur premier article (Colorni *et al.*, 1991) introduit la métaheuristique ACO et décrit également trois algorithmes permettant de résoudre le TSP : *ant-density*, *ant-quantity* et *ant-cycle*. L'algorithme le plus efficace, *ant-cycle*, permet de trouver de très bonnes solutions pour des problèmes ayant recours à des graphes de 50 à 75 villes. Leur deuxième article (Colorni *et al.*, 1992) relate des investigations menées pour déterminer le paramétrage adéquate de leur algorithme *ant-cycle*, sa complexité et les résultats empiriques obtenus par rapport à d'autres techniques résolvant le TSP.

Dans (Dorigo *et al.*, 1996), les mêmes auteurs comparent *ant-cycle* à d'autres métaheuristiques (telles que la recherche tabou ou le recuit simulé) sur le problème du voyageur de commerce. Ce même article présente également les résultats obtenus par ACO sur le problème du voyageur de commerce asymétrique. Dans (Dorigo *et al.*, 1997), un algorithme étendant les principes de l'algorithme *ant-cycle* est proposé, appelé *ant colony system*, pour résoudre à la fois le TSP et l'ATSP. En incluant des heuristiques spécifiques au problème (comme 2-opt), les auteurs ont rapporté les meilleurs résultats jamais obtenus pour de grandes instances de l'ATSP.

Wagner (Wagner *et al.*, 1998 ; Wagner *et al.*, 1999 ; Yanowski *et al.*, 2003) considèrent le problème de la couverture de zones par des robots, en supposant qu'il existe un graphe qui décrit les régions de l'environnement devant être visitées aussi rapidement que possible par un groupe de robots. Couvrir un graphe se réduit alors à visiter tous ses arcs ou tous ses nœuds. Pour atteindre cette tâche, les auteurs définissèrent plusieurs stratégies, en terme de lois de comportement local devant être exhibées par chacun des robots. Une stratégie est considérée comme meilleure qu'une autre si elle permet de couvrir plus rapidement une région. L'efficacité d'une stratégie est évaluée en mesurant le temps entre l'instant où le premier nœud est visité et l'instant où le dernier nœud non encore visité est atteint. Les robots ne disposent que d'une quantité très limitée de mémoire (uniquement pour leur permettre de parcourir en sens inverse un chemin) et sont capables de déposer ou de percevoir sur le sol des traces. Dans

3. ou TSP pour *Traveling Salesman Problem*
4. ou *distributed covering by ants-robots*
5. ou *vehicle routing problem* (VRP)

(Wagner *et al.*, 1998), un algorithme nommé *Vertex Ant Walk (VAW)* est présenté, et il fût montré qu'il est capable de couvrir les nœuds d'un graphe uniquement en percevant les traces déposées sur les nœuds adjacents. Dans (Wagner *et al.*, 1999), les auteurs décrivent l'algorithme *Edge Ant Walk*, qui permet également de couvrir un graphe, à la différence que les agents doivent percevoir les traces des arcs partant du nœud courant. EAW est utilisé dans (Yanowski *et al.*, 2003) pour patrouiller les nœuds d'un graphe. Le critère du temps couvrant (*blanket time*), défini comme étant l'instant où le ratio entre le nombre de traversées de n'importe quelle paire d'arcs n'excède pas 2, fût utilisé pour évaluer les performances de la stratégie EAW. Il fût alors prouvé que le temps couvrant reste inférieur à une certaine valeur.

Nous tenons à préciser que le critère d'évaluation d'une stratégie de patrouille, ainsi que les hypothèses émises sur les caractéristiques des robots et les stratégies de patrouille dans les travaux de Wagner diffèrent ostensiblement de ceux décrits dans la section 2 et sur lesquels sont basés nos algorithmes de patrouille.

Enfin, de nombreux algorithmes utilisant plusieurs colonies de fourmis ont été utilisés (Gambardella *et al.*, 1999 ; Bell *et al.*, 2004 ; Kawamura *et al.*, 2000), dont (Gambardella *et al.*, 1999) et (Bell *et al.*, 2004) pour traiter le VRP. Rappelons que le problème du VRP consiste à minimiser le cumul des distances des itinéraires d'un ensemble de m véhicules devant fournir des marchandises à un ensemble de n clients, en sachant que chaque client doit être visité une seule fois, que chaque véhicule commence et termine sa tournée au dépot et que la capacité de chaque véhicule est limitée.

Dans (Gambardella *et al.*, 1999), une nouvelle méthodologie mettant en jeu plusieurs colonies de fourmis est proposée pour optimiser de multiples fonctions de coût. Les auteurs appliquent cette méthodologie pour résoudre le VRP en minimisant le nombre de véhicules fournisseurs et la distance cumulée des itinéraires des véhicules. Pour cela, deux colonies de fourmis sont utilisées et collaborent en échangeant des informations. Chaque colonie optimise une des fonctions de coût et utilise un type différent de phéromones. Dans (Bell *et al.*, 2004), les auteurs proposent un algorithme permettant de résoudre le VRP et utilisant plusieurs colonies indépendantes de fourmis. Chaque colonie utilise un type unique de phéromones et permet de déterminer l'itinéraire optimal d'un seul véhicule. Selon l'avis des auteurs, la séparation des colonies permet d'accroître la probabilité qu'un véhicule continue d'emprunter les chemins les plus prometteurs tout en évitant d'être distraits par les chemins de phéromones des autres fourmis assignées aux autres véhicules. Enfin, il est proposé dans (Kawamura *et al.*, 2000) une extension de l'algorithme ACO utilisant plusieurs colonies de fourmis et des interactions inter-colonies. Dans cette nouvelle approche, le comportement des fourmis d'une colonie peut être influencé positivement ou négativement par les phéromones d'autres colonies selon les interactions définies par le concepteur de l'algorithme.

Par rapport à ces travaux, nos deux approches utilisent également plusieurs colonies, mais ces colonies sont mises en compétition afin de mettre en évidence la meilleure solution plus rapidement que si une seule colonie était utilisée et ainsi accélérer la recherche dans l'espace des solutions. Dans l'une de nos méthodes (GU/AA),

les colonies échangent des informations via le dépôt de phéromones : la meilleure stratégie de patrouille multi-agent est construite par combinaison des meilleures stratégies individuelles déterminées au sein de l'ensemble des colonies. Dans l'autre de nos méthodes (GU/AWA), les colonies sont indépendantes et n'échangent aucune information : la meilleure stratégie multi-agent émerge alors parmi les autres stratégies générées en parallèle.

3.4. *ACO pour résoudre le TSP*

Puisque notre algorithme est largement inspiré de l'algorithme *ant-cycle* de Colorni, nous en fournissons ici une brève description.

Dans cet algorithme, les fourmis sont localisées sur des nœuds (villes) différents [6]. A chaque pas de temps, chaque fourmi choisit avec une certaine probabilité le prochain nœud qu'elle va visiter. Cette probabilité dépend de la quantité de phéromones et de la longueur de l'arc que la fourmi va emprunter, en sachant que les arcs possédant une grande concentration de phéromones et un poids (longueur) faible seront préférentiellement empruntés.

Chaque fourmi visite uniquement les nœuds qu'elle n'a pas encore visités (à l'aide d'une liste *tabu*), jusqu'à ce qu'elle ait effectué un circuit complet de tous les nœuds. Une fois que toutes les fourmis ont terminé leur tour, chaque fourmi dépose une certaine quantité de phéromone sur tous les arcs qu'elle a empruntés. La quantité de phéromones déposée est d'autant plus importante que la longueur de l'arc est petite et cette quantité s'évapore au fil du temps sur chaque arc. Un autre cycle de parcours des nœuds du graphe est répété, chaque fourmi recommençant son parcours à partir de son nœud initial. Le meilleur tour trouvé à l'issue d'un certain nombre de cycles constitue alors la solution finale.

4. Utilisation de ACO pour résoudre le PPMA

L'application de ACO pour résoudre le problème de la patrouille multi-agent paraît évidente. D'une part, un graphe constitue l'une des données de ce problème. D'autre part il requiert des comportements de coordination et de suivi de chemins, qui sont des tâches auxquelles les fourmis excellent.

Nous avons développé deux algorithmes basées sur ACO : GU/AA et GU/AWA. Tous deux mettent à jour la quantité de phéromones sur les arcs du graphe de manière globale, c'est-à-dire après que les fourmis aient trouvé une solution. Avant de décrire ces algorithmes plus en détails, nous donnons tout d'abord leur structure générale commune.

6. Les auteurs ont montré que les meilleures performances étaient obtenues lorsque le nombre de fourmis était égal au nombre de nœuds du graphe et lorsque celles-ci étaient placées sur des nœuds différents.

4.1. *Algorithme général*

Comme stipulé dans (Chevaleyre, 2004), chaque instance du problème de la patrouille impliquant un seul agent peut être réduit à une instance du *graphical*-TSP (GTSP) (Reinelt, 1994). Rappelons qu'un voyageur de commerce dans GTSP doit trouver un circuit dans un graphe connecté G, qui n'est pas nécessairement complet. Il est permis au voyageur de visiter une ville ou de traverser un arc plus d'une fois. Le circuit de longueur minimale passant par tous les nœuds de G constitue alors la meilleure stratégie possible de patrouille d'un agent.

Il a été montré dans (Reinelt, 1994) qu'une instance du GTSP peut facilement être transformée en une instance du TSP équivalent. Soit $G_n = (V_n, E_n)$ le graphe complet de n nœuds. Pour chaque paire de nœuds (i, j), le plus court chemin du nœud i au nœud j dans le graphe G est calculé ; d_{ij} représente la longueur de ce chemin entre i et j. Le poids de chaque arc ij dans G_n est alors égal à la longueur d_{ij} de ce chemin (voir figure 1 ci-dessous).

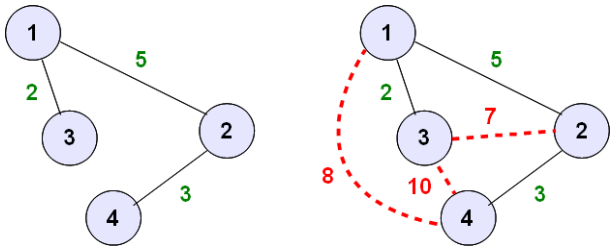


Figure 1. *Transformation d'un graphe de GTSP vers un graphe complet de TSP*

Une fois qu'une solution au TSP est trouvée, le plus court circuit dans G_n peut finalement être transformé en un plus court circuit visitant toutes les villes dans G.

Une méthode pratique pour résoudre le problème de la patrouille avec un seul agent (c'est-à-dire le GTSP) consiste à utiliser l'algorithme *ant-cycle* de Colorni pour résoudre le TSP correspondant et à utiliser A^* (Hart *et al.*, 1968 ; Hart *et al.*, 1972) pour calculer les chemins entre les paires de nœuds de G.

A partir de cet algorithme simple qui permet de traiter efficacement le problème de la patrouille avec un agent, nous avons élaboré une généralisation au problème de la patrouille multi-agent.

Soulignons tout d'abord que résoudre le TSP avec une approche ACO consiste à mettre m fourmis en compétition. Chaque fourmi travaille pour un voyageur de commerce qui désire trouver le plus court circuit visitant une et une seule fois toutes les villes (nœuds) d'un graphe.

Dans le problème de la patrouille multi-agent, chacun des r agents i désire trouver une stratégie individuelle π_i (c'est-à-dire la liste des nœuds qu'il doit visiter) telle que la stratégie de patrouille multi-agent $\pi = \{\pi_1, \pi_2, \ldots, \pi_r\}$ optimise un certain critère de qualité. En utilisant la métaphore précédente des fourmis travaillant pour un voya-

geur de commerce, nous aurons alors dans notre cas m fourmis, chacune appartenant à des colonies différentes, qui travailleront pour un seul agent (figure 2).

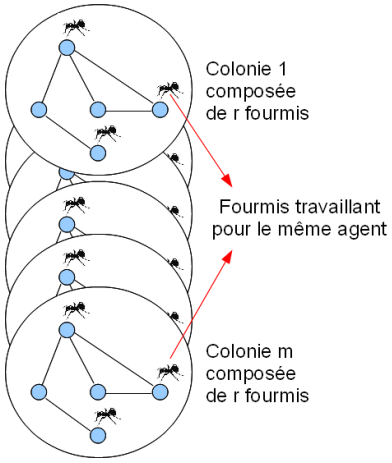


Figure 2. *Compétition entre m colonies composées chacune de r fourmis : chaque fourmi i d'une colonie détermine la stratégie individuelle de patrouille de l'agent i et chaque colonie génère une stratégie de patrouille multi-agent*

Soient $\{a_{i,1}, a_{i,2}, \ldots, a_{i,m}\}$ l'ensemble des m fourmis travaillant pour l'agent i, $\forall i \in [1; r]$. Puisqu'une bonne stratégie de patrouille nécessite de la part des membres impliqués une coordination de leurs actions et un partage des nœuds à visiter, il en sera de même pour les fourmis travaillant pour eux. Le fait de les organiser en colonies permettra de les rendre collaboratives. Les fourmis $a_{1,j}, a_{2,j}, \ldots, a_{r,j}$ seront ainsi regroupées ensemble dans la colonie j.

Les fourmis appartenant à la même colonie auront pour objectif de déterminer la meilleure stratégie qu'un agent patrouilleur devra adopter, tandis que les m colonies de fourmis seront engagées dans une compétition, chacune d'entre elles essayant de découvrir la meilleure stratégie de patrouille multi-agent.

Au sein d'une colonie, une fourmi se déplace d'un nœud visité vers un nœud non encore visité. Chaque fourmi k d'une colonie l mémorise la liste des nœuds qu'elle a déjà visités dans la liste $tabu_{k,l}$ et elle connaît les nœuds qui ont été visités par les autres fourmis de la même colonie, c'est-à-dire qu'elle peut avoir accès aux listes $tabu_{x,l}, x \neq k$. Une fourmi sélectionne le prochain nœud à visiter en utilisant la probabilité $p_{ij}^{k,l}$ de traverser un arc (i, j). Plus $p_{ij}^{k,l}$ est grande, plus le nœud j est susceptible d'être visité. La définition de cette probabilité dépend de la variante de l'algorithme et sera donné dans la section correspondante (cf. sections 4.2 et 4.3).

En ce qui concerne la règle de mise à jour de la quantité de phéromones sur les arcs, nous avons adopté celle de Colorni (Colorni *et al.*, 1991 ; Colorni *et al.*, 1992 ; Dorigo *et al.*, 1996) pour leur algorithme *ant-cycle*. Ainsi à chaque cycle de l'algorithme, l'intensité des phéromones diminue d'une certaine valeur sur chaque arc du graphe

(les phéromones s'évaporent) et chaque fourmi dépose sur les arcs qu'elle a traversé durant le circuit trouvé une quantité de phéromone dépendant de la longueur du circuit.

Tandis que dans la variante GU/AA, les phéromones sont déposées sur les arcs d'un unique graphe (le graphe de patrouille), nous considérons dans la variante GU/AWA que chacun des r agents possède une copie du graphe de patrouille. Les m fourmis travaillant pour lui peuvent alors y déposer leurs phéromones. Les sections suivantes 4.2 et 4.3 décrivent plus en détail la mise à jour des phéromones pour GU/AA et GU/AWA, respectivement.

4.2. *Global Update with all ants (GU/AA)*

Dans GU/AA, la probabilité de sélectionner le prochain nœud à visiter par la fourmi k appartenant à la colonie l est donnée par l'équation suivante :

$$p_{ij}^{k,l} = \begin{cases} \frac{[\tau_{ij}(T)]^{\alpha}\ [\eta_{ij}]^{\beta}}{\sum_{u \in allowed_l} [\tau_{iu}(T)]^{\alpha}\ [\eta_{iu}]^{\beta}} & \text{si } j \in allowed_l \\ 0 & \text{sinon} \end{cases} \quad [1]$$

où $allowed_l = \{V - \sum_{i=1}^{r} tabu_{i,l}\}$ représente l'ensemble des nœuds non encore visitées par les fourmis de la colonie l, V est l'ensemble des nœuds du graphe, $\tau_{ij}(T)$ est l'intensité des phéromones sur l'arc (i, j) au cycle T, $\eta_{ij} = 1/c_{ij}$ est la visibilité du nœud j et α et β sont les paramètres qui contrôlent l'importance relative respectivement de l'intensité des phéromones et de la visibilité.

L'intensité des phéromones est mise à jour comme suit :

$$\tau_{ij}(T+1) = \rho\ \tau_{ij}(T) + \Delta\tau_{ij} \quad [2]$$

ρ étant le coefficient d'évaporation, $\tau_{ij}(T+1)$ et $\tau_{ij}(T)$ étant les intensités des phéromones sur l'arc (i, j) au cycle $T+1$ et au cycle T, respectivement. $\Delta\tau_{ij}$ représente la quantité de phéromones déposée sur l'arc (i, j) par toutes les fourmis de toutes les colonies au cours de ce cycle.

4.3. *Global Update with Agents' Working Ants (GU/AWA)*

Dans GU/AWA, nous considérons que chaque agent possède une copie du graphe de patrouille, sur les arcs duquel les m fourmis travaillant pour lui peuvent y déposer des phéromones. La probabilité de sélectionner le prochain nœud à visiter par la fourmi k appartenant à la colonie l est alors donnée par :

$$p_{ij}^{k,l} = \begin{cases} \frac{[\tau_{ij}^{k}(T)]^{\alpha}\ [\eta_{ij}]^{\beta}}{\sum_{u \in allowed_l} [\tau_{iu}^{k}(T)]^{\alpha}\ [\eta_{iu}]^{\beta}} & \text{si } j \in allowed_l \\ 0 & \text{sinon} \end{cases} \quad [3]$$

où $\tau_{ij}^{k}(T)$ est l'intensité des phéromones sur l'arc (i, j) au cycle T pour le graphe de l'agent k.

L'intensité des phéromones est mise à jour ainsi :

$$\tau_{ij}^{k}(T+1) = \rho\ \tau_{ij}^{k}(T) + \Delta\tau_{ij}^{k} \quad [4]$$

où $\Delta\tau_{ij}^{k}$ est la quantité de phéromones déposée sur l'arc (i,j) par les m fourmis travaillant pour l'agent k. Cette variante suggère que les fourmis travaillant pour des agents différents n'interagissent pas entre elles et que les fourmis travaillant pour un agent donné ne sont sensibles qu'à un seul type de phéromones.

4.4. *Description formelle des algorithmes*

Plus formellement, les algorithmes GU/AA et GU/AWA effectuent les opérations suivantes :

1) Initialiser $T = 0$ (T est le compteur de cycles)
   Pour chaque arc (i,j) de G Faire
   Initialiser la quantité de phéromones $\tau_{ij}(T) = c$ (ou $\tau_{ij}^{k}(T) = c, \forall k$)
   Initialiser $\Delta\tau_{ij} = 0$ (ou $\Delta\tau_{ij}^{k} = 0, \forall k$).
   FinPour
   Placer les r fourmis de chaque colonie sur le nœud initial des agents correspondant.

2) Pour chaque colonie l Faire
   Pour chaque fourmi k Faire
   Vider la liste $tabu_{k,l}$ et placer le nœud initial de la fourmi k dans $tabu_{k,l}(0)$.
   FinPour
   FinPour

3) Pour chaque colonie l Faire
   Initialiser $s = 0$
   Tant que $\cup_{i\in[1;r]}\, tabu_{i,l}$ est pleine Faire
   $s = s + 1$
   Pour chaque fourmi k Faire
   Choisir le nœud j vers lequel aller, avec la probabilité $p_{ij}^{k,l}$ donné par l'équation (1) ou (3)
   Déplacer la fourmi k sur le nœud j et insérer le nœud j dans $tabu_{k,l}(s)$
   FinPour
   FinPour

4) Pour chaque colonie l Faire
   Pour chaque fourmi k Faire
   Placer la fourmi k sur le nœud indiqué dans la liste $tabu_{k,l}(0)$
   Calculer la longueur $L_{k,l}$ du tour découvert par la fourmi k, en utilisant la distance des chemins calculés avec l'algorithme A^*.
   Construire dans $walk_{k,l}$ le chemin valide dans G en utilisant les chemins

calculés avec A^*.
Pour chaque arc (i, j) de G Faire
Pour chaque fourmi k Faire
$\Delta\tau_{ij}^k = 0$ $\hookleftarrow$ **uniquement dans GU/AWA**
Pour chaque colonie l Faire
$\sigma_{ij}^{k,l} = \begin{cases} \frac{Q}{L_{k,l}} & \text{si } (i,j) \in \text{chemin décrit dans } walk_{k,l} \\ 0 & \text{sinon} \end{cases}$
$\Delta\tau_{ij} = \Delta\tau_{ij} + \sigma_{ij}^{k,l}$ $\hookleftarrow$ **uniquement dans GU/AA**
$\Delta\tau_{ij}^k = \Delta\tau_{ij}^k + \sigma_{ij}^{k,l}$ $\hookleftarrow$ **uniquement dans GU/AWA**
FinPour
FinPour
FinPour

5) Pour chaque arc (i, j) de G Faire
Calculer $\tau_{ij}(T + 1)$ (ou $\tau_{ij}^k(T + 1)$, $\forall k$), d'après l'équation (2) ou (4).
FinPour
$T = T + 1$
Pour chaque arc (i, j) de G Faire
Initialiser $\Delta\tau_{ij} = 0$ (ou $\Delta\tau_{ij}^k = 0, \forall k$).
FinPour

6) Si ($T < T_{MAX}$) Alors
Vider toutes les listes
Retourner à l'étape 2
Sinon
Afficher les chemins des r fourmis de la meilleure colonie, c'est-à-dire de la colonie pour laquelle la somme des longueurs des tours empruntés par ses fourmis est la plus petite.
Quitter l'algorithme
FinSi

Q est une constante définie par l'utilisateur. Elle représente la quantité fixe de phéromones déposées sur un arc.
Sélectionner comme solution la colonie pour laquelle la somme des longueurs des tours empruntés par ses fourmis est la plus petite permet de s'assurer que les fourmis emprunteront les plus courts chemins pour aller d'un nœud à un autre et donc que le critère WI sera minimisé.

5. Résultats expérimentaux

Les stratégies de patrouille multi-agent furent calculées sur six graphes différents de complexité variable (tableau 1), avec des populations constituées de 2 à 20 agents (plus précisément de $2, 3, 4, 5, 6, 7, 8, 9, 10, 15$ et 20 agents).

Nom	Nombre de nœuds	Nombre d'arcs	Degré	Complexité (Nombre d'états)
Cercle	56	56	2	1345
Couloir	70	69	2	1641
Carte B	50	69	5	59145
Grille	80	142	4	97633
Ile	50	84	6	726337
Carte A	50	106	7	16520713

Tableau 1. *Graphes étudiés (une mesure de la complexité de la tâche de patrouille est donnée pour chaque graphe en nombre d'états possibles, calculé en utilisant GBLA)*

Nous avons développé une plateforme permettant de faciliter l'implémentation et l'évaluation de n'importe quel comportement multi-agent pouvant être exhibé dans un environnement tridimensionnel. Nous l'avons utilisé (figure 3) pour évaluer nos stratégies de patrouille multi-agent.

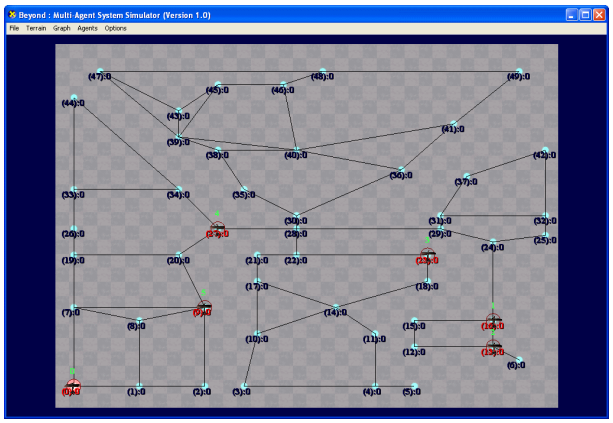


Figure 3. *Simulateur de systèmes multi-agents*

5.1. *Protocole expérimental*

L'apprentissage des stratégies de patrouille en utilisant GBLA a été réalisée de la manière suivante. Pour obtenir des stratégies aussi robustes que possible, leur apprentissage fut divisé en plusieurs essais. A chaque essai, les statistiques relatives au graphe (les oisivetés instantanées des nœuds ainsi que l'oisiveté moyenne du graphe) sont initialisées à zéro, tous les agents sont placés sur le même nœud initial et apprennent à patrouiller pendant plusieurs itérations en utilisant GBLA. Le nœud de départ des agents change d'un essai à l'autre. Ainsi, un total de 72 stratégies de patrouille (12 population d'agents $\times$ 6 graphes) furent apprises.

Nous avons conduit plusieurs expériences préliminaires afin de déterminer les meilleurs paramètres d'apprentissage de GBLA, comme le nombre d'essais, le nombre d'itérations par essai, le taux d'apprentissage α, le facteur de pondération γ et la probabilité d'exploration ϵ. Le tableau 2 présente les valeurs empiriques que nous avons utilisées lors de l'apprentisage des PDM des agents avec GBLA.

Nombre d'essais	1000
Nombre d'itérations par essai	10000
Taux d'apprentissage (α)	0.9
Facteur de pondération (γ)	0.9
Probabilité d'exploration (dans le Q-Learning)	10 %

Tableau 2. *Paramétrage utilisé lors de l'apprentissage dans GBLA*

Pour les techniques GU/AA et GU/AWA, nous avons utilisé les paramètres de contrôle indiqués dans le tableau 3.

Nombre de Cycles (T_{MAX})	10
Nombre de fourmis par agent (n)	10
Constante d'initialisation des phéromones (c)	0.01
Quantité de phéromones (Q)	100
Importance relative de la phéromone (α)	1
Importance relative de la visibilité (β)	5
Taux d'évaporation (ρ)	0.8

Tableau 3. *Paramétrage utilisé par les techniques GU/AA et GU/AWA*

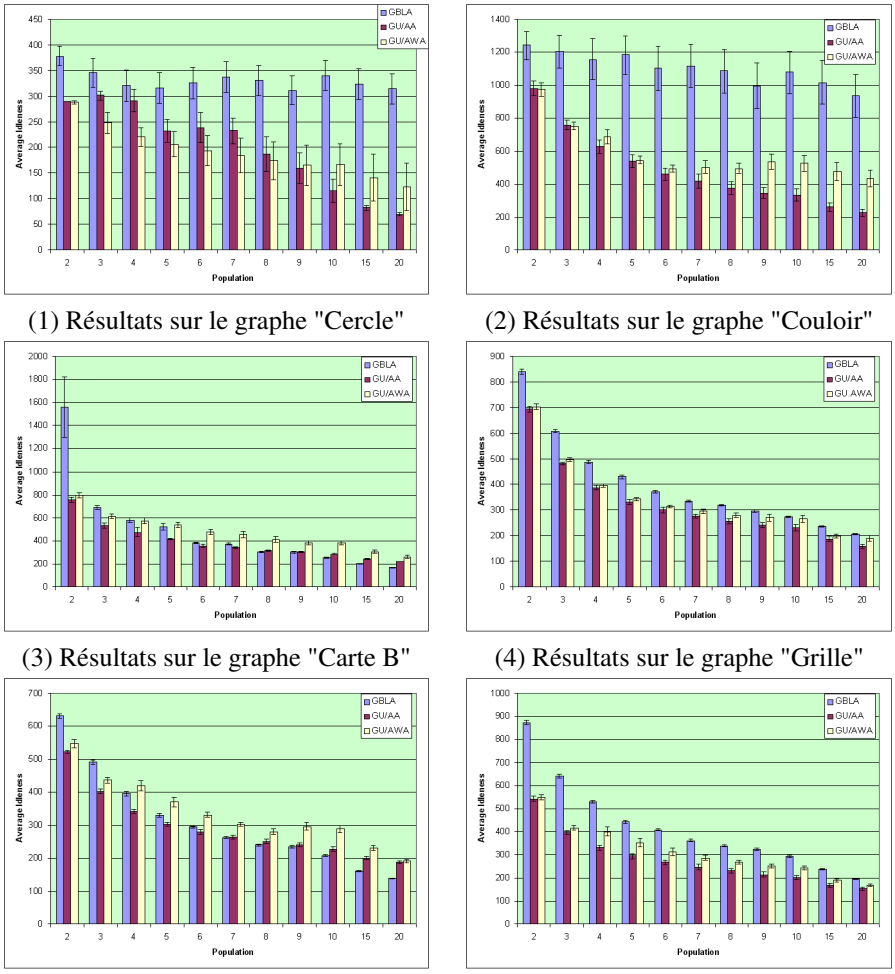
Dans les prochaines sections sont présentés les résultats expérimentaux obtenus avec GBLA et nos algorithmes basés sur ACO. Ces résultats permettent d'évaluer la robustesse des stratégies de patrouille multi-agent lorsque le nœud initial à partir duquel les agents commencent à patrouiller est modifié.

5.2. *Comparaison de GBLA et de GU/AA et GU/AWA*

Les figures 4.1 à 4.6 (page suivante) présentent l'oisiveté moyenne du graphe obtenue après une simulation de patrouille multi-agent en utilisant soit des stratégies apprises à l'aide de GBLA, soit des stratégies calculées à l'aide de nos méthodes basées sur des colonies de fourmis.

Chaque stratégie de patrouille fût évaluée 20 fois en modifiant la position de départ des agents et en utilisant $50\,000$ cycles de simulation. Les résultats suivants indiquent en fait la moyenne des oisivetés moyennes du graphe sur les 20 évaluations. L'intervalle de confiance indiquée sur les figures a été calculé en utilisant un risque de $5\,\%$.

Nous pouvons tout d'abord remarquer que les stratégies calculées à l'aide de nos algorithmes permettent aux agents de coordonner efficacement leurs actions pour



(1) Résultats sur le graphe "Cercle"

(2) Résultats sur le graphe "Couloir"

(3) Résultats sur le graphe "Carte B"

(4) Résultats sur le graphe "Grille"

(5) Résultats sur le graphe "Ile"

(6) Résultats sur le graphe "Carte A"

Figure 4. *Résultats de GBLA et de GU/AA et GU/AWA*

n'importe quel graphe. En effet, le critère de l'oisiveté moyenne du graphe décroît lorsque le nombre d'agents augmente. Ce n'est pas le cas pour GBLA, où les agents n'ont appris à coordonner leurs actions que pour les graphes "Carte A", "Carte B", "Grille" et "Ile". Malgré leur degré moindre, apprendre à patrouiller sur les graphes "Cercle" et "Couloir" paraît donc plus compliqué.

Les algorithmes GU/AA et GU/AWA sont significativement meilleurs que GBLA pour trois graphes, c'est-à-dire pour "Couloir", "Cercle" et "Carte A", quel que soit le nombre d'agents impliqués. Pour le graphe "Grille", seulement GU/AA est significativement meilleur que GBLA, indépendamment du nombre d'agents. GU/AA reste

significativement meilleur que GBLA sur les deux graphe "Ile" et "Carte B" lorsqu'au plus 6 agents patrouillent. Au-delà de 6 agents, GBLA donne de meilleurs résultats que GU/AA et GU/AWA.

En observant le comportement des agents patrouillant sur notre simulateur, nous avons récolté les trois remarques suivantes qui peuvent expliquer ces résultats.

Premièrement, GBLA est capable d'engendrer trois classes d'agents patrouilleurs. La première classe $\mathcal{C}_1$ est composée d'agents responsables uniquement d'une seule région : ils patrouillent les nœuds de cette région pendant toute la simulation. La deuxième classe $\mathcal{C}_2$ comprend des agents capables de visiter les nœuds de deux régions distinctes, et en particulier, également les nœuds liant ces régions, ce qui permet d'éviter de réduire les performances. Néanmoins, quelquefois le trajet emprunté par les agents n'est pas optimal, c'est-à-dire qu'ils peuvent traverser un même arc deux fois dans un délai très court. Ce comportement a été observé uniquement sur les quatre graphes les plus complexes ("Carte A", "Carte B", "Ile" et "Grille"). La dernière classe d'agents, la classe $\mathcal{C}_3$, est constituée d'agents adoptant la même stratégie de patrouille au même moment. Cette classe a été observée sur les graphes les plus simples, c'est-à-dire "Cercle" et "Couloir". Sur ces graphes en effet, tous les agents suivent la même politique : ils parcourent les nœuds du graphe dans la même direction et en même temps. Ceci explique pourquoi l'oisiveté moyenne du graphe ne décroît pas lorsque la taille de la population d'agents augmente.

Deuxièmement, GU/AA et GU/AWA traitent le problème de la patrouille multi-agent d'un point de vue global, en donnant la possibilité aux fourmis (et donc aux agents) d'atteindre de manière optimale (c'est-à-dire en suivant le chemin le plus court) des nœuds qui ne sont pas adjacents dans le graphe de patrouille G. Sur les graphes les plus complexes, les deux mêmes classes d'agents $\mathcal{C}_1$ et $\mathcal{C}_2$ fûrent observées. Néanmoins, le trajet emprunté par les agents dans GU/AA et GU/AWA est plus direct que dans le cas de GBLA, puisque les fourmis connaissent le chemin le plus court entre n'importe quel couple de nœuds grâce à l'algorithme A*. En ce qui concerne les graphes "Cercle" et "Couloir", le groupe initial des agents, qui sont placés sur le même nœud, est capable de se scinder en deux groupes distincts dès le départ de la patrouille. Chaque groupe d'agents continue alors de patrouiller dans des directions opposées. Au cours de la patrouille, ces groupes sont capables de se diviser encore plusieurs fois. Les agents quittant un groupe se dirigent alors à chaque fois dans le sens opposé à celui du groupe auquel ils appartenaient.

Troisièmement, les stratégies de patrouille calculées à l'aide de GU/AA et de GU/AWA contraignent les agents à commencer et à finir leur patrouille sur le même nœud. Cette hypothèse peut générer des stratégies efficaces uniquement sur certaines topologies de graphes.

6. Conclusion et perspectives

Nous avons décrit une application de l'outil d'optimisation par colonie de fourmis (ACO) au problème de la patrouille multi-agent lorsque les agents sont disposés initialement sur le même nœud. Deux nouveaux algorithmes ont été présentés et ils ont été comparés expérimentalement avec l'approche basée sur l'apprentissage par renforcement de Santana (Santana *et al.*, 2004). Nos deux algorithmes basés sur ACO utilisent des colonies de fourmis compétitives. Chaque colonie tente de découvrir la meilleure stratégie de patrouille multi-agent. Chaque fourmi d'une colonie coordonne ses actions avec les autres fourmis de la même colonie afin d'élaborer le circuit de patrouille d'un agent aussi court que possible. Les deux méthodes proposées permettent de surpasser de manière significative la technique de Santana sur quatre des six topologies de graphes étudiées, indépendamment du nombre d'agents patrouillant impliqués.

Plusieurs directions de recherche peuvent être explorées dans le but de trouver de meilleures solutions à ce problème multi-agent complexe en utilisant des algorithmes basées sur ACO.

Premièrement, des investigations sont en cours afin de déterminer les paramètres de contrôle adéquats pour nos algorithmes.

Deuxièmement, comme il a déjà été souligné dans la section 5, les stratégies de patrouille trouvées par nos méthodes contraignent chaque agent à commencer et terminer leur patrouille à partir du même nœud. Cette stratégie fournit des résultats satisfaisants uniquement pour certaines topologies de graphes. Lorsque les agents commencent à patrouiller à partir du même nœud, deux phases devraient être requises. La première phase consisterait à déterminer les stratégies individuelles des agents lorsqu'ils partent tous du même nœud initial s. La seconde phase aurait pour but de déterminer les stratégies individuelles des agents lorsque ceux-ci partent de l'avant dernier nœud atteint lors du circuit de la phase précédente. Cette approche reviendrait à exécuter GU/AA ou GU/AWA deux fois, une fois pour chaque phase.

Troisièmement, la qualité des solutions pourraient encore être vraisemblablement améliorée en intégrant des heuristiques spécifiques au problème de la patrouille multi-agent (comme par exemple $2-opt$), comme cela avait été réalisé par Dorigo et Gambardella (Dorigo *et al.*, 1997) pour le problème du voyageur de commerce.

Quatrièmement, la mise en compétition de nos colonies est actuellement relativement simple. En particulier, aucune information n'est échangée efficacement entre les colonies. Il serait par exemple intéressant d'introduire dans une colonie plusieurs types de phéromones, qui proviendraient d'autres colonies. Les fourmis seraient alors également guidées par le type de phéromones trouvé sur un arc. Cette voie de recherche permettrait d'accélérer davantage la recherche dans l'espace des solutions et serait donc susceptible d'améliorer la qualité de la solution trouvée.

Remerciements

Les auteurs tiennent à remercier les deux rapporteurs de cet article, qui grâce à leurs commentaires ont permis d'améliorer sa précision ainsi que sa pertinence.

7. Bibliographie

Almeida A., Ramalho G., *al*, « Recent Advances on Multi-Agent Patrolling », *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, p. 474-483, 2004.

Bell J., McMullen P., « Ant Colony Optimization Techniques for the Vehicle Routing Problem », *Advanced Engineering Informatics*, vol. 18, p. 41-48, 2004.

Chevaleyre Y., « Theoretical Analysis of the Multi-Agent Patrolling Problem », *International Joint Conference on Intelligent Agent Technology*, p. 302-308, 2004.

Chevaleyre Y., « The Patrolling Problem », *Annales du LAMSADE, Paris-Dauphine University, France*, 2005.

Colorni A., Dorigo M., Maniezzo V., « Distributed Optimization by ant colonies », *First European Conference on Artificial Life*, p. 134-142, 1991.

Colorni A., Dorigo M., Maniezzo V., « An Investigation of some properties of an ant algorithm », *Parallel Problem Solving from Nature Conference*, p. 509-520, 1992.

Dorigo M., Gambardella L., « Ant colony system : a cooperative learning approach to the travelling salesman problem », *IEEE Transactions on Evolutionary Computation*, vol. 1, p. 53-66, 1997.

Dorigo M., Maniezzo V., Colorni A., « The Ant System : Optimization by a colony of cooperating agents », *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 26, p. 1-13, 1996.

Gambardella L., Taillard E., Agazzi G., « MACS-VRPTW : A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows », *in* D. Corne, M. Dorigo, F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, p. 63-76, 1999.

Hart P., Nilsson N., Raphael B., « A Formal Basis for the Heuristic Determination of Minimum Cost Paths », *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, p. 100-107, 1968.

Hart P., Nilsson N., Raphael B., « Correction to 'A Formal Basis for the Heuristic Determination of Minimum Cost Paths' », *SIGART Newsletter 37*, p. 28-29, 1972.

Kawamura H., Yamamoto M., Suzuki K., Ohuchi A., « Multiple Ant Colonies Algorithm Based on Colony Level Interactions », *IEICE Transactions on Fundamentals*, vol. E83-A, 2000.

Kitano H., « RoboCup Rescue : A Grand Challenge for Multi-Agent Systems », *Proceedings of the 4th International Conference on Multi Agent Systems*, p. 5-12, 2000.

Lauri F., Charpillet F., « Ant Colony Optimization applied to the Multi-Agent Patrolling Problem », *IEEE Swarm Intelligence Symposium, Indianapolis, Indiana, USA*, 2006.

Machado A., Almeida A., *al*, « Multi-Agent Movement Coordination in Patrolling », *Proceedings of the 3rd International Conference on Computer and Game*, 2002a.

Machado A., Ramalho G., *al*, « Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures », *Proceedings of the 3rd International Workshop on Multi-Agent Based Simulation*, p. 155-170, 2002b.

Reinelt G., « The Traveling Salesman : Computational Solutions for TSP Applications », *Lecture Notes in Computer Science 840, Springer Verlag*, 1994.

Reuter E., Baude F., « System and Network Management Itineraries for Mobile Agents », *4th International Workshop on Mobile Agents for Telecommunications Applications*, p. 227-238, 2002.

Santana H., Ramalho G., *al*, « Multi-Agent Patrolling with Reinforcement Learning », *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, p. 1122-1129, 2004.

Schoonderwoerd R., Holland O., Brutten J., Rothkrantz L., « Ant-based load balancing in telecommunication networks », *Adaptive Behaviour*, vol. 5, p. 169-207, 1997.

Sempe F., Auto-organisation d'une collectivité de robots – Application à l'activité de patrouille en présence de perturbation, PhD thesis, Université Pierre et Marie Curie (Paris, France), 2004.

Sutton R., Barto A., *Reinforcement Learning : An Introduction*, Cambridge, MA, 1998.

Wagner I., Lindenbaum M., Bruckstein A., « Efficiently Searching a Graph by a Smell-Oriented Vertex Process », *Annals of Mathematics and Artificial Intelligence*, vol. 24, p. 211-223, 1998.

Wagner I., Lindenbaum M., Bruckstein A., « Distributed Covering by Ant-Robots Using Evaporating Traces », *IEEE Transactions on Robotics and Automation*, vol. 15, p. 918-933, 1999.

Yanowski V., Wagner I., Bruckstein A., « A Distributed Ant Algorithm for Efficiently Patrolling a Network », *Algorithmica*, vol. 37, p. 165-186, 2003.