Analyse théorique du problème de la patrouille multi-agent en utilisant le cadre des processus décisionnels de Markov

Fabrice Lauri, François Charpillet, Daniel Szer

Analyse théorique du problème de la patrouille multi-agent en utilisant le cadre des processus décisionnels de Markov

Fabrice Lauri, François Charpillet et Daniel Szer

LORIA/INRIA, Campus Scientifique, B.P. 239,
54506 Vandœuvre-Lès-Nancy

Mél : {lauri,charp,szer}@loria.fr

Résumé

Patrouiller implique habituellement une équipe d'agents dont le but consiste à visiter aussi fréquemment que possible les zones stratégiques d'un environnement. Pour une telle tâche, les agents impliqués doivent coordonner leurs actions afin d'atteindre des performances optimales. Les recherches actuelles sur le problème de la patrouille multi-agent (ou PPMA) considère généralement que l'environnement est réduit à un graphe métrique. Sous cette hypothèse, ce problème peut donc concerner une large gamme d'applications, telles que la gestion d'un réseau informatique, les jeux vidéo ou la détermination d'itinéraires de véhicules. Dans cet article, nous concentrons notre attention sur des instances particulières de ce problème. Nous considérons uniquement le pire cas où tous les agents commencent à patrouiller à partir d'un nœud donné. Nous formulons le problème de la patrouille multi-agent à l'aide d'un processus décisionnel de Markov (PDM). Trouver une politique optimale de patrouille se réduira alors à résoudre ce PDM. Nous prouvons d'une part que les stratégies multi-agents optimales sont nécessairement *cycliques*. D'autre part, nous avons montré que déterminer une stratégie de patrouille multi-agent consiste à trouver deux politiques à horizon fini. Un algorithme *meilleur d'abord* est utilisé pour déterminer une telle politique. Les résultats expérimentaux montrent que, pour toutes les configurations testées, notre approche améliore substantiellement ceux obtenus avec la méthode d'apprentissage par renforcement proposée par Santana *et al.* [10].

1 Introduction

Une patrouille est une mission impliquant une équipe de plusieurs agents dont le but consiste à visiter *continuellement* les zones pertinentes d'un environnement, afin de le superviser, le contrôler ou le protéger de manière efficace. Une équipe de facteurs réalisant leur tournée, une colonie de fourmis cherchant et rassemblant leur nourriture, ou un groupe de marines sécurisant un lieu sont tous des exemples de patrouilles. Une telle tâche nécessite alors que tous les membres impliqués coordonnent leurs actions afin d'atteindre des performances optimales.

Les techniques permettant de résoudre le problème multi-agent de la patrouille peuvent être utilisées dans de nombreuses applications : de la gestion d'un réseau informatique [9], au sauvetage par des robots de personnes blessées après une catastrophe naturelle [5, 6], en passant par la détection de menaces ennemies ou la protection d'une ville dans un jeu vidéo [7].

Ce problème a été abordé de manière rigoureuse seulement récemment [8, 1, 10, 2]. Dans ces travaux, une myriade de stratégies de patrouille ont été imaginées et validées expérimentalement en utilisant des critères d'évaluation communs [8]. Ces travaux utilisent des approches différentes, comme des lois heuristiques permettant aux agents de mieux choisir le prochain nœud à visiter [8], des mécanismes de négociation [1], des techniques d'apprentissage par renforcement [10] ou des techniques issues de la théorie des graphes [2]. La plupart des solutions trouvées produisent de bons résultats empiriques sur des graphes constitués de moins de cinquante nœuds et d'au plus une centaine d'arcs.

Nous proposons une analyse théorique du problème multi-agent de la patrouille (PPMA) dans le cadre des processus décisionnels de Markov. La principale question qui est adressée ici est : comment caractériser totalement le PPMA afin de pouvoir trouver des solutions optimales ? En d'autres termes, existe-t'il un espace de recherche permettant à des agents de synchroniser leurs actions et donc de trouver des stratégies de patrouille optimales, quel que soit le graphe de patrouille et la population d'agents. Pour réaliser cette étude, nous utilisons la définition formelle du PPMA introduite dans [8]. Par ailleurs, nous supposerons que tous les agents sont situés initialement sur le même nœud du graphe.

Pour répondre à cette question, le PPMA est tout d'abord formulé à l'aide d'un PDM. Nous montrons alors que trouver une stratégie multi-agent de patrouille équivaut à déterminer une politique π à horizon fini constitué de deux sous-politique π_{init} et π_{cyc}, π_{init} représentant la politique des agents dans une phase d'initialisation et π_{cyc} étant la politique cyclique que les agents suivent indéfiniment.

La suite de cet article est organisée de la manière suivante. La section 2 décrit dans un premier temps le cadre de travail du problème multi-agent de la patrouille, et présente dans un second temps l'état de l'art. La section 3 propose une formulation du PPMA en utilisant un PDM. Les résultats expérimentaux sont indiqués et analysés dans la section 4. Finalement, une conclusion et des perspectives de recherche sont données dans la section 5.

2 Le problème de la patrouille multi-agent

2.1 Cadre de travail mathématique

Le problème de la patrouille multi-agent est habituellement formulé comme suit [8, 2, 10]. L'environnement à patrouiller se 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 $\sigma_i : \mathbb{N} \to V$, $\sigma_i(j)$ représentant le j-ème nœud visité par l'agent i, avec $\sigma_i(j+1) = x$ ssi $(\sigma_i(j), x) \in E$.

Il est communément admis qu'une stratégie de patrouille pertinente est une stratégie qui minimise pour chaque nœud le délai entre deux visites.
Plusieurs critères ont été imaginés dans [8] 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, $I_n(0) = 0$, $\forall n = 1, 2, \cdots, |V|$.
A un instant t donné, GI_t est alors l'*oisiveté moyenne du graphe* (OMG), c'est-à-dire :

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

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 :

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

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 [3], 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, [8, 7] 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 communications entre agents (permises ou interdites), la perception des agents (locale vs 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.
[1] améliora les meilleures architectures proposées dans [7] 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 [3, 2] 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 [10], 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

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

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 [10].
Toutes les appproches décrites précédemment fûrent évaluées dans [1] et comparées sur 12 configurations (c'est-à-dire pour six topologies de graphes avec 5 et 15 agents). Il a été montré que la stratégie à cycle unique [2] donne les meilleurs résultats sur toutes les configurations sauf une, tandis que les deux meilleures architectures proposées dans [7, 8] produisirent les moins bons résultats. Toutes les autres techniques présentées dans [1] et [10] fournirent des performances équivalentes.
Notons que lorsque les positions initiales des agents sont fixées *a priori*, la stratégie à cycle unique [2] ne peut être utilisée puisque qu'elle considère que les positions initiales des agents sont également des inconnues du problème. C'est la raison pour laquelle nous avons comparé notre approche à l'une des meilleures de l'état de l'art dans ce cas : [10] en fait partie.

3 Un processus décisionnel de Markov pour le problème de la patrouille multi-agent (MDP-MAPP)

Nous faisons maintenant le lien entre le problème de la patrouille multi-agent et la théorie de contrôle discret basée sur les processus décisionnels de Markov. Il est d'abord expliqué comment adresser le PPMA sur des graphes unitaires. La section 3.3 prolonge ensuite notre approche aux graphes non unitaires.

3.1 Définition du PDM

Le problème de la patrouille multi-agent sur un *graphe unitaire* peut être caractérisé de manière globale par un processus décisionnel de Markov $\theta = (S, A, T, R)$, tel que :
- $S = |V|^r \times \mathbb{N}_+^{|V|}$ incorpore la *position actuelle* de chacun des r agents, ainsi que l'*oisiveté* de chacun des nœuds du graphe.
- $A \subseteq |V|^r$ représente l'ensemble des *actions jointes* possibles des agents.
- $T : S \times A \times S \to \{0, 1\}$ est la *fonction de transition* déterministe entre les états.
- R est égale au *critère d'évaluation* exposé dans la section 2.1 qui permet de produire des stratégies optimales, c'est-à-dire WI.

La fonction de transition peut être dérivée à partir de la structure du graphe. Si nous notons un état $s = \langle (p_1, \dots p_r), (I_1, \dots I_{|V|}) \rangle$, un autre état $s' = \langle (p'_1, \dots p'_r), (I'_1, \dots I'_{|V|}) \rangle$, et une action $a = (a_1, \dots a_r)$, alors $T(s, a, s') = 1$ ssi

$$(\forall k) : \quad \begin{array}{l} a_k = p'_k \\ (p_k, a_k) \in E \\ \begin{cases} I'_k = 0, & \text{si } (\exists m) \text{ tel que } a_m = k \\ I'_k = I_k + 1, & \text{sinon} \end{cases} \end{array} \tag{1}$$

Les PDMs multi-agents avec une observabilité totale et un espace d'actions jointes ont été appelés MMDPs [4]. En dépit du fait que le PDM défini ci-dessus ne soit pas totalement observable, puisque chaque agent est seulement conscient de sa propre position, les positions des autres agents et les valeurs des oisivetés de tous les nœuds peuvent être calculées à tout moment. Ceci est vrai dans le cas d'une planification optimale, puisque toutes les politiques individuelles sont connues par chaque agent avant leur exécution. La même remarque ne peut pas être formulée dans le cas d'un apprentissage.
Le MMDP θ est a priori un PDM infini, puisque son espace d'états n'est pas borné. Nous allons maintenant prouver que θ possède en fait une structure particulière qui facilite sa résolution. Nous commençons par présenter le concept de *cycle de nœuds* et celui de *cycle*.

Définition 3.1 (Cycle de nœuds)**.** *Nous définissons un* ***cycle de nœuds*** *de longueur L comme étant une séquence de positions jointes $(p_1, \dots p_r)_1, \dots (p_1, \dots p_r)_L$ telle que $(p_1, \dots p_r)_1 = (p_1, \dots p_r)_L$. Ceci signifie que les agents sont revenus à leur position initiale à l'issue de L pas de temps.*

Définition 3.2 (Cycle)**.** *Nous appelons un* ***cycle*** *de longueur k une séquence d'états $s_1, \dots s_k$ telle que $s_1 = s_k$. Ceci signifie en particulier que non seulement les positions des agents constituent un cycle, mais également la valeur des oisivetés des nœuds.*

Nous montrons maintenant que l'espace d'états pour résoudre le PPMA de manière optimale est effectivement fini.

Lemme 3.1. *Considérant une stratégie de patrouille optimale σ, la valeur de l'oisiveté de chaque nœud est* ***bornée****.*

Démonstration. Il est en effet simple de montrer que la valeur de WI_σ peut être bornée même pour une stratégie non-optimale visitant tous les nœuds. Nous définissons pour cela une stratégie σ' qui est cyclique sur les nœuds et de longueur L. Il est évident que l'exécution répétée d'une telle stratégie borne la pire oisiveté $WI_{\sigma'} \leq L$. Puisque la valeur pour n'importe quelle stratégie optimale est au pire aussi grande, alors $WI_\sigma \leq WI_{\sigma'} \leq L$. □

Lemme 3.2. *Le nombre d'états qui sont effectivement visités par une stratégie de patrouille optimale σ est* ***fini****.*

Démonstration. Il suffit de constater que le nombre des positions jointes possibles est fini (puisque le graphe a une taille finie), et que les valeurs des oisiveté pour chaque nœud est finie (voir lemme 1). Ceci garantit que le nombre d'états qui sont effectivement visités est fini. □

Nous pouvons maintenant établir notre théorème principal, qui stipule que la résolution du PPMA peut se réduire à trouver des stratégies finies ayant une structure particulière.

Théoreme 3.1. *Pour n'importe quel PPMA, il existe une stratégie de patrouille à horizon infinie σ qui est* ***cyclique****, et qui peut être représentée par $\sigma = \sigma_{init} \cdot (\sigma_{cyc})^*$.*

Démonstration. Supposons une stratégie optimale σ telle que $WI_\sigma = L$. Nous savons que le nombre d'états visités par une stratégie optimale est fini. Ceci signifie que σ contient des cycles. Pour chaque sous-stratégie cyclique $\sigma_{cyc} \subset \sigma$, nous pouvons garantir que $WI_{\sigma_{cyc}} \leq L$. Si nous appelons σ_{init} la sous-stratégie qui précède σ_{cyc} dans σ, nous pouvons aussi garantir que $WI_{\sigma_{init}} \leq L$. Nous pouvons donc construire une nouvelle stratégie $\sigma^* = \sigma_{init} \cdot (\sigma_{cyc})^*$ qui est au moins aussi bonne que σ. □

3.2 Résolution du PDM

Nous proposons ici une approche pour résoudre le PDM θ basé sur une recherche *meilleure d'abord*. Pour un graphe unitaire donné G et une distribution initiale des agents sur les nœuds, l'algorithme suivant explore l'espace d'états en choisissant, à chaque itération, à partir d'une liste appelée *OPEN*, l'état s avec la plus petite valeur $f(s)$. $f(s)$ est en fait la récompense obtenue (dans le sens du PDM θ) en atteignant l'état s. La liste OPEN stocke les états qui doivent être étendus. Les successeurs s' de s sont alors générés. Si un cycle est constitué, c'est-à-dire que s' et un de ses parents p sont égaux, alors il est considéré comme étant l'état final s_e du meilleur cycle découvert jusqu'à présent. Seulement les états s' qui forment un cycle et tels que $f(s') < f(s_e)$ pourront désormais remplacer l'état final d'un précédent meilleur cycle. Sinon, si un état généré s' n'appartient pas encore à la liste $OPEN$, alors il y est ajouté pour une prochaine expansion. Une fois que nous sommes assurés qu'aucun autre meilleur cycle ne peut être trouvé, c'est-à-dire lorsque $f(s') > f(s_e)$, l'expansion des états est terminée et les stratégies de patrouille individuelles des agents peuvent être déterminées en suivant les liens de s_e à s_0.
Notre algorithme *meilleur d'abord* peut être formellement décrit comme suit :

1. Placer les r agents sur les nœuds du graphe unitaire G
   Générer l'état initial s_0 correspondant, tel que
   $$s_0 = \langle (p_1, \ldots p_r), \quad \underbrace{(0, 0, \ldots, 0)}_{|V|} \rangle$$
   Ajouter s_0 à la liste OPEN.
2. Tant que la liste OPEN n'est pas vide
   (a) Récupérer l'état s ayant la plus petite valeur $f(s)$ à partir de la liste OPEN et le retirer de la liste
   (b) Si un cycle a déjà été découvert et $f(s) > f(s_e)$ Alors
   Sortir de la boucle *Tant que*
   Endif
   (c) Générer chaque successeur s' de s selon l'équation 1.
   (d) Pour chaque successeur s' de s
   i. Initialiser le parent de s' comme étant s
   ii. Définir $f(s') = \max\{f(s), \max_{n=1}^{|V|} I_n\}$
   iii. Si un cycle a été découvert, c'est-à-dire qu'il existe un parent p de s' tel que $p = s'$, Alors
   Si ce cycle est le premier Ou $f(s') \leq f(s_e)$ Alors
   $s_e = s'$
   Continuer (c'est-à-dire réitérer dans la boucle *Pour*)
   FinSi
   FinSi
   iv. Si s' est déjà dans la liste OPEN Alors
   L'ignorer et continuer
   v. Sinon Si un cycle n'a pas encore été trouvé Ou $f(s') \leq f(s_e)$ Alors
   Ajouter s' dans la liste OPEN
   FinSi
   FinPour
   FinTantQue
3. Construire les stratégies de patrouille individuelles σ_i de chaque agent i en suivant les liens des états s_e à s_0.

3.3 Considération des graphes métriques

N'importe quel graphe métrique $G = (V, E)$ peut être facilement transformé en un graphe unitaire $G' = (V', E')$ équivalent. Avant la transformation, $V' = V$ et $E' = \emptyset$. Soit c le coût minimal d'un arc reliant deux nœuds dans G. Ce coût sera considéré comme étant désormais le coût unitaire d'un arc dans G'. Alors, pour chaque paire de nœuds i et j tel que $i, j \in V$, $m_{ij} = (\frac{c_{ij}}{c} - 1)$ nœuds seront ajoutés à V'. Si $m_{ij} = 0$ alors seulement les arcs (i, j) et (j, i) seront ajoutés à E'. Sinon, si $n_1, n_2, \ldots, n_{m_{ij}}$ sont les m_{ij} nœuds ajoutés à V' afin d'obtenir des arcs unitaires entre les nœuds i et j, alors les *arcs dirigés*

$(i, n_1), (n_1, n_2), \ldots, (n_{m_{ij}}, j)$ seront ajoutés à E'. Les arcs dirigés contraindront ainsi les agents à suivre automatiquement les arcs reliant deux nœuds i et j tels que $i, j \in V$, sans réemprunter un arc déjà traversé. La figure 1 montre un exemple de graphe unitaire G' obtenue à partir d'un graphe métrique G en utilisant cette méthode.

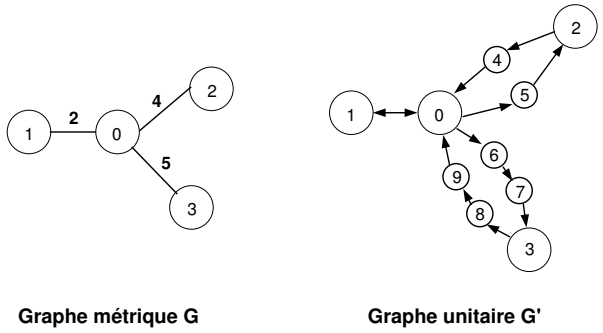


FIG. 1 – Graphe métrique G et son graphe unitaire G' correspondant

L'algorithme *meilleur d'abord* décrit dans la section précédente peut alors être appliqué à G' sous les conditions que $S = |V'|^r \times \mathbb{N}_+^{|V|}$, puisque seules les oisivetés des nœuds de G sont pertinents, et qu'un cycle constitue une séquence d'états $s_1, s_2, \ldots, s_k$ telle que $s_1 = s_k$ et $(p_1^1, p_2^1, \ldots, p_r^1), (p_1^k, p_2^k, \ldots, p_r^k) \in V^r$. Cette dernière condition garantit que les cycles trouvés sont valides, c'est-à-dire que les agents se trouvent sur les nœuds de G (et non quelque part sur un arc de G) lorsqu'ils commencent un cycle.

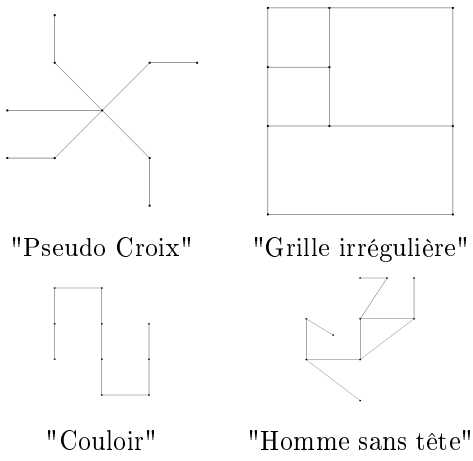
4 Evaluation expérimentale

Les strategies de patrouille multi-agents fûrent calculées sur quatre topologies de graphe de complexité différente (figure 1) avec une population constitué de un à quatre agents.
Nous avons développé un simulateur permettant de faciliter l'implémentation et l'évaluation de comportements multi-agents qui pourraient être exhibés dans un environnement tridimensionnel. Nous l'avons utilisé pour évaluer nos stratégies de patrouille.

4.1 Protocole expérimental

L'apprentissage des stratégies de patrouille en utilisant GBLA fût réalisé de la manière suivante. Afin d'obtenir des stratégies aussi robustes que possibles, leur apprentissage a été divisé en *essais*. Au début de chaque essai, les statistiques sur le graphe (oisiveté des nœuds et oisiveté moyenne du graphe) sont remises à zéro, tous les agents sont placés sur le même nœud initial et ils apprennent à patrouiller pendant un certain nombre d'itérations. Le nœud initial est changé d'un essai à l'autre. Ainsi, un total de 16 stratégies de patrouille (les 4 populations d'agents $\times 4$ topologies de graphes) fûrent apprises.
Nous avons conduit des expériences préliminaires afin de déterminer les paramètres d'apprentissage de GBLA les plus efficaces, tels que le nombre d'essais, le nombre d'itérations par essai, le taux d'apprentissage α, le facteur d'escompte γ et la probabilité d'exploration ϵ. Le tableau 2 montre les valeurs déterminées empiriquement et que nous avons utilisées pour réaliser l'apprentissage des PDMs.
Dans les prochaines sections seront présentés les résultats expérimentaux que nous avons obtenus avec GBLA et notre approche afin d'évaluer la robustesse des stratégies de patrouille multi-agent, lorsque le nœud où commencent à patrouiller tous les agents change.

TAB. 1 – Topologies de graphe



TAB. 2 – Paramètres utilisés dans la phase d'apprentissage de GBLA

Nombre d'essais	1000
Nombre d'iterations par Essai	10000
Taux d'apprentissage (α)	0.9
Facteur d'escompte (γ)	0.9
Probabilité d'exploration (dans Q-Learning)	10 %

4.2 Comparaison expérimentale

Les figures 2,3,4 et 5 indiquent l'oisiveté moyenne du graphe obtenue après une simulation de patrouille multi-agent en utilisant soit des stratégies calculées par MDPMAPP, soit des stratégies apprises par GBLA.
Chaque stratégie de patrouille fût évaluée 10 fois en changeant le nœud initial des agents et en utilisant

50000 cycles de simulation. Ainsi, les résultats suivants représentent l'oisiveté moyenne du graphe sur 10 simulations. Les intervalles de confiance indiqués sur les figures ont été calculés en utilisant un risque de 5%.

Il est à noter que seulement des graphes avec au plus dix noeuds ont été examinés. En effet, résoudre le MAPP de manière globale a un inconvénient principal : la durée de calcul nécessaire à sa résolution augmente exponentiellement avec la complexité du graphe et le nombre d'agents impliqués. Comme ordre d'idée, 80 minutes ont été nécessaires pour résoudre le MAPP sur le graphe "Pseudo Croix" avec seulement un agent, tandis que moins de 30 secondes étaient suffisantes pour GBLA.

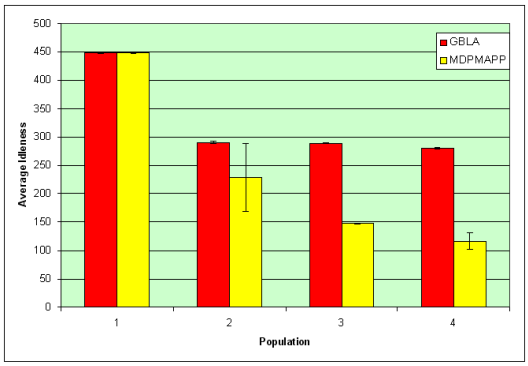


FIG. 2 – Résultats de comparaison sur le graphe "Pseudo Croix"

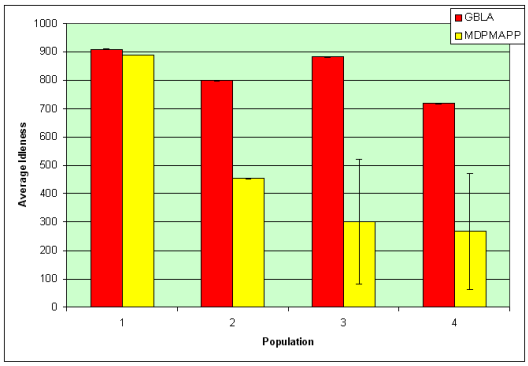


FIG. 3 – Résultats de comparaison sur le graphe "Grille irrégulière"

Nous pouvons déjà remarquer que les stratégies calculées par l'algorithme MDPMAPP permettent à des agents de coordonner leurs actions pour n'importe quelle topologie de graphe, puisque l'oisiveté moyenne du graphe diminue quand le nombre d'agents augmente. Ce n'est pas le cas en utilisant GBLA, où les agents ont appris à coordonner leurs actions seulement sur le graphe "Homme sans tête". Patrouiller sur le graphe "Couloir", qui possède pourtant un degré

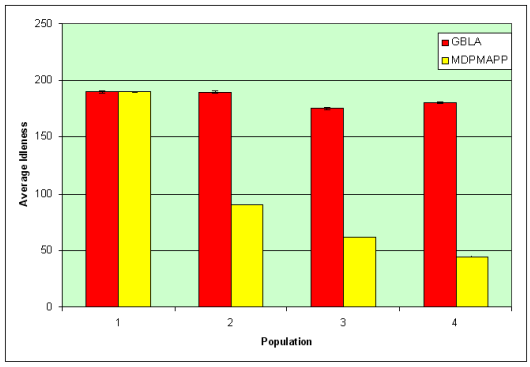


FIG. 4 – Résultats de comparaison sur le graphe "Couloir"

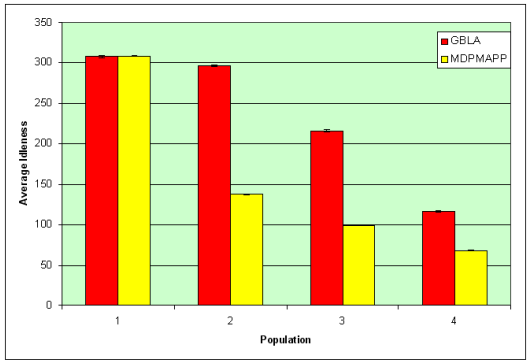


FIG. 5 – Résultats de comparaison sur le graphe "Homme sans tête"

minimal (degré 2), semble ainsi être plus compliqué. Comme prévu, l'algorithme MDPMAPP est significativement meilleur que GBLA pour tous les graphes, excepté pour le graphe "Pseudo Croix", quand un ou deux agents patrouillent.

En observant le comportement des agents sur notre simulateur, nous avons noté l'émergence de plusieurs organisations d'agents qui pourraient expliquer les résultats obtenus par GBLA. En effet, GBLA semble pouvoir créer des agents patrouillant qui peuvent être divisés en deux classes. La première classe se compose d'agents qui sont responsables d'une seule région du graphe : ils patrouillent uniquement les noeuds de cette région durant toute la simulation. La deuxième classe se compose d'agents qui traversent une région pour se rendre vers une autre, spécialement pour visiter les nœuds qui relient plusieurs régions, ce qui évite ainsi de diminuer les performances. Mais parfois, les chemins empruntés par les agents ne sont pas optimaux, c'est-à-dire qu'ils leur arrive de traverser un arc deux fois de suite dans un délai relativement court. Ces comportements fûrent uniquement observés sur le graphe "Homme sans tête". Sur les autres graphes, tous les agents suivent la même politique : ils parcourent tous chaque graphe dans la même direction et en même

temps. Ceci explique pourquoi l'oisiveté moyenne de ces graphes ne diminue pas quand le nombre d'agents augmente.

5 Conclusion

Nous avons présenté une analyse formelle du problème multi-agent de la patrouille à l'aide des processus décisionnels de Markov. Tandis que toutes les précédentes approches abordant ce problème permettent de trouver des solutions approximatives, notre approche permet de résoudre le PPMA de manière optimale. Nous avons prouvé que les stratégies de patrouille optimales peuvent toujours être représentées par des moyens finis, et nous avons introduit un algorithme de programmation dynamique capable de les déterminer. Les politiques obtenues fûrent testées sur quelques unes des topologies de graphes classiquement employées dans ce domaine afin de montrer leur efficacité. Nous envisageons d'une part d'étendre notre approche à des problèmes d'une plus grande complexité. D'autre part, le recours à des fonctions heuristiques admissibles permettra d'accélerer la recherche d'une stratégie de patrouille multi-agent optimale.

Références

[1] A. Almeida, G. Ramalho, and *al.* Recent Advances on Multi-Agent Patrolling. In *Proceedings of the 17th Brazilian Symposium on Artificial Intelligence*, pages 474–483, 2004.

[2] Y. Chevaleyre. Theoretical Analysis of the Multi-Agent Patrolling Problem. In *International Joint Conference on Intelligent Agent Technology*, pages 302–308, 2004.

[3] Y. Chevaleyre. The Patrolling Problem. In *Annales du LAMSADE nř4, Paris-Dauphine University, France*, 2005. available in french at http ://www.lamsade.dauphine.fr/~chevaley/papers/anna_patro.pdf.

[4] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the of the AAAI*, pages 746–752, 1998.

[5] RoboCup Rescue Homepage : http ://www.rescuesystem.org/robocuprescue/, last visit in November 2005.

[6] H. Kitano. RoboCup Rescue : A Grand Challenge for Multi-Agent Systems. In *Proceedings of the 4th International Conference on Multi Agent Systems*, pages 5–12, 2000.

[7] A. Machado, A. Almeida, and *al.* Multi-Agent Movement Coordination in Patrolling. In *Proceedings of the 3rd International Conference on Computer and Game*, 2002. available at http ://www-poleia.lip6.fr/~machado/files/aicg_patrol_final.pdf.

[8] A. Machado, G. Ramalho, and *al.* Multi-Agent Patrolling : an Empirical Analysis of Alternatives Architectures. In *Proceedings of the 3rd International Workshop on Multi-Agent Based Simulation*, pages 155–170, 2002.

[9] E. Reuter and F. Baude. System and Network Management Itineraries for Mobile Agents. In *4th International Workshop on Mobile Agents for Telecommunications Applications*, pages 227–238, 2002.

[10] H. Santana, G. Ramalho, and *al.* Multi-Agent Patrolling with Reinforcement Learning. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 1122–1129, 2004.