Travailler en équipe : le choix social appliqué au problème de la patrouille multi-agents

Cyril Poulet, Vincent Corruble, Amal El Fallah-Seghrouchni

▶ **To cite this version:**

Cyril Poulet, Vincent Corruble, Amal El Fallah-Seghrouchni. Travailler en équipe : le choix social appliqué au problème de la patrouille multi-agents. journées francophones sur les systèmes multi-agents (JFSMA '12), Oct 2012, Honfleur, France. hal-00753752

HAL Id: hal-00753752
https://hal.science/hal-00753752

Submitted on 19 Nov 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Travailler en équipe : le choix social appliqué au problème de la patrouille multi-agents

C. Poulet cyril.poulet@lip6.fr
V. Corruble vincent.corruble@lip6.fr
A. El Fallah Seghrouchni amal.elfallah@lip6.fr

Laboratoire d'Informatique de Paris 6 (LIP6),
Université Pierre et Marie Curie, France

Résumé

La patrouille multi-agents est un problème complexe dont le potentiel applicatif est vaste : dans les simulations à base d'agents, le management de crises, etc. Dans cet article, nous proposons deux stratégies coopératives à base d'enchères sur les nœuds à visiter. Ces stratégies s'inspirent de la théorie du choix social computationnel, et permettent aux agents de raisonner sur les performances du groupe plutôt que sur leurs performances individuelles. Nous montrons que ces stratégies présentent des performances similaires à celles des meilleures stratégies centralisées actuelles, et sont même meilleures pour certaines mesures.

Mots-clés : *coordination, patrouille, système ouvert*

Abstract

The multi-agent patrolling task constitutes a challenging issue for Artificial Intelligence and has the potential to cover a variety of domains ranging from agent-based simulations to crises management. In this article, we propose two decentralized, cooperative, auction-based strategies in which agents trade the nodes they have to visit. These strategies are inspired from the computational social choice theory and allow the agents to reason on the performances of the group rather than on their own. We show that these strategies perform at least as well as the state-of-the-art centralized performances, and better on specific criteria.

Keywords: *coordination, patrolling, open system*

1 Introduction

Le problème de la patrouille modélise de façon formelle la situation dans laquelle une équipe d'agents, humains ou virtuels, doit visiter de manière répétitive tous les points d'intérêt d'une zone. Deux variations de ce problème ont été définies ces dernières années : la *patrouille avec adversaires* dans laquelle la zone doit être protégée contre des adversaires qui tentent de s'introduire, et la *patrouille temporelle* où l'on cherche à rendre les visites aussi régulières et rapprochées que possible sur l'ensemble des points d'intérêt.

Le problème de la patrouille temporelle est très intéressant pour l'étude de la coordination dans les Systèmes Multi-Agents (SMA). D'une part il est à la fois suffisamment simple pour être implémenté facilement, et suffisamment vaste pour permettre la comparaison expérimentale de nombreuses stratégies de coordination. En effet, les performances atteintes par les diverses stratégies sont directement reliées à l'efficacité de la coordination entre les agents et de la façon dont ils se répartissent les visites sur l'ensemble des points d'intérêt. D'autre part, il permet de modéliser de nombreuses situations tant réelles qu'artificielles. Dans les jeux vidéos de guerre récents, il peut être important de patrouiller une zone pour détecter des changements dans l'environnement. Dans la vie réelle, le problème peut apparaître lorsque des agents humains ou robotiques sont chargés d'effectuer des actes répétitifs de maintenance ou de prévention (par exemple vérifier l'intégrité des lignes électriques pour éviter la panne électrique). Pour toutes ces raisons, ce problème a été proposé comme benchmark pour les SMA [6].

Récemment, une variante du problème de la patrouille temporelle a été proposée : la patrouille en système ouvert [9]. Dans ce nouveau problème, les agents sont libres d'entrer et sortir de la tâche de la patrouille à n'importe quel moment, obligeant ainsi le système à s'adapter et à se reconfigurer lorsque le nombre d'agents change. Cette dynamicité permet de modéliser des applications plus complexes, telles que des scénarios de sauvetage dans lesquels un sauveteur trouvant une victime peut et va quitter l'équipe patrouillant les ruines pour commencer les opérations de sauvetage. Un autre exemple possible est la gestion du changement d'équipes dans les usines nécessitant une maintenance per-

manente.
Utiliser des enchères pour allouer les nœuds aux agents patrouilleurs est une stratégie de coordination efficace et décentralisée qui permet d'éviter les goulots d'étranglement lorsque la population grandit, tant pour le calcul que pour les communications. Cette stratégie a été explorée avec succès dans [5], avec des communications synchrones. Cependant, lever cette hypothèse et utiliser des communications asynchrones amène une perte importante de performances (voir section 3.1). Dans ce contexte, nous proposons deux stratégies inspirées de la littérature concernant le choix social computationnel : **Minimax** et **Minisum**. Le choix social computationnel a suscité beaucoup d'intérêt récemment dans de nombreux problèmes de coordination ([2]), et en particulier pour l'allocation de ressources [7]. Cette voie a aussi été explorée dans le problème de l'exploration multi-robots ([3]). Nous proposons un protocole d'enchères basé sur [5] et montrons comment l'adapter à des enchères coopératives. Nous présentons ensuite en détail les stratégies Minimax et Minisum. Pour finir, nous montrons que ces deux stratégies décentralisées présentent des performances aussi bonnes que les meilleures stratégies centralisées actuelles et meilleures sur certains critères.
Dans cet article, la section 2 définit formellement la tâche de la patrouille, les métriques qui y sont associées, et présente l'état de l'art. La section 3 présente le protocole d'enchères, et détaille l'utilisation des critères de choix social dans ce mécanisme. La section 4 présente nos expériences, et nous discutons les résultats obtenus dans la section 5. La section 6 conclut cet article et propose des pistes de réflexion.

2 Le problème de la patrouille

2.1 la tâche de la patrouille

La tâche de la patrouille s'effectue dans un environnement représenté sous forme de *graphe*. Chaque nœud du graphe est un point d'intérêt de l'environnement, et chaque arête est un chemin entre deux points voisins. En étendant le cadre formel proposé dans [1] et [9], nous proposons la formalisation suivante de la tâche de la patrouille multi-agents.
La tâche de la patrouille multi-agents est représentée formellement par le tuple $\langle G, S, M\rangle$, dans lequel G est un graphe, S une société d'agents et M un ensemble de métriques. Le graphe $G = \langle N, E\rangle$ est composé d'un ensemble de nœuds N et de l'ensemble d'arêtes E correspondant. Chaque nœud $n_i \in N$ est pourvu d'une priorité p_i. Chaque arête $e_j \in E$ possède une longueur l_j représentant la distance entre les deux extrémités d'e_j. G peut être statique, ou évoluer au cours du temps : les nœuds peuvent devenir accessibles ou inaccessibles, les arêtes peuvent devenir impraticables, les priorités peuvent changer.
La société d'agents $S = \{a_i\}_{i\in N_S}$ est un ensemble d'agents a_i de taille N_S. Chaque agent est défini par les ensembles de perceptions et d'actions dont il est pourvu. Les perceptions principales sont :
– **perceptions de l'environnement** : Le temps interne de la simulation (P_t), la position de l'agent (P_{Self}), le graphe autour de l'agent(P_G) jusqu'à une distance d_G autour de lui. D'autres perceptions peuvent être ajoutées si nécessaire.
– **perception de la société d'agents** : La position des autres agents (P_{Soc}) s'ils sont situés à moins de d_{Soc} de l'agent, et les communications (P_C).
Les actions possibles sont :
– **sur l'environnement** : Visiter le nœud sur lequel l'agent est situé (A_{visit}), et se déplacer vers un nœud voisin (A_{GoTo}). D'autres actions peuvent être ajoutées si nécessaire.
– **sur la société d'agents** : Envoyer un message (A_C) en broadcast ou à un destinataire unique. Dans les deux cas, le destinataire ne peut être à une distance supérieure à d_C de l'envoyeur pour pouvoir recevoir le message.
La société S peut être fermée - le nombre d'agents est constant dans le temps - ou ouverte - les agents peuvent rejoindre ou quitter la société à tout moment.
Enfin, M est un ensemble de critères d'évaluation basés sur la distribution temporelle des visites. La tâche de la patrouille temporelle multi-agents est alors pour les agents de S de visiter tous les nœuds de G de manière répétitive pour optimiser les critères de M.
Cet article s'intéresse avant tout à l'aspect dynamique de la société S. Nous allons donc considérer que G est statique. De même, tous les nœuds sont de même importance, donc les priorités sont toutes égales. Enfin, les communications ne sont pas restreintes : les agents peuvent communiquer aussi souvent et aussi loin que nécessaire ($d_C = \infty$).

2.2 Métriques

La première métrique proposée historiquement pour le problème de la patrouille en système

fermé est l'oisiveté $Id_i(t)$ [4]. Si t^i_{visit} est la date de la dernière visite sur le nœud n_i, l'oisiveté instantanée du nœud est $Id_i(t) = t - t^i_{visit}$. Ce critère peut alors être moyenné sur le graphe (oisiveté instantanée du graphe $Id_G(t)$) et sur la durée (oisiveté du graphe $Id_G{}^{t_1}_{t_2}$). Cependant, l'oisiveté du graphe est difficile à relier directement aux événements qui se déroulent pendant la simulation. Pour cette raison, [10] propose des métriques basées sur les intervalles entre les visites : l'intervalle moyen I_{av} et l'intervalle carré moyen (Mean Square Interval ou MSI). Nous définissons N comme l'ensemble des nœuds de G, $n_i\_Intervals$ celui des intervalles entre les visites du nœud n_i durant la simulation, $G\_Intervals$ celui des intervalles de la simulation et $|I_{n_i}|$ est la longueur (durée) de l'intervalle I_{n_i} du nœud n_i. Alors le MSI s'écrit comme présenté dans l'eq. 1. Avec $|N|$ le nombre de nœuds de G et T_{max} la durée totale (en cycles) de la simulation, I_{av} s'écrit comme présenté dans l'eq. 2. Or, nous pouvons démontrer que l'intuition de [10] est bonne en reliant ces deux critères à l'oisiveté du graphe par la relation 3. Nous proposons donc d'utiliser I_{av} et MSI à la place de l'oisiveté du graphe, puisqu'elles fournissent des informations supplémentaires : la fréquence moyenne des visites pour le premier critère, et la bonne répartition de ces visites sur l'ensemble du graphe pour le deuxième. Nous garderons en revanche $Id_G(t)$ lorsque nous chercherons à caractériser l'état du système à un instant t.

$$MSI = \sqrt{\frac{\sum_{\{n_i \in N\}} \sum_{\{I_{n_i} \in n_i\_Intervals\}} |I_{n_i}|^2}{card(G\_Intervals)}} \tag{1}$$

$$I_{av} = \frac{\sum_{\{n_i \in N\}} \sum_{\{I_{n_i} \in n_i\_Intervals\}} |I_{n_i}|}{card(G\_Intervals)} = \frac{|N| \times T_{max}}{card(G\_Intervals)} \tag{2}$$

$$Id_G{}^{T_{max}}_0 = \frac{MSI^2}{2I_{av}} - \frac{1}{2} \tag{3}$$

Pour le problème de la patrouille en système ouvert, plusieurs critères additionnels ont été proposés dans [9] pour mesurer les performances durant les phases de transition :

– *temps de stabilisation* : représente le temps mis par le système pour retrouver sa stabilité. Pour mesurer ce temps, [9] proposent de calculer une moyenne dans la phase stable suivant la transition sur l'un des critères, puis de la comparer à des moyennes calculées sur une période glissante de 100 cycles à partir du début de la période de transition (changement du nombre d'agents). Le temps de stabilisation est mesuré lorsque la moyenne glissante est à moins de 1% de la moyenne stable.
– *amplitude de variations* : mesure la perte de performances pendant la phase de transition. [9] proposent pour cela d'utiliser le ratio entre la valeur maximale de $Id_G(t)$ mesurée durant la transition et l'oisiveté moyenne du graphe durant la phase stable montrant les pires performances entre celle avant et celle après la transition.

2.3 État de l'art

De nombreuses stratégies ont été proposées pour la tâche de la patrouille en système fermé (par exemple [4]). Dans cet article, nous ne détaillerons que celle qui présentent les meilleures performances actuelles et qui donc serviront de références. Pour les approches centralisées, deux stratégies se distinguent :

– **la stratégie à cycle unique (Single Cycle ou SC)** proposée dans [1]. Un coordinateur calcule le cycle minimal du graphe par une méthode proche du voyageur de commerce (P_G avec d_G=∞), puis distribue uniformément les agents autour de ce chemin unique en fonction de leurs points de départ (P_{Soc} avec d_{soc}=∞, A_C avec d_C=∞). Les autres agents sont de simples exécutants : ils n'ont besoin que de leur propre position (P_{Self}), de la position de la prochaine destination (P_G avec d_G=1) et de savoir comment se déplacer et visiter un nœud. Enfin, pour synchroniser correctement les exécutants, un agent est pris comme référence. Les autres agents doivent pouvoir détecter son passage (P_{Soc} avec d_{soc}=0).
Ainsi que décrit dans [9], l'ouverture du système implique que les agents doivent être redistribués après chaque événement (arrivée ou départ d'agents participant à la patrouille). Les agents sont alors arrêtés pour diverses durées de manière à générer des espaces pour les nouveaux arrivants, ou au contraire à combler les espaces laissés par les agents sortant.
– **l'Heuristic Pathfinding Cognitive Coordinated agent (HPCC)** décrit dans [4]. Après chaque visite, un coordinateur central calcule la nouvelle cible de l'agent ayant réalisé la visite en combinant la distance à la cible et l'oisiveté prévue lors de la future visite. L'agent visitant calcule alors le chemin le plus intéressant pour se rendre à sa cible en cher-

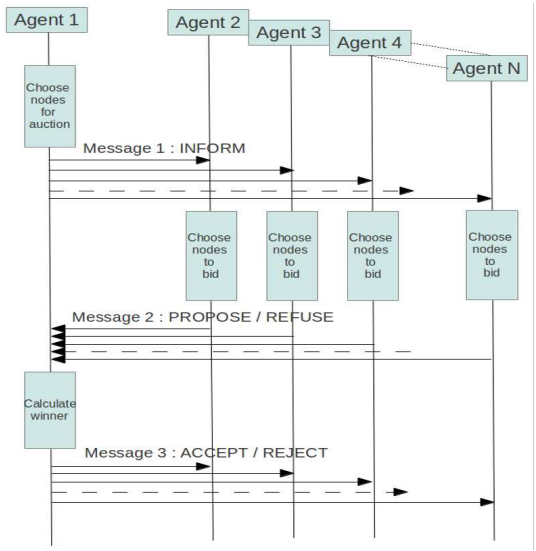

FIGURE 1 – Protocole d'enchères

chant à maximiser l'oisiveté des nœuds visités le long de ce chemin. Cette stratégie est naturellement adaptée à un système ouvert. Elle suppose que tous les agents, coordinateur compris, connaissent la totalité du graphe (P_G avec d_G=∞) et que chaque agent puisse communiquer avec le coordinateur à partir de toutes les positions possibles (A_C avec d_C=∞).

La table 1 récapitule les capacités et contraintes associées de ces stratégies.
De nombreuses approches décentralisées ont aussi été proposées. On peut citer l'utilisation d'enchères avec le Flexible Bidder agent de [5] sur lequel nous reviendrons dans la section 3, l'apprentissage par renforcement dans [12] ou l'approche gravitationnelle de [10].

3 Protocole d'enchères et critères sociaux

Le choix social computationnel a reçu beaucoup d'attention ces dernières années ([2], [7]) mais n'a jamais été appliqué au problème de la patrouille. Cette tâche est pourtant extrêmement coopérative par définition. Cela en fait un banc d'essai idéal pour comparer des stratégies basées sur la théorie du choix social à d'autres stratégies de coordination. Nous allons dans un premier temps présenter le protocole d'enchères, puis détailler les critères sociaux que nous avons choisis.

3.1 Protocole d'enchères

Dans la lignée de [5], le protocole d'enchères que nous avons sélectionné est scellé et à valeurs privées (fig. 1). Une enchère se déroule de la manière suivante :
– L'agent initiateur (agent 1 dans la fig. 1) choisit 1 ou 2 nœuds qu'il souhaite échanger, en utilisant une fonction de coût donnée. Il informe alors les autres agents du début des enchères par un message INFORM contenant les nœuds proposés. Chaque agent peut initier des enchères.
– Les autres agents peuvent alors décider de proposer des nœuds en échange (1 pour 1, 1 pour 2 ou 2 pour 2) via un message PROPOSE contenant les nœuds proposés, ou refuser de participer aux enchères (message REFUSE). Seul l'agent initiateur a connaissance des réponses des agents.
– Enfin, l'agent initiateur classe les propositions reçues lorsqu'il a reçu toutes les réponses attendues. Il accepte l'échange le plus intéressant (message ACCEPT) et rejette les autres (message REJECT).

Pour le bon fonctionnement de ce protocole, les agents doivent connaître tout le graphe (pour la coût) et pouvoir communiquer avec tous les autres agents (A_C avec d_C=∞)(voir table 1). Cependant, et contrairement aux stratégies centralisées, ces contraintes peuvent être assouplies simplement. Limiter temporellement la période où les enchères sont possibles permettrait d'ôter le besoin de communication avec tous les agents, et donc d'enlever la contrainte d_C=∞. Les agents pourraient aussi avoir une connaissance du graphe limitée à un sous-graphe contenant leurs propres nœuds et leurs voisinages, et ne participer que s'ils connaissent les nœuds proposés aux enchères.
Ce protocole a été proposé pour des enchères compétitives dans la stratégie du Flexible Bidder Agent (FBA) ([5]). Les agents cherchent à maximiser leur gain dans chaque échange en échangeant les nœuds coûtant le plus (c'est à dire maximisant la différence entre la longueur actuelle du chemin et la longueur du chemin après que le nœud soit échangé). Cette stratégie a montré de très bons résultats, mais a été implémentée avec des communications synchrones. Ce mode de communication est une condition extrêmement restrictive pour une stratégie décentralisée, car il nécessite que tous les agents soient instantanément disponibles au déclenchement des enchères. Utiliser des communications asynchrones est une approche plus réaliste pour un système multi-agent, car alors chaque agent

Stratégies	Perceptions / Actions	Contraintes max
HPCC	coord. : $P_G,\ P_C,\ A_C$ agents : $P_{Self},\ P_G,\ P_C,$ A_{visit}, A_{GoTo}, A_C	$d_G = \infty$ $d_C = \infty$
SC	coord. : $P_G,\ P_{Soc}, A_C$ agents : $P_{Self},\ P_G,\ P_{Soc},$ P_C, A_{visit}, A_{GoTo}	$d_G = \infty,\ d_{soc} = \infty$ $d_C = \infty$
stratégies à base d'enchères	$P_{Self},\ P_G,\ P_C$ A_{visit}, A_{GoTo}, A_C	$d_G = \infty, d_C = \infty$ (peuvent être levées, voir 3.1)

TABLE 1 – Contraintes des stratégies de patrouille.

est libre de gérer les messages qu'il reçoit au moment le plus adéquat pour lui.
Pour cette raison, nous proposons d'adapter cette stratégie pour des communications asynchrones : les différentes enchères peuvent maintenant être simultanées, et un agent ne peut pas proposer un nœud dans plus d'une enchère simultanée (les nœuds sont bloqués jusqu'à la fin de l'enchère concernée). Cependant, cette condition sur les nœuds implique qu'un agent ne peut pas envisager tous les échanges possibles pour une enchère donnée, puisque certains nœuds ne sont pas disponibles pour cette enchère. Ainsi, alors que dans la version synchrone du FBA ([5]) c'est toujours le meilleur échange possible qui est choisi, ce n'est plus le cas notre version asynchrone. De plus, ce meilleur échange peut ne plus être possible dans le futur, puisque les agents n'acceptent que des échanges mutuellement avantageux. Ceci est la cause d'une perte importante de performances de notre version par rapport à la version de [5] (ainsi que nous le montrons dans la partie 4). Cette observation nous amène à poser la question suivante : comment cet effet négatif peut-il être contrebalancé pour atteindre de bonnes performances sur la tâche de la patrouille ? Nous montrons dans cet article que la théorie du choix social propose des réponses intéressantes à cette question.
Dans la suite de cet article, FBA fera référence à notre version asynchrone de cette stratégie.

3.2 Offre et attribution sociales

Utiliser des critères de choix sociaux dans ce protocole d'enchères permet aux agents de considérer des échanges qui n'étaient pas permis dans des enchères compétitives : les échanges qui ne sont pas mutuellement avantageux mais qui amélioreront la situation de la société. Ceci permet une meilleure allocation des nœuds entre les agents, et donc permettre de meilleures performances sur la tâche de la patrouille. Cependant, nous ne proposons pas de garanties d'optimalité, à cause de la parallélisation des processus d'enchères.

Description générale. Le protocole général est le même que celui de la stratégie FBA, avec la différence suivante : les agents peuvent proposer des dons, c'est à dire des échanges 0 contre 1 (défini dans [11] sous le terme *O-contract*), pour soulager l'agent initiateur.
Soit N_i l'ensemble des nœuds de l'agent i, et f la fonction de bien-être individuel calculée sur l'ensemble de nœuds de l'agent. Pour des raisons de clarté, et f étant toujours égale dans cet article à la longueur du plus court chemin passant par tous les nœuds de N_i, nous noterons PL (pour PathLength) cette fonction de bien être individuel. Par la suite, $PL(N_i)$ désignera toujours la longueur du chemin de l'agent i avant transaction. Nous avons alors :

– L'agent initiateur a_1 choisit les nœuds n_{11}, n_{12} qui, otés de son parcours, minimise le nouveau parcours :

$$\begin{cases} n_{11} = \text{argmin}_{n_{1i} \in N_1} PL(_{N_1 \setminus \{n_{1i}\}}) \\ n_{12} = \text{argmin}_{n_{1i} \in N_1 \setminus \{n_{11}\}} PL(_{N_1 \setminus \{n_{11}, n_{1i}\}}) \end{cases}$$

Il envoie alors "INFORM n_{11}, $i_1(n_{11})$, n_{12}, $i_1(n_{12})$ ", dans lequel i_1 est une fonction de coût qui permet aux autres agents d'évaluer s'ils doivent proposer une transaction non avantageuse.

– Les agents a_2 à a_n choisissent alors de participer ou non. Pour n_{11} par exemple, a-t'on $PL(N_k \cup \{n_{11}\} \setminus \{n_k\}) < PL(N_k)$?
Si c'est le cas, il existe un échange avantageux pour a_k : a_k envoie à a_1 la liste des couples $(n_k, i_k(n_{11}, n_k))$ vérifiant cette équation. S'il n'en existe pas, l'agent recherche les échanges non avantageux mais intéressants pour la collectivité. On appellera C^R la condition d'acceptation qui, si elle est vérifiée, signifie qu'un don d'a_1 à a_k tombe dans cette catégorie. a_k propose alors à a_1 de faire

ce don en envoyant le message "PROPOSE $i_k(n_{11})$". Si C^R n'est pas vérifiée, a_k envoie un "REFUSE".
– La phase d'attribution s'effectue en deux étapes. D'abord, a_1 classifie les propositions en 4 catégories : dons, échanges avantageux vérifiant
$PL(N_1 \cup \{n_k\} \setminus \{n_{11}\}) < PL(N_1)$,
échanges non avantageux mais collectivement intéressants (vérifiant une condition d'attribution C^A), et les échanges inintéressants. Puis les propositions des 3 premières catégories sont classées selon le critère de bien-être choisi, et le meilleur est accepté. Les autres sont rejetés.

Les fonctions coût dépendent du critère de bien-être choisi. Nous allons donc présenter maintenant deux critères possibles : minimax et minisum.

Minimax. Le critère de *bien-être social égalitaire* cherche à maximiser l'utilité de l'agent le plus pauvre de la société. Pour appliquer ce critère à la tâche de la patrouille, nous choisissons comme utilité la longueur du chemin que chaque agent doit parcourir pour visiter l'ensemble de ses nœuds. Bien sûr, nous cherchons alors à minimiser le chemin le plus long dans la société, d'où la dénomination de **minimax** :

$$criterion \ : \ \min \max_{a \in A} PL(N_a)$$

Pour permettre aux agents de proposer des dons, nous proposons la valuation et la condition C^R suivante :

$$\begin{cases} i_1(n_{11}) = PL(N_1) \\ C^R \ : \ PL(N_1) > PL(N_k \cup \{n_{11}\}) \end{cases}$$

Si la condition C^R est vérifiée, cela signifie que donner n_{11} à a_k ne crée pas de chemin plus long qu'un chemin actuel. Si a_1 est l'agent le plus pauvre, il sera moins pauvre après le don. Sinon, le chemin le plus long ne sera pas modifié. Le don est donc intéressant pour la collectivité.
Pour classifier les propositions, nous proposons :

$$\begin{cases} i_k(n_{11}, n_k) = PL(N_k \cup \{n_{11}\} \setminus \{n_k\}) \\ C^A \ : \ PL(N_k \cup \{n_{11}\} \setminus \{n_k\}) \\ \qquad\qquad > PL(N_1 \cup \{n_k\} \setminus \{n_{11}\}) \end{cases}$$

si C^A est vérifiée, le nouveau chemin d'a_k est toujours plus long que le nouveau chemin d'a_1. Sachant que l'échange a été proposé par a_k et qu'il est donc avantageux pour celui-ci, cela signifie que si l'échange se fait, soit a_k était l'agent le plus pauvre et il s'est enrichi (tout en s'assurant qu'a_1 n'est pas devenu l'agent le plus pauvre), soit le chemin le plus long n'a pas été modifié. L'échange est donc intéressant pour la collectivité.
Ces conditions assurent que quelle que soit la transaction choisie, le chemin le plus long est réduit ou inchangé. Enfin, les propositions sont classées de la façon suivante :
– On considère en premier les échanges avantageux pour a_1 (qui par construction sont donc mutuellement avantageux). On choisit alors celui qui maximise le gain d'a_1 (i.e. maximisant la différence entre la longueur actuelle de son chemin et la longueur après échange).
– On s'intéresse ensuite aux échanges non avantageux pour a_1 : ils ont été proposés par des agents dont le chemin est plus long que celui d'a_1. On choisit celui dont i_k est maximal, en considérant que c'est probablement celui ayant le chemin courant le plus long.
– Enfin, on regarde les dons : si aucun échange n'a été trouvé, c'est que a_1 est l'agent le plus pauvre. On choisit alors le don ayant le plus petit i_k. Ainsi, on décharge a_1 sur l'agent qui souffrira le moins du surcroît de travail.

Une autre classification possible pourrait être de choisir la proposition venant de l'agent le plus pauvre parmi ceux qui ont répondu. Cependant, cela demanderait aux agents de partager son utilité courante en sus de son utilité projetée. Nous avons considéré que les agents partageraient le moins d'information possible, même au coût d'une stabilisation plus longue.

Minisum. Le critère de *bien-être social utilitaire* cherche à maximiser l'utilité moyenne des agents de la société. Comme pour le minimax, l'utilité que nous avons utilisée est la longueur du chemin de l'agent, et nous cherchons à minimiser la somme des utilités :

$$criterion \ : \ \min \sum_{a \in A} PL(N_a)$$

Pour des raisons de clarté, nous introduisons la notation *diff* : avec S_1, S_2 deux ensembles de nœuds,

$$diff_{(N_k, S_1, S_2)} = PL_{(N_k)} - PL_{((N_k \cup S_1) \setminus S_2)}$$

Les valuations et conditions pour proposer un don s'écrivent alors :

$$\begin{cases} i_1(n_{11}) = diff(N_1, \emptyset, \{n_{11}\}) \\ C^R \ : \ diff(N_1, \emptyset, \{n_{11}\}) \\ \qquad\qquad > diff(N_k, \{n_{11}\}, \emptyset) \end{cases}$$

si C^R est vérifiée, la différence entre le chemin actuel et le chemin prévu est plus grande pour a_1 que pour a_k. Donner le nœud va donc diminuer la somme des longueurs de leurs chemins, ce qui est intéressant pour la collectivité.
Pour catégoriser les propositions, nous proposons les valuation et condition suivantes :

$$\begin{cases} i_k(n_{11}, n_k) = diff(N_k, \{n_{11}\}, \{n_k\}) \\ C^A \ : \ diff(_{N_k,\{n_{11}\},\{n_k\}}) \\ \qquad\qquad > diff(_{N_1,\{n_k\},\{n_{11}\}}) \end{cases}$$

Ici encore, sachant que si C^A est examinée, c'est que l'échange est avantageux pour a_k, le fait que C^A soit vérifiée signifie que a_k gagne plus qu'a_1 ne perd en cas d'échange. La somme des longueurs de leurs chemins diminue donc, ce qui est intéressant pour la collectivité.
Pour l'attribution, nous pouvons écrire que pour des enchères données, commençant à t_1, finissant à t_2 et attribuées à a_b,

$$\sum_{a \in A} PL(N_a)(t_1) - \sum_{a \in A} PL(N_a)(t_2) =$$

$$\begin{cases} \text{don :} \\ \qquad diff(N_1, \emptyset, \{n_{11}\}) - diff(N_b, \{n_{11}\}, \emptyset) \\ \text{echange :} \\ \qquad diff(N_b, \{n_{11}\}, \{n_b\}) \\ \qquad\qquad -diff(N_1, \{n_b\}, \{n_{11}\}) \end{cases}$$

a_1 attribue donc les enchères à la proposition maximisant le gain sur le critère minisum.

3.3 Choix du mécanisme de transition

Parmi de nombreux mécanismes de réorganisation possibles (qui ne sont pas le focus principal de l'article et sont décrits dans [8]), nous avons envisagé deux mécanismes, chacun adapté à l'une des stratégies précédentes, pour gérer l'entrée et la sortie des agents. Pour la stratégie Minimax, nous avons utilisé les mécanismes par proximité, qui fonctionnent comme suit :

– Quand un agent entre, il publie sa position. Chaque autre agent (les accueillants) lui donne alors au plus la moitié de ses nœuds, pris parmi ceux situés à moins de d_{prox} de la position reçue. Si un accueillant n'a aucun nœud éligible, il envoie le nœud le plus proche. L'entrant accepte tous les nœuds du voisinage, et complète si nécessaire avec les autres nœuds.
– Quand l'agent sort, des enchères sont lancées sur ses nœuds. Seuls les accueillants ayant des nœuds à moins de d_{prox} des nœuds proposés peuvent y participer. Chaque nœud est alors attribué à l'accueillant possédant le nœud le plus proche. Tous les nœuds non attribués après le premier tour sont attribués par proximité au nœuds déjà attribués (i.e. à l'accueillant à qui a déjà été attribué le nœud le plus proche).

Pour ces deux mécanismes, les meilleures performances ont été obtenues pour une valeur de d_{prox} égale à la longueur moyenne des arêtes du graphe.
Pour la stratégie Minisum, les mécanismes de proximité se sont révélés inadaptés car certains agents se retrouvent avec trop de nœuds que le protocole minisum ne permet pas de donner (l'agent receveur perd plus que l'agent donneur ne gagne). Nous avons donc opté pour des mécanismes répartissant mieux les nœuds :

– via le mécanisme des pires nœuds, à l'entrée d'un agent, chaque accueillant envoie un nombre donné de nœuds, ceux qui permettent individuellement de raccourcir le plus leur chemin actuel ;
– via le mécanisme du pire groupe précalculé, lorsqu'un agent sort, le sortant calcule des groupes de nœuds d'une certaine taille par proximité, et débute des enchères. Chaque accueillant cherche à obtenir le groupe qui rallonge le moins son chemin actuel.

Dans la section suivante, nous évaluons les stratégies minimax et minisum et les comparons aux stratégies SC et HPCC.

4 Évaluation expérimentale

Pour expérimenter nos stratégies, nous avons utilisé le simulateur Simpatrol, lancé par le CIn, UFPE (Recife, Brazil)([6]). Nous avons choisi comme environnement les cartes décrites dans [4] et utilisées à de nombreuses reprises dans la littérature. Elles contiennent une carte aléatoire fortement connectée (50 nœuds, 106 arêtes) : la carte A, une carte aléatoire faiblement connectée (50/69) : la carte B, un cercle (50/50), un corridor (49/50), une grille (50/90) et une carte de 9 îles fortement connectées à l'intérieur et faiblement interconnectées (50/84). Sur chaque carte, nous avons testé les diverses stratégies en système fermé (30 expériences pour chaque nombre d'agents) pour mesurer les performances en phase stable. Nous avons ensuite fait de longues expériences (20000 cycles) en système ouvert dans lesquelles la taille de la population varie entre 2 et 13 agents. Ces expériences nous permettent de mieux comprendre le fonctionnement des stratégies sur le long terme.

Pour chacune de ces expériences, l'allocation originale des nœuds pour FBA, Minimax et Minisum est uniforme entre les agents et aléatoire. Pour comparer les performances moyennes des stratégies, nous avons normalisé les mesures sur chaque carte et pour chaque taille de population en multipliant les performances par le nombre d'agents dans le système. Cette normalisation nous amène au premier résultat obtenu, qui est que toutes les stratégies montrent des performances quasi-linéaires pour I_{av} et le MSI lorsque le nombre d'agents évolue, et ce sur toutes les cartes. Les écarts-types constatés sont inférieurs à 5% pour I_{av} (Interval Moyen) et à 20% pour le MSI (Mean Square Interval). Ce résultat nous amène à considérer que chaque stratégie peut donc être représentée de manière pertinente par la moyenne de ses normalisations sur les différentes tailles de population. Nous utilisons ces moyennes dans la suite de l'article.
Un autre résultat est que I_{av} n'est pas un critère discriminant pour comparer les stratégies. En effet, sur chaque carte et pour toutes les tailles de population, HPCC présente les meilleures performances sur I_{av}, suivie de près par Minisum, mais les autres stratégies sont moins de 20% derrière (au dessus) d'HPCC, parfois même à moins de 2% de différence (carte en cercle, pour toutes les tailles de population). En moyenne, c'est la stratégie SC qui présente les pires performances sur ce critère.
Le MSI est en revanche plus discriminant (voir Fig. 2). Le Single Cycle présente le MSI le plus faible sur toutes les cartes, suivi de près par HPCC. Minisum et Minimax présente des performances similaires sur ce critère, et FBA présente les pires performances sur toutes les cartes.
Le but principal de Minimax étant de minimiser le chemin le plus long de la société d'agents, il est naturel de mentionner le critère de l'intervalle maximal (I_{max}). Avec le même calcul de moyenne, on peut voir dans la Fig. 3 les performances des diverses stratégies sur ce critère. Ainsi que l'a démontré [1], la stratégie SC est optimale sur ce critère, et donc présente les meilleures performances sur toutes les cartes. Cependant, Minimax présentent des performances particulièrement bonnes et obtient la 2nde place sur 4 cartes, et une 3e place disputée sur les 2 autres. Ainsi que nous le prédisions dans la section 3.1, FBA obtient des performances particulièrement mauvaises sur ce critère.
Nous avons créé les expériences longues de la manière suivante : le système commence avec 5 agents. Tous les 100 cycles, il y a une probabilité de 10% qu'un événement arrive. Un événement concerne 1 à 4 agents (avec une probabilité décroissante) et déclenche l'entrée ou la sortie du système de ces agents. La probabilité du type d'événement dépend du nombre d'agents dans le système : plus il y a d'agents, plus l'événement à de chances d'être une sortie. Nous avons choisi de limiter la taille de la population à un maximum de 13 agents, soit $\sim$1/4 du nombre de nœuds des cartes. Pour chaque carte, nous avons fait 15 expériences avec des positions et des allocations de départ aléatoires.
Ces expériences nous permettent d'observer le comportement des diverses stratégies dans un environnement complexe et changeant. Une première observation est que la stratégie FBA se comporte mieux dans les expériences longues en système ouvert que dans les expériences en système fermé. Cependant, il n'en reste pas moins derrière les autres stratégies sur les critères MSI et I_{max}. Nous discutons ce résultat dans la section 5.
Sur les critères concernant les phases de transition, nous pouvons faire les observations suivantes :

– temps de stabilisation : HPCC est le plus rapide à s'adapter à un événement (en se stabilisant en moyenne en 132 cycles), suivi par Minisum (152), FBA (170), Minimax (173) et pour finir SC (205) ;
– Amplitude de variations : FBA est la stratégie offrant le moins de pertes de performances avec un coefficient $log(Max/Moy)$ de 0,013 (soit 3,4% de pertes), suivi par Minimax et HPCC (0,016/5%), Minisum (0,023/7%) et enfin SC (0,24/85%).

Il est enfin intéressant de noter que pour 1 message envoyé dans la stratégie SC, une moyenne de 82 messages sont envoyés pour la stratégie HPCC, 274 pour FBA, 277 pour Minimax et 264 pour Minisum.

5 Discussion

Ces résultats nous offrent une image complète du fonctionnement des diverses stratégies. La stratégie Single Cycle est optimisée pour visiter chaque nœud avec la même régularité, ce qui lui permet d'obtenir les meilleures performances sur les critères MSI et I_{max}. En revanche, elle n'est pas optimisée pour I_{av} et est extrêmement coûteuse à reconfigurer, tant en temps qu'en performances.
Ainsi que [9] l'explique, HPCC est une stratégie locale, au sens ou chaque agent va tendre à rester dans une région du graphe et à visiter sou-

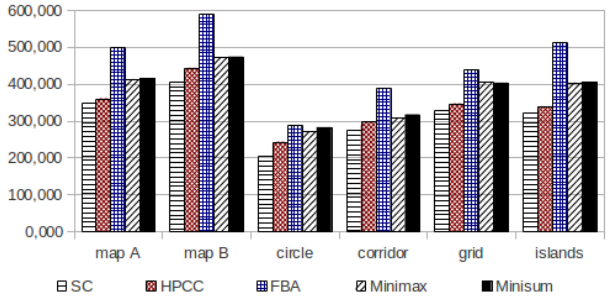

FIGURE 2 – MSI normalisé des différentes stratégies, par carte.

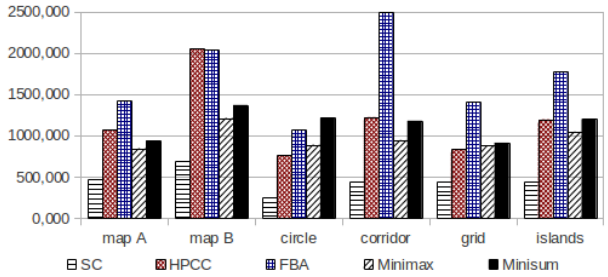

FIGURE 3 – Intervalle Max des différentes stratégies, par carte.

vent les nœuds qui l'entourent. Ceci permet à la fois de bonnes performances sur le MSI et I_{max} en évitant les "déplacements longue-distance", et sur I_{av} car certains nœuds sont visités extrêmement souvent. Étant centralisée, la stratégie HPCC réagit très rapidement aux changements de la population d'agents. Cependant, son caractère local devient alors désavantageux, puisque les agents ne vont pas aller directement vers leur nouvel emplacement mais vont multiplier les visites sur le trajet. Ceci explique la perte de performances décrite.

Les résultats confirment nos prédictions sur l'utilisation de communications asynchrones pour la stratégie FBA. Cette stratégie obtient de bonnes performances sur le critère I_{av}, de mauvaises sur le MSI et de pires sur I_{max}. Ces performances sont dues à une mauvaise allocation des nœuds, dans laquelle certains agents ont peu de nœuds qu'ils visitent donc très souvent (d'où le bon I_{av}) et n'ont aucune incitation à aider et décharger les agents plus désavantagés par leur allocation (d'où les mauvais MSI et I_{max}). C'est sur ce comportement que les expériences en système ouvert ont un effet bénéfique, en forçant régulièrement les agents à redistribuer les nœuds.

Pour les stratégies à base d'enchères, la perte de performances peut être vue comme le gain entre la ré-allocation due à l'événement et l'allocation atteinte après les enchères de réorganisation. Alors, la faible perte de performances constatée pour le FBA s'explique par le fait que, les performances globales étant mauvaises, il est difficile d'atteindre une allocation vraiment pire que celle en cours, et que par ailleurs la réorganisation n'améliore pas vraiment l'allocation due à l'événement. Il est intéressant de noter que pour FBA comme pour Minimax, les mécanismes de proximité ont été utilisés pour gérer les entrées/sorties (voir section 3.3). En effet, ils permettent à FBA d'éviter des ré-allocations dans lesquelles un nœud est réalloué à un agent situé très loin, mais dont aucun agent ne veut par la suite car il n'est pas intéressant pour lui.

La stratégie Minimax montre de très bon résultats : minimiser I_{max} permet d'assurer un bon MSI (car aucun agent n'a un mauvais ensemble de nœuds, donc tous les nœuds sont visités régulièrement), ce qui à son tour assure un bon I_{av}. En revanche, tout changement dans la population déclenche un long processus de réallocation (commun à toutes les stratégies à base d'enchères). On voit cependant que la perte de performances est plus élevée que pour la stratégie FBA, car la ré-allocation est plus efficace. Les mécanismes de transition par proximité ont été choisis car ils présentent les meilleures performances des mécanismes testés.

Enfin, la stratégie Minisum montre des performances similaires à la stratégie Minimax, à quelques différences près. Choisir de minimiser la moyenne de la longueur des chemins plutôt que la longueur maximale permet d'obtenir un I_{av} un peu meilleur mais un I_{max} un peu moins bon. En effet, un agent peut avoir un chemin un peu plus long que la moyenne, mais aucun autre agent ne peut le décharger à un coût inférieur à son gain. Ceci explique l'I_{max} supérieur au Minimax. En revanche, si certains agents sont au dessus de la moyenne, d'autres sont en dessous, et visitent leurs nœuds très souvent, d'où un meilleur I_{av}. En ce qui concerne les performances en transition, le temps de transition est très similaire à ceux de Minimax et FBA. Par contre, l'amplitude de variation est plus importante que pour Minimax. Ceci s'explique par l'utilisation de mécanismes de transition assurant une allocation plus mélangée lors des variations de population, permettant d'éviter d'avoir des optima locaux mais présentant des performances moindres (voir section 3.3). La réallocation des nœuds est donc aussi efficace que pour Minimax, mais part d'une allocation plus défavorable.

En résumé, les stratégies Minimax et Minisum permettent toutes les 2 d'atteindre une alloca-

tion efficace des nœuds, présentant de bons résultats sur l'ensemble des critères de performances. Minimax est particulièrement optimisé pour le critère I_{max}, sur lequel il bat la stratégie HPCC, tandis que Minisum propose un compromis entre I_{av} et MSI.

6 Conclusion

Dans cet article, nous avons présenté deux stratégies pour la tâche de la patrouille temporelle multi-agents, dans un système ouvert et à communications asynchrones. Ces stratégies sont basées sur l'utilisation d'enchères, sont complètement décentralisées et s'inspirent de la théorie du choix social : ce sont les stratégies Minimax et Minisum. Nous avons décrit comment les agents peuvent participer aux enchères en cours et allouer les enchères qu'ils ont initiées de manière à optimiser le bien-être social choisi, égalitaire pour Minimax et utilitaire pour Minisum. Nous avons ensuite évalué ces stratégies en système fermé et en système ouvert. Les résultats obtenus montrent que ces stratégies obtiennent de meilleures performances que la stratégie FBA, qui est une référence actuelle sur cette tâche et est elle aussi basée sur l'utilisation d'enchères. Ils montrent aussi que ces stratégies peuvent atteindre un niveau de performances équivalent aux stratégies centralisées de référence, tout en évitant les inconvénients de la centralisation. Notre étude montre que les performances atteintes sont quasi-linéaires dans le nombre d'agents du système, ce qui est très prometteur pour un futur passage à l'échelle.
Plusieurs pistes de recherche s'ouvrent pour de futurs travaux : étudier l'impact d'une limitation des communications sur les performances des diverses stratégies, étudier le comportement des diverses stratégies lors d'un passage à l'échelle de la tâche (plusieurs centaines de nœuds, plusieurs dizaines d'agents). Une autre piste intéressante est l'étude de la patrouille lorsque le graphe est dynamique. Un problème de la patrouille ayant un environnement et une population dynamiques est en effet très proche d'application réelle de recherche de survivants après une catastrophe naturelle, par exemple.

Références

[1] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proc. of the Intelligent Agent Technology, IEEE/WIC/ACM Int. Conf.*, 2004.

[2] Y. Chevaleyre, U. Endriss, J. Lang, and N. Maudet. A short introduction to computational social choice. *SOFSEM 2007 : Theory and Practice of Computer Science*, 2007.

[3] M. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain. Auction-based multi-robot routing. In *Robotics : Science and Systems*, 2005.

[4] A. Machado, G. Ramalho, J.D. Zucker, and A. Drogoul. Multi-agent patrolling : An empirical analysis of alternative architectures. *Lecture notes in computer science*, 2003.

[5] T. Menezes, P. Tedesco, and G. Ramalho. Negotiator agents for the patrolling task. *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, 2006.

[6] D. Moreira, G. Ramalho, and P. Tedesco. Simpatrol - towards the establishment of multi-agent patrolling as a benchmark for multi-agent systems. In *ICAART*, 2009.

[7] Antoine Nongaillard and Philippe Mathieu. Allocation de ressources et maximisation de bien-être social. In *Actes des 5e Journées Francophones sur les Modèles Formels de l'Interaction (MFI'09)*, 2009.

[8] C. Poulet, V. Corruble, and A. El Fallah Seghrouchni. Auction-based strategies for the open-system patrolling task. In *PRIMA - 15th Int. Conf. on Principles and Practice of Multi-Agent Systems, Proc.*, Lecture Notes in Computer Science. Springer, 2012. to be published.

[9] C. Poulet, V. Corruble, A.E.F. Seghrouchni, and G. Ramalho. The open system setting in timed multiagent patrolling. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2011 IEEE/WIC/ACM Int. Conf.*, IEEE, 2011.

[10] P.A. Sampaio, G. Ramalho, and P. Tedesco. The gravitational strategy for the timed patrolling. In *Tools with Artificial Intelligence (ICTAI), 2010 22nd IEEE Int. Conf.*, IEEE, 2010.

[11] T. Sandholm. Contract types for satisficing task allocation. In *Proc. of the AAAI spring symposium : Satisficing models*, 1998.

[12] H. Santana, G. Ramalho, V. Corruble, and B. Ratitch. Multi-agent patrolling with reinforcement learning. In *AAMAS-2004, Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*, 2004.