Étude du Problème de la Patrouille Multi-Agents en Système Ouvert

Cyril Poulet

Université Pierre & Marie Curie, Laboratoire d'Informatique de Paris VI
Paris, France
cyril.poulet@lip6.fr

Résumé : De nombreuses stratégies ont été proposées ces dernières années pour le problème de la patrouille multi-agents. Elles se basent toutes sur l'hypothèse d'un système fermé (population d'agents constante, aucun agent n'entre ou sort du système). Cette hypothèse est forte et limite l'applicabilité des modèles de patrouille multi-agents. Dans cet article, nous proposons de reprendre certaines des techniques proposées dans la littérature pour les adapter à une hypothèse de système ouvert, et de comparer leurs performances sur un scénario simple dans lequel un agent décide de quitter la tâche de la patrouille. **Mots-clés** : patrouille, multi-agents, système ouvert

1. Introduction

Le problème de la patrouille est la modélisation formelle de la situation dans laquelle une zone géographique contenant de multiples points d'intérêt doit être surveillée le plus étroitement possible par une équipe d'agents, humains ou artificiels. Deux variations de ce problème ont été étudiées ces dernières années : la *patrouille anti-intrusion* dans laquelle la zone doit être protégée contre des adversaires cherchant à s'introduire dedans, et la *patrouille temporelle* dans laquelle la tâche est de visiter chaque point d'intérêt aussi fréquemment et régulièrement que possible. Cet objectif est mesuré par des critères à optimiser de distribution temporelle des visites.
Ce problème est un bon cas d'étude pour la coordination dans les systèmes multi-agents : il est suffisamment simple pour être facilement implémenté, et

Financé dans le cadre du projet TerraDynamica (programme FUI8, 2010-2012)

de nombreuses stratégies de coordination peuvent être comparées expérimentalement grâce aux métriques retenues. Il permet cependant de modéliser une grande variété de situations tant réelles qu'artificielles. Pour toutes ces raisons, ce problème a été proposé comme benchmark pour les systèmes multi-agents dans Moreira *et al.* (2009).
L'exemple qui suit fut la base de notre réflexion. Une catastrophe se produit (tremblement de terre ou tsunami, par exemple), et les victimes doivent être détectées et sauvées. Une équipe d'agents est donc déployée sur la zone pour chercher les victimes dans les décombres. En modélisant le problème comme une tâche de patrouille, comment gérer la découverte d'une victime nécessitant d'être dégagée et soignée ? Avec la modélisation existante, la stratégie naturelle était que l'agent signale la victime à une autre équipe de sauveteurs, puis reprends sa tâche en cours. Cependant cette stratégie est sous-optimale ici. L'agent doit donc être capable de s'arrêter de patrouiller et commencer les opérations de sauvetage, pendant que le reste de l'équipe patrouillant se réorganise pour ré-optimiser la fréquence des visites partout sur la zone.
Nous avons donc décidé d'étudier le problème de la patrouille dans un système ouvert, dans lequel les agents peuvent rejoindre ou quitter la tâche quand ils le veulent. Il nous a semblé naturel d'adapter et comparer certaines des stratégies proposées dans divers travaux pour la patrouille en système fermé. Cependant, nous nous intéressons ici particulièrement aux phases de transition, durant desquelles le système multi-agents doit se réadapter à son nouvel état (des agents en plus ou en moins), car les phases stable (nombre d'agents fixe) ont été beaucoup étudiés dans les travaux évoqués.
Dans cet article, nous commençons par définir précisément la tâche de la patrouille dans la section 2, et exposons l'état de l'art dans la section 3. La section 4 présente les agents adaptés que nous allons comparer, ainsi que les nouveaux critères d'évaluation associés au problème en société ouverte. La section 5 présente enfin les résultats expérimentaux obtenus, ainsi que l'interprétation de ces résultats.

2. La tâche de la patrouille

Le problème de la patrouille utilise des environnements représentés en *graphes* dont les nœuds sont les points d'intérêt qui doivent être visités et les arêtes sont les chemins que les agents peuvent emprunter pour se rendre d'un point à un autre. Ces arêtes ont une longueur qui représente la distance entre les points, ou comme dans nos simulations le nombre de cycles néces-

saires pour la parcourir.
Nous n'utilisons dans cet article que des environnements stables et non évolutifs, malgré l'intérêt que peuvent présenter des graphes évoluant au cours du temps. De plus, nous considérons que la patrouille et le sauvetage sont deux tâches séparées, et que la seconde n'impacte pas la première (en particulier au niveau du graphe). Nous ne priorisons pas non plus les nœuds, considérant que tous les points à patrouiller présentent le même niveau d'intérêt (patrouille uniforme). Enfin, les communications entre les agents ne sont pas restreintes : les agents peuvent communiquer aussi loin et souvent que nécessaire.
La tâche qui nous intéresse ici est la tâche de la patrouille temporelle, telle que décrite dans Machado *et al.* (2003) : les agents doivent visiter de manière répétitive l'ensemble des nœuds du graphe, dans le but de minimiser des métriques basées sur la distribution temporelle des visites (régularité, fréquence, etc.).
L'ouverture du système implique que le nombre d'agents peut changer au cours du temps. Dans ce cadre, la façon dont les agents prennent en compte ces changement est, à nos yeux, une facette importante du problème de la patrouille. Dans cette optique, nous proposons que minimiser les variations des métriques pendant les phases de transition et limiter la durée de ces phases font parti des buts à atteindre.

3. Etat de l'Art

3.1. Critères d'évaluation

Le premier critère d'évaluation proposé (Machado *et al.* (2003)) fut l'*oisiveté*, qui est le temps depuis lequel un nœud n'a pas été visité. Les métriques qui y sont attachées sont : l'*oisiveté instantanée d'un nœud* (temps ou nombre de cycles depuis la dernière visite de ce nœud), l'*oisiveté instantanée du graphe* (moyenne des oisivetés instantanées des nœuds du graphe) et l'*oisiveté moyenne du graphe* (moyenne de l'oisiveté instantanée du graphe sur N cycles). Alors que l'oisiveté instantanée se comprend intuitivement, l'oisiveté moyenne du graphe est beaucoup moins aisée à comprendre, car la moyenne sur le temps et l'espace la rend difficile à relier à des interprétations simples telles que l'intervalle moyen entre les visites. Pour cette raison, l'*intervalle* est proposé dans Sampaio *et al.* (2010) comme nouveau critère d'évaluation, en exploitant l'intervalle moyen entre les visites, la variance des intervalles et l'in-

tervalle maximum pour comparer les différentes stratégies. La mesure retenue dans ces travaux est l'*intervalle quadratique moyen* (appellé *Mean Square interval*, MSI).

3.2. Stratégies proposées

3.2.1. Descriptions des différents agents

De nombreuses stratégies ont été proposées ces dix dernières années. Machado *et al.* (2003) ont, les premiers, proposé une étude comparative d'agents, en les divisant entre réactifs et cognitifs, communicants ou non, avec un coordinateur central ou non, et par stratégies de choix de direction. On peut citer parmi les stratégies proposées le Random Reactive (RR), le Conscientious Reactive (CR), ou le Cognitive Coordinated (CC).
Almeida *et al.* (2003) prennent en compte le temps mis par l'agent pour atteindre un nœud en plus de l'oisiveté de celui-ci pour choisir vers quel nœud se diriger (heuristique), et calculent la somme estimée des oisivetés des nœuds intermédiaires pour choisir le chemin le plus intéressant vers le nœud-but (pathfinding). Ils comparent les nouvelles versions des agents (ex. l'HPCC) dans Machado *et al.* (2003).
La stratégie suivante apparait dans Santana *et al.* (2004), qui propose d'utiliser l'apprentissage par renforcement. On peut citer le Grey-Box Learner Agent (GBLA) qui connait le prochain nœud-but de chaque autre agent. Au même moment, Chevaleyre (2004)) propose l'agent Single-Cycle (SC) : un coordinateur calcule le cycle minimal dans le graphe, puis distribue uniformément les agents autour de ce cycle.
Almeida *et al.* (2004) et Menezes *et al.* (2006) proposent ensuite des agents basés sur des techniques d'enchères permettant de répartir les nœuds en zones attribuées chacune à un agent, tel que le Flexible Bidder Agent (FBA). Plus récemment, Sampaio *et al.* (2010) ont proposé une technique inspirée par la loi de la gravitation.

3.2.2. Résultats

Les résultats transparaissant de l'ensemble des travaux cités montrent que les agents cognitifs sont plus adaptés à la tâche que leurs homologues réactifs, et que plus les agents sont coordonnés, plus ils conviennent à la tâche.

Les résultats montrent aussi que les heuristique et pathfinding proposés améliorent tout le temps la stratégie à laquelle ils sont combinés. Enfin, l'ordre suivant apparait lorsqu'on s'intéresse au critère de l'oisiveté : $SC > FBA \simeq GBLA \simeq HPCC > CC > CR > RR$.

4. Adaptation du problème à l'ouverture du système

Le choix d'un système ouvert, dans lequel les agents peuvent rejoindre ou quitter la tâche, est une modification importante du problème original de la patrouille multi-agents. Dans ce nouveau problème, nous nous intéressons à comprendre comment les différentes stratégies peuvent être adaptées pour être capable de gérer le changement des agents dans le système, et comment ces adaptations se comporte en pratique : comment les performances varient-elles pendant les phases de transitions ? Quelle est la durée de ces phases ? Pour étudier ces questions, nous proposons d'abord de nouveaux critères de performance pour évaluer les stratégies, puis nous exposons les modifications que nous avons apporté à certains agents pour les adapter à cette nouvelle tâche.

4.1. Nouveaux critères

Pour étudier la durée des phases de transition, ou leur forme (c'est à dire la vitesse à laquelle les stratégies retrouvent la stabilité), nous proposons les métriques suivantes :

- *temps de stabilisation* : en combien de temps (en cycles) la simulation retrouve t-elle un état stable ? Pour cela, nous proposons de calculer la moyenne temporelle de l'un des critères au choix sur la phase stabilisée atteinte, puis d'y comparer des moyennes temporelles sur de courtes périodes de 100 cycles prises entre le moment où l'événement se produit (changement dans le nombre d'agent) et le moment où l'on atteint une différence de moins de 1% avec la valeur stable ;
- *courbes de stabilisation* : Comment les variations se répartissent-elles dans le temps ? Nous proposons de comparer la vitesse à laquelle les simulations atteignent 50% de la valeur stabilisée, 25%, 15%, etc.

En plus de ces nouvelles métriques, nous utiliserons des métriques basées sur les intervalles pour mesurer la robustesse des stratégies, et la force des variations.

4.2. Adaptation des agents

Dans cet article, nous nous sommes focalisés sur quelques-uns des agents proposés par l'état de l'art, dans l'intention de fournir des premiers résultats pouvant servir de cadre de référence pour des travaux futurs. Nous avons choisi les agents suivants :

- le *Random Reactive Agent (RR)* : Réactif, non communicant, il choisit le prochain nœud-but au hasard parmi les nœuds connectés à celui sur lequel il se trouve
- le *Conscientious Reactive Agent (CR)* : Réactif, non communicant, il choisit comme prochain nœud-but le nœud connecté à celui sur lequel il se trouve qui présente la plus haute oisiveté instantanée
- le *Cognitive Coordinated Agent (CC)* : Cognitif, communicant avec un coordinateur global qui choisit pour lui le prochain nœud-but : le nœud du graphe présentant la plus grande oisiveté instantanée
- le *Heuristic Pathfinding Cognitive Coordinated Agent (HPCC)* : même agent que le précédent, mais le coordinateur utilise l'heuristique décrite dans Almeida *et al.* (2003) et l'agent utilise le pathfinding du même article
- le *Single Cycle (SC)* : Le plus court cycle passant par tous les nœuds est calculé, puis les agents sont distribués uniformément le long de ce cycle. Pour cela, un agent est démarré, puis chaque agent part un certain nombre de cycles après avoir détecté le passage de cet agent sur le nœud où il est lui-même situé. Ce nombre de cycles augmente pour chaque agent doublé.

Nous avons choisi ces agents pour diverses raisons. Les RR et SC sont reconnus respectivement comme pire et meilleure stratégies en système fermé, nous les avons donc choisies comme références. CR est un agent réactif simple, mais guidé par l'oisiveté, c'est donc une stratégie intéressante pour voir comment les agents réactifs se comportent en système ouvert. CC est un agent coordonné simple, permettant de voir comment un coordinateur peut gérer des agents entrant et sortant. Enfin, HPCC nous permettra de vérifier si les techniques d'heuristique et de pathfinding procurent toujours un avantage en système ouvert.

Nous n'avons pas eu besoin de modifier les agents réactifs, puisqu'ils ne sont pas conscient des autres agents du système. Ils ne sont donc pas concernés par le changement du nombre d'agents dans le système.

Dans les stratégies à coordinateur, seul celui-ci a été modifié : il doit pouvoir

détecter que le nombre d'agents à changé. Si un agent rejoint la tâche, le coordinateur lui attribue un nœud-but lorsque celui-ci le lui demande. Si un agent quitte la tâche, le coordinateur le détecte et enlève le nœud-but associé de la liste des buts courant, permettant ainsi à ce nœud d'être ré-alloué à un autre agent par la suite.
L'adaptation du SC a soulevé plus de questions : le cycle n'a pas besoin d'être recalculé puisque le graphe n'a pas changé, mais les agents doivent être ré-répartis autour de celui-ci, sauf bien sur s'il ne reste qu'un agent. Nous avons distingué trois cas :

- un agent quitte la tâche : le coordinateur stoppe les agents, puis fait attendre chaque agent un temps incrémental avant de repartir, en commençant par l'agent situé juste après l'agent qui a quitté la tâche. Ainsi, on comble le trou laissé. Nous avons appelé cette méthode la stratégie *Fast*
- un agent entre dans le système : le coordinateur stoppe les agents, puis calcule la nouvelle distance entre les agents. Il les redémarre en utilisant la méthode présentée en 4.2. (un agent est utilisé comme référence par les autres). Nous avons fait deux versions de cet algorithme : l'une (appelée *slow 2*) dans laquelle les agents repartent l'un après l'autre, ce qui entraine une phase de transition plus longue mais moins violente, et l'autre (*slow 1*) dans laquelle tous les agents repartent en même temps, ce qui permet une transition moins longue mais plus violente.
- un agent entre au même moment qu'un agent sort : le coordinateur le place à la position laissée par l'agent qui sort. Si N agents entrent et sortent (changement d'équipe par exemple), le coordinateur remplit chaque position vacante avec l'agent libre le plus proche.

5. Resultats expérimentaux et discussion

5.1. Cadre des simulations

Nous avons utilisé le simulateur Simpatrol, lancé par le CIn de l'UFPE (Recife, Brazil) (voir Moreira *et al.* (2009)). Nous avons utilisé la carte A (Fig 1) comme environnement, décrite dans Machado *et al.* (2003). Elle présente 50 nœuds et 106 arêtes. Pour chaque taille de population d'agents dans le système au début de la simulation, 30 départs au hasard ont été calculés. Les simulations durent alors 3000 cycles.
Pour ce premier article, nous nous sommes concentré sur un scénario simple : au cycle 1000, un agent quitte le système. Pour N agents (N parmi 5, 10, 15 et

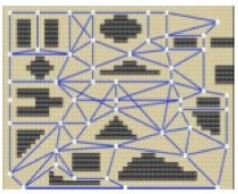

FIGURE 1: Carte A

25), nous avons fait 30 simulations pour une population stable de N-1 agents, 30 pour N agents, et 30 pour N agents au début et N-1 à partir du cycle 1000. Nous avons ensuite analysé les phases de transition (qui se terminent avant le cycle 2000) et les phases stables (cycle 2000 à 3000).

5.2. Résultats

En premier résultat, En notant $Intervals\_of\_node\_n$ l'ensemble des intervalles du nœud n, I_n la longueur d'un intervalle du nœud n, et $\mid Intervals\_on\_graph \mid$ le cardinal de l'ensemble des intervalles du graphe, le MSI s'écrit :

$$MSI = \sqrt{\frac{\sum_{\{Nodes \in graph\}} \sum_{\{Intervals\_of\_node\_n\}} I_n^2}{\mid Intervals\_on\_graph \mid}}$$

Avec en plus Nb_{nodes} le nombre de nœuds du graphe, et Nb_{cycles} la longueur en cycles de la simulation, l'intervalle moyen I_{av} s'écrit :

$$I_{av} = \frac{\sum_{\{Nodes \in graph\}} \sum_{\{Intervals\_of\_node\_n\}} I_n}{\mid Intervals\_on\_graph \mid} = \frac{Nb_{nodes} \times Nb_{cycles}}{\mid Intervals\_on\_graph \mid}$$

Alors on peut démontrer que l'oisiveté moyenne du graphe Id_{av} s'écrit :

$$Id_{av} = \frac{MSI^2}{2I_{av}} - \frac{1}{2}$$

Pour cette raison, dans la suite de cet article nous utiliserons l'intervalle moyen et le MSI, que nous pensons complémentaire : le premier renseigne sur la fréquence globale des visites, le second sur la régularité de celles-ci.
Cependant, l'oisiveté instantanée du graphe reste une métrique utile pour évaluer l'évolution du système au cours du temps, car c'est une moyenne sur

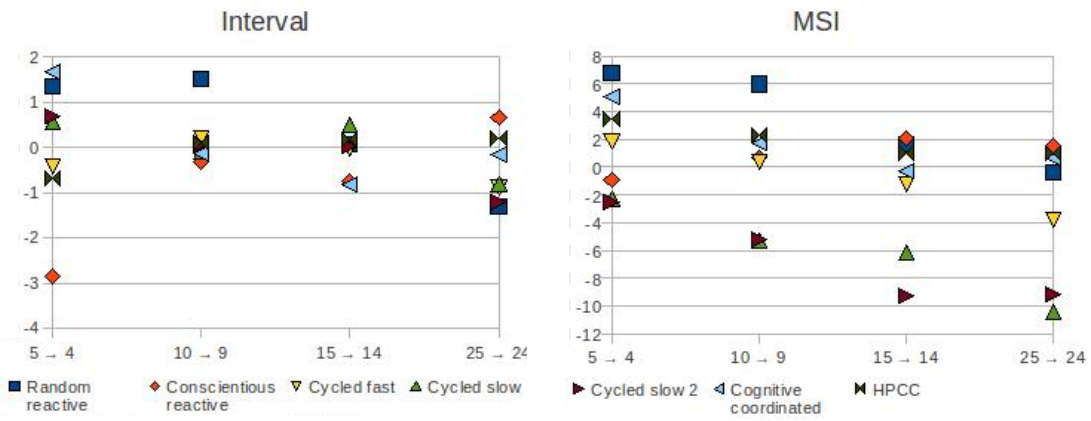

FIGURE 2: Intervalle moyen et MSI (gain en %) en phase stable.

l'ensemble des nœuds du nombre de cycles depuis la dernière visite. C'est pourquoi nous avons choisi de l'utiliser pour certaines figures de cet article.
Les métriques que nous avons utilisé pour les scénarios $5 \rightarrow 4$, $10 \rightarrow 9$, $15 \rightarrow 14$, et $25 \rightarrow 24$ sont les suivantes :

- Pour la phase stable, les gains sont calculés comme suit (avec $M^X_{a \rightarrow b}$ la valeur de la métrique pour X agents moyennée entre les cycles a et b) :

$$g^M_{2000 \rightarrow 3000} = \frac{M^{N-1}_{2000 \rightarrow 3000} - M^{N \rightarrow N-1}_{2000 \rightarrow 3000}}{M^{N-1}_{2000 \rightarrow 3000}} \times 100$$

  Ce gain se lit : "Les agents modifiés ont obtenus des performances $g^M\%$ meilleures que les agents non modifiés sur la même période de stabilité, sur la métrique M". Si le gain est nul, les agents modifiés présentent exactement les mêmes performances que les agents non modifiés.
- pour la phase de transition, le gain sur l'oisiveté instantanée du graphe compare la plus haute valeur de celle-ci pendant la phase de transition à l'oisiveté moyenne du graphe sur la phase stabilisée suivante. Le temps de stabilisation est le nombre de cycle nécessaire pour atteindre la valeur stable à 1% près.

Grâce à ces métriques, on peut voir sur les figures 2 que les stratégies adaptées ont des performances statistiquement similaires aux stratégies non modifiées sur les phases de stabilité, à nombre d'agents équivalent. Cela montre que les adaptations sont correctes, et que ces stratégies adaptées sont capables de passer d'une phase stable à une autre et de présenter dans ces phases des performances aussi bonnes qu'en système fermé.

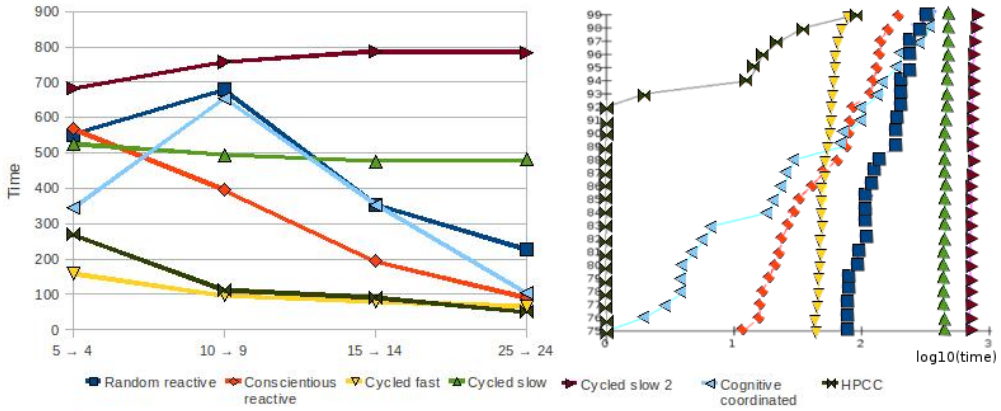

FIGURE 3: temps de stabilisation (gauche), courbes de stabilisation pour le scénario $15 \rightarrow 14$ (droite)

On peut cependant noter que les deux solutions SC slow montrent des gains MSI négatifs marqués, avec une chute de 6 à 10% en régularité selon la taille du système. Ce phénomène s'explique par un court délai de 1 ou 2 cycles qui peut apparaître quand les agents reçoivent de nouvelles instructions. Ces délais sont courts (ils n'apparaissent pas dans le gain d'intervalle moyen) mais plus le nombre d'agents est grand, plus les intervalles se réduisent et plus ces délais deviennent significatifs.

Etudions maintenant la phase de transition elle-même. Le temps de stabilisation est visible dans la figure 3 gauche. On peut en extraire les résultats suivants, sur lesquels nous reviendrons par la suite :

- En termes de rapidité, on a SC fast et HPCC, suivi du CR, puis du CC et du RR, et enfin par les deux SC slow ;
- Les SC fast et slow 1 s'améliorent légèrement avec le nombre d'agents, alors que le SC slow 2 devient plus lent à se stabiliser ;
- Les variations des SC (fast et slow), HPCC et CR sont proportionnelles (ou inversement proportionnelles pour le SC slow 2) au nombre d'agents, alors que les RR et CR empirent jusqu'à atteindre suffisamment d'agents pour repartir à la hausse.

Sur la forme elle-même de la phase de transition, les figures 4 et 5 représentent l'oisiveté instantanée du graphe pour les différentes stratégies sur le scénario $15 \rightarrow 14$. Ces formes sont représentatives des stratégies, la différence entre les scénarios tient au niveau général de ces courbes : moins il y a d'agents,

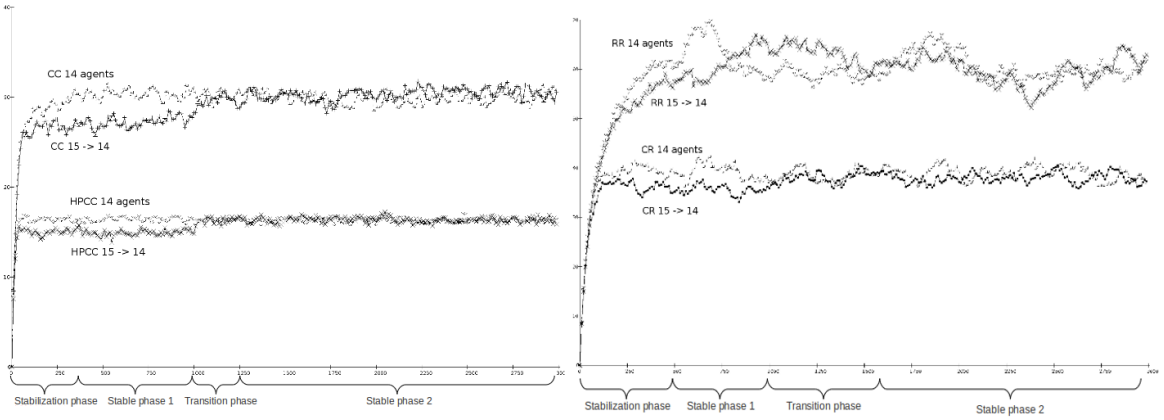

FIGURE 4: oisiveté instantanée RR/CR et CC/HPCC
gauche : RR 14 agents, RR $15 \rightarrow 14$, CR 14 agents et CR $15 \rightarrow 14$
droite : CC 14 agents, CC $15 \rightarrow 14$, HPCC 14 agents et HPCC $15 \rightarrow 14$

plus les courbes sont hautes sur le graphe (oisiveté instantanée plus grande), mais l'ordre et la forme de change pas. Dans ces figures, une division verticale représente 10 cycles, ce qui nous permet de comparer les stratégies. On constate tout d'abord que sur les phases stables, on a l'ordre SC > HPCC > CC > CR > RR qui apparait, comme prévu. Pour les formes elles-mêmes, on peut noter :

- RR ne semble pas présenter de transition : le hasard des décisions rend l'oisiveté instantanée du graphe impossible à prédire à un instant précis. C'est uniquement en calculant des moyennes temporelles que l'on peut voir que l'intervalle moyen a augmenté après la transition.
- CR présente une transition lisse de N à N-1 agents : l'oisiveté instantanée augmente progressivement vers la nouvelle valeur ;
- CC et HPCC présentent un saut visible d'une phase à l'autre : la transition est courte (plus courte pour HPCC que pour CC) mais sans temps perdu : il n'y a pas de pic plus haut que la valeur de la seconde phase ;
- Enfin, les SC montrent le comportement prévu à la section 4.2. : la stratégie *fast* montre un petit pic du au fait que les agents sont arrêtés pour des raisons de coordination (mais pourrait disparaître si l'on ne les arrête pas) ; la stratégie *slow 2* exhibe une forme très similaire à la première phase de transition, qui utilise le même algorithme de répartition ; enfin, *slow 1* présente un pic plus haut mais plus court temporellement,

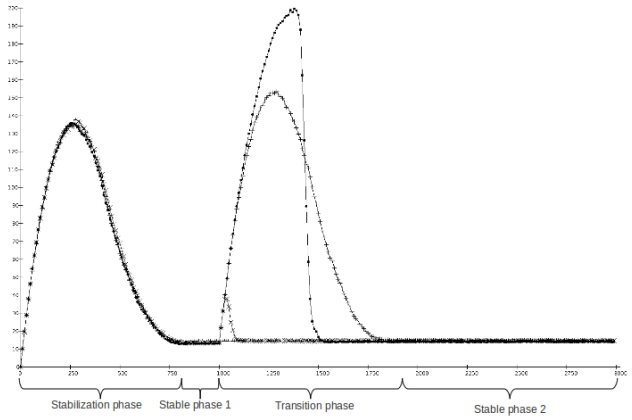

FIGURE 5: oisiveté instantanée. Du pic le plus haut à la courbe plate : SC slow 1, SC slow 2, SC fast, SC 14 agents

avec une forte pente verticale à la fin, quand tous les agents repartent en même temps.

Nous pouvons approfondir l'étude de la transition par la figure 3 droite, qui représente la vitesse à laquelle les stratégies se stabilisent (% de la valeur stable en fonction du log10 du temps). Comme pour les figures précédentes, nous n'incluons que le scénario $15 \rightarrow 14$, mais les courbes sont représentatives des stratégies indépendamment du scénario. Nous pouvons voir sur la figure 3 droite que :

- L'ordre suivant est confirmé : SC fast > HPCC > CR > CC > RR > SC slow 1 > SC slow 2 (avec de faibles variations selon le scenario, voir Fig. 3) ;
- les stratégies SC tombent de 75% à 99% de la valeur stable presque instantanément, mais tard, ce qui signifie que l'effort de réorganisation est fait avant, et qu'une fois fait, les stratégies retrouvent très vite les performances stabilisées ;
- les CC et HPCC atteignent 75% plus rapidement que les CR et RR mais prennent ensuite plus de temps pour se stabiliser complètement.

5.3. Discussion

Nous avons montré que l'on peut distinguer clairement deux groupes de stratégies : l'un possédant un ordre de préférence clair en terme de valeurs stabilisées et de temps de transition (CC > CR > RR), et l'autre dont les stratégies

sont meilleures que celles du premier groupe mais où le choix d'une stratégie va dépendre directement des caractéristiques recherchées pour le système. Si l'on recherche une transition lisse et courte, au prix de performances globales légèrement moindres, alors HPCC est la stratégies désignée. Si une perte de performance est permise pendant la transition, alors SC fast est le meilleur choix.
Toutes ces stratégies adaptées sont aussi valables pour des scénarios de type N $\rightarrow$ N+1, sauf le SC fast qui ne fonctionne que pour la perte d'agents car les agents ont une vitesse stable et ne peuvent accélérer pour réduire l'espace entre eux. Pour cette raison, étudier SC slow 1 et 2 nous permet un comportement comparatif de stratégies SC pour les scénarios N $\rightarrow$ N+1, bien qu'ils présentent peu d'intéret pour le scenario N $\rightarrow$ N-1.
Dans cet esprit, nous pouvons généraliser ces résultats à tout scénario impliquant la perte ou le gain d'un agent, et probablement de plus d'un agent :

- Si le système doit rester aussi stable que possible quelle que soit la phase, alors il parait raisonnable d'accepter une perte légère de performance pendant les phases stables et choisir la stratégie HPCC ;
- Si le système doit proposer les meilleures performances pendant les phases stables, mais peut prendre du temps et perdre des performances pour se réorganiser, alors les stratégies SC slow ou une combinaison SC fast/SC slow sont à choisir. Le choix entre slow 1 et slow 2 est alors un nouveau compromis entre vitesse et performance pendant la transition.

6. Conclusion

Dans cet article, nous avons présenté une nouvelle version du problème de la patrouille temporelle qui est pour nous un pas vers la modélisation de nouvelles situations et problèmes : la patrouille en système ouvert. Nous avons proposé de nouvelles métriques pour comparer les différentes stratégies sur ce nouveau problème, et adapté les cinq premières solutions historiquement proposées pour la patrouille en système fermé. Nous avons montré que parmi ces stratégies, les Heuristic Pathfinding Cognitive Coordinated et Single Cycle adaptés présentent de meilleures performances que les autres stratégies testées, et que choisir l'une ou l'autre dépend des caractéristiques que l'on souhaite pour le système.
Nous sommes très conscients que le travail présenté dans cet article est limité et préliminaire : un scenario simple, un seul graphe testé et des stratégies choisies autant pour leur intérêt historique que pour leurs performances. Ce-

pendant, comme nous l'avons montré à la section 5.3., nous pensons que ces résultats sont toujours valables dans le cas d'un agent entrant dans le système. Dans le futur, nous prévoyons d'étendre notre étude en adaptant des stratégies plus récentes et plus performantes. Une autre direction que nous souhaitons creuser est d'étudier des scénarios plus complexes. Enfin, nous souhaitons définir un cout d'entrée et de sortie de la patrouille en système ouvert, de manière à pouvoir interfacer cette tâche avec une architecture de décision plus globale.

Références

ALMEIDA A., CASTRO P., MENEZES T. & RAMALHO G. (2003). Combining idleness and distance to design heuristic agents for the patrolling task. In *II Brazilian Workshop in Games and Digital Entertainment*, p. 33–40.

ALMEIDA A., RAMALHO G., SANTANA H., TEDESCO P., MENEZES T., CORRUBLE V. & CHEVALEYRE Y. (2004). Recent advances on multi-agent patrolling. *Advances in Artificial Intelligence–SBIA 2004*, p. 126–138.

CHEVALEYRE Y. (2004). Theoretical analysis of the multi-agent patrolling problem. In *Proceedings of the Intelligent Agent Technology, IEEE/WIC/ACM International Conference*, p. 302–308.

MACHADO A., RAMALHO G., ZUCKER J. & DROGOUL A. (2003). Multi-agent patrolling : An empirical analysis of alternative architectures. *Lecture notes in computer science*, p. 155–170.

MENEZES T., TEDESCO P. & RAMALHO G. (2006). Negotiator agents for the patrolling task. *Advances in Artificial Intelligence-IBERAMIA-SBIA 2006*, p. 48–57.

MOREIRA D., RAMALHO G. & TEDESCO P. (2009). Simpatrol - towards the establishment of multi-agent patrolling as a benchmark for multi-agent systems. In *ICAART*, p. 570–575.

SAMPAIO G., RAMALHO G. & TEDESCO P. (2010). A technique inspired by the law of gravitation for the timed multi-agent patrolling. *22nd IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, p. 113–120.

SANTANA H., RAMALHO G., CORRUBLE V. & RATITCH B. (2004). Multi-agent patrolling with reinforcement learning. In *AAMAS-2004, Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*.