Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

Auto-organisation coopérative pour la conception de collectifs adaptatifs et robustes

Gauthier Picard and Marie-Pierre Gleizes

Laboratoire IRIT (CNRS - INP - UPS)
Université Paul Sabatier
118, route de Narbonne
F-31062 Toulouse Cedex
{picard,gleizes}@irit.fr

Résumé Cet article propose un modèle d'agent coopératif et souligne les bénéfices de l'usage de la coopération comme un moteur d'adaptation et de robustesse pour des systèmes multi-agents. Notre travail est basé sur l'approche des systèmes multi-agents adaptatifs (AMAS) qui considère la coopération comme un mécanisme auto-organisateur pour obtenir des comportements globaux émergents pour des systèmes plongés dans des environnements dynamiques. Une tâche de transport de ressources multi-robot illustre l'instanciation du modèle d'agent coopératif doté de règles comportementales réactives et anticipatives pour la coopération. Plusieurs expérimentations montrent la pertinence de cette approche dans des environnements difficiles, statiques ou dynamiques.

Mots-Clefs. Systèmes multi-agents ; Coopération ; Auto-organisation.

1 Introduction

Les phénomènes émergents sont bien connus des biologistes et des éthologues, qui les lient souvent aux mécanismes auto-organisateurs [1,2]. Les colonies de fourmis sont un exemple illustratif de collectifs auto-organisés capables de s'adapter très efficacement à des environnements complexes, alors que les comportements de micro-niveau (les fourmis) sont relativement faciles à modéliser, de manière stochastique [3]. Ces phénomènes ont directement inspiré les algorithmes fourmis pour le fourragement, par exemple [4].

Pour obtenir un comportement collectif auto-organisateur, les parties composants le système ont besoin d'un critère local de réorganisation afin de fournir, au final, une fonction plus adaptée aux demandes de l'environnement. Nous présupposons donc qu'à une organisation interne du système correspond une fonction produite par le système. Les concepteurs doivent spécifier ce critère et donner aux parties des capacités à manipuler ce critère. L'approche AMAS répond au premier point par l'utilisation d'une notion inspirée par la sociologie et l'éthologie, proche de la symbiose : la *coopération* entre agents autonomes, dotant les parties de capacités décisionnelles [5,6]. Ces agents coopératifs ont pour but d'éviter les *situations non coopératives* (NCS) comme une règle générique et proscriptive de réorganisation. L'avantage est d'assurer, en donnant aux

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

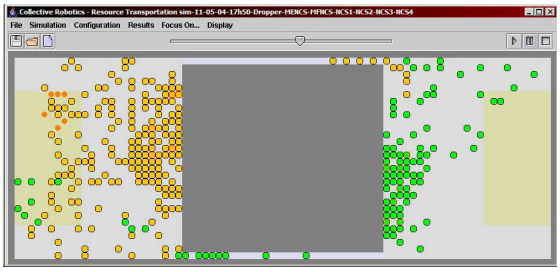

Fig. 1. Environnement du problème de transport de ressources et les différentes entités en jeu : les boîtes (disques sans bordure), les robots sans boîte (disques gris clair avec bordure), et les robots avec boîte (disques gris foncé avec bordure).

agents des comportements localement coopératifs, que le collectif fournira une fonction globale adéquate et adaptée. Pour concevoir de tels agents, un modèle d'agent coopératif est proposé dans la méthode ADELFE [7]. Pour mettre en avant l'utilisation de cette notion de coopération et du modèle d'agent coopératif, nous proposons de développer un système multi-robot devant effectuer une tâche de transport de ressources dans un environnement contraint. Dans ce problème, plusieurs robots similaires doivent transporter des boîtes d'une salle de retrait à une salle de dépôt, comme l'illustre la figure 1. Ces salles sont séparées par des couloirs étroits, produisant par conséquent des interférences spatiales. Sans extension, un comportement nominal et individualiste peut, très souvent, mener à des blocages.

Cet article se focalise principalement sur la capacité, pour le système multi-agent, à réagir à des changements environnementaux et sur l'observation de phénomènes émergents, plutôt que sur l'efficacité nominale de résolution de la tâche de transport. La section 2 présente les agents du système –les robots– en développant leurs différents modules et leur comportement nominal. La section 3 explique les différentes règles de coopération que les robots doivent respecter –comme un code de la route simpliste– pour atteindre leur but. Une plate-forme d'expérimentation a été développée pour tester l'instanciation du modèle d'agent coopératif. Plusieurs résultats sont présentés dans la section 4. Enfin, une discussion, dans la section 5 soulève des difficultés de conception avant de conclure dans la section 6.

2 Robots agents coopératifs

Le système multi-agent que nous proposons de modéliser pour illustrer l'approche AMAS est un collectif de robots. En utilisant la méthode ADELFE, les robots sont décrits par différents modules qui représentent un partitionnement de leurs capacités physiques, sociales et cognitives [7].

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

2.1 Les modules d'agent coopératif

Le *module de perceptions* représente les données entrantes des agents, provenant de l'environnement. Cet article se focalise uniquement sur l'adaptation à un problème de circulation, plus qu'à un problème de fourragement (la tâche des robots n'est pas de trouver des ressources). Ainsi la liste des perceptions pour un robot transporteur : la position de la zone de retrait, la position de la zone de dépôt, un cône de perception dans lequel les classes d'objets sont identifiables (robot, boîte ou mur), des capteurs de proximité (avant, arrière, droite et gauche), une boussole et la position absolue dans l'environnement. L'environnement est discrétisé en une grille dont les cases représentent des parties atomiques sur lesquelles peuvent se situer des robots, des boîtes ou des murs. Le module de perceptions définit aussi des valeurs limites pour les perceptions (par exemple, 5 cases dans les expérimentations présentées).

Le *module d'actions* représente les données sortantes (ou effecteurs) de l'agent vers son environnement. Les actions d'un robot transporteur sont : repos (*rest*), prendre (*pick*), déposer (*drop*), avancer (*forward*), reculer (*backward*), aller à gauche (*left*) et aller à droite (*right*). Les robots ne peuvent pas déposer leur boîte n'importe où dans l'environnement, mais uniquement sur la zone de dépôt. Ils ne peuvent pas non plus effectuer d'action communicative directe ou marquer physiquement leur environnement.

Le *module de compétences* contient les connaissances sur la tâche que l'agent doit remplir. Les compétences permettent aux robots d'atteindre leur but de transport. Par conséquent, un robot est capable de calculer quel objectif doit être rempli en fonction de son état actuel : s'il transporte une boîte, il doit aller à la zone de dépôt, sinon il doit atteindre la zone de retrait. En fonction de son état interne, le module de compétences propose une action à effectuer pour atteindre le but. Les buts des robots sont : atteindre la zone de retrait (*reach claim zone*) et atteindre la zone de dépôt (*reach laying zone*). De plus, les robots possèdent des caractéristiques physiques intrinsèques comme leur vitesse, le nombre de boîtes transportables en même temps ou des préférences sur les déplacements, comme, par exemple, préférer se déplacer vers l'avant plutôt que de faire demi-tour (comme les fourmis). De telles préférences sont appelées *valeurs réflexes*.

Le *module de représentations* contient les connaissances sur l'environnement (physique ou social). Les représentations qu'un robot peut manipuler sont très limitées. À partir de ses perceptions, il ne peut distinguer un robot d'un autre, mais peut savoir s'il transporte une boîte ou non. Il peut aussi mémoriser ses précédents but, position absolue, direction et action.

Le *module d'aptitudes* permet à un agent de choisir une action en fonction de ses perceptions, compétences et représentations. Concernant les robots, un choix de conception doit être établi à cette étape. En fonction du but courant, le module de compétences fournit des préférences sur chaque action que l'agent peut effectuer. Le module d'aptitudes choisit parmi ces actions quelle sera la prochaine action pour atteindre le but. Plusieurs politiques de décision peuvent être considérée ; par exemple, une politique arbitraire (celle ayant la plus grande préférence est choisie) ou un tirage Monte Carlo, qui est celle utilisée dans notre

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

exemple, étant donné qu'il a été appliqué avec succès sur des tâches de fourragement [8]. Par conséquent, le module d'aptitudes peut se résumer en une fonction de tirage Monte Carlo sur des vecteurs de préférences, qui attribue à chaque action une valeur de préférence entière, fournie par le module de compétences. De même, le *module de coopération* propose des préférences sur les actions pour résoudre les NCS et choisit l'une d'entre elles par tirage, si nécessaire. Ce dernier module est décrit dans les sections 2.5, puis 3.

2.2 Fonctionnement interne

Lors de la phase de perception, le module de perceptions met à jour les valeurs des entrées. Ces données peuvent impliquer des changements directs dans les compétences ou les représentations. Une fois les connaissances mises à jour, la phase de décision doit conduire au choix d'une action. Durant cette phase, les modules d'aptitudes et de coopération calculent en parallèle. Le premier doit fournir une action correspondant à un comportement nominal. Le second doit détecter si l'agent est en situation non coopérative et s'il doit agir. Dans ce cas, l'action coopérative subsume l'action nominale. Si aucune action coopérative n'est proposée, le robot est dans un état coopératif et peut donc agir de façon normale. Une fois une action choisie, l'agent agit lors de la phase d'action afin d'activer ses effecteurs et/ou changer ses connaissances.

2.3 Choix des actions

À chaque instant t, un robot devra donc choisir entre les différentes actions proposées par les différents modules de décision (aptitudes et coopération). À l'instant t, chaque action act_j du robot r_i va être évaluée. Pour chaque action, cette valeur est calculée en fonction des perceptions, des représentations et des réflexes dans le cas d'un fonctionnement nominal :

$$V_{r_i}^{nomi}(act_j, t) = wp_{r_i}(act_j, t) + wm_{r_i}(act_j, t) + wr_{r_i}(act_j)$$

où :

- $V_{r_i}^{nomi}(act_j, t)$ est la valeur de l'action act_j au temps t pour le robot r_i ;
- $wp_{r_i}(act_j, t)$ est la valeur calculée en fonction des perceptions ;
- $wm_{r_i}(act_j, t)$ est la valeur calculée en fonction de la mémoire ;
- $wr_{r_i}(act_j, t)$ est la valeur calculée en fonction des réflexes.

Toutes les $V_{r_i}^{nomi}(act_j, t)$ sont regroupées dans un vecteur, appelés vecteur de préférences sur les actions. Ce vecteur représente la distribution de probabilité sur l'ensemble des actions d'un agent. Plus la valeur d'une action sera élevée, plus l'agent aura de préférences à exécuter l'action correspondante. Comme pour les aptitudes, un vecteur d'actions coopératives est généré par le module de coopération : $V_{r_i}^{coop}(act_j, t)$. Une fois les valeurs calculées pour chacune des actions du robot, par les deux modules, le vecteur sur lequel portera le tirage de Monte Carlo est une combinaison des deux vecteurs dans lequel le vecteur de coopération subsume le vecteur nominal :

$$V_{r_i}(t) = V_{r_i}^{nomi}(t) \prec V_{r_i}^{coop}(t)$$

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

Tab. 1. Spécification du comportement nominal.

Perceptions (Conditions)	**Effets sur les actions**
$\neg car \wedge cBox$	$\nearrow wp_{r_i}(pick, t)$
$\neg car \wedge \neg cBox \wedge sBox$	$\nearrow wp_{r_i}(forward, t)$
$\neg car \wedge \neg cBox \wedge \neg sBox \wedge \neg inCZ$	$\nearrow wp_{r_i}(\langle CZdir \rangle, t)$
$\neg car \wedge \neg cBox \wedge \neg sBox \wedge inCZ$	$\nearrow wp_{r_i}(backward, t)$ $\nearrow wp_{r_i}(forward, t)$ $\nearrow wp_{r_i}(left, t)$ $\nearrow wp_{r_i}(right, t)$
$car \wedge cLZ$	$\nearrow wp_{r_i}(drop, t)$
$car \wedge \neg cLZ$	$\nearrow wp_{r_i}(\langle LZdir \rangle, t)$

Où :
- car : r_i porte une boîte ;
- $cBox$: r_i est proche d'un boîte ;
- $sBox$: r_i voit une boîte ;
- $inCZ$: r_i est dans la zone de retrait ;
- cLZ : r_i est proche de la zone de dépôt ;
- $inLZ$: r_i est dans la zone de dépôt ;
- $\langle CZdir \rangle$: déplacement à effectuer pour atteindre la zone de retrait ;
- $\langle LZdir \rangle$: déplacement à effectuer pour atteindre la zone de dépôt ;
- $\nearrow$: augmentation des valeurs de préférence.

2.4 Comportement nominal

Le comportement nominal est décrit grâce à des règles qui modifient les valeurs dans le vecteur de préférences V^{nomi}. Ce vecteur est obtenu en ajoutant des valeurs provenant des perceptions ($wp_{r_i}(act_j, t)$) et des valeurs provenant des réflexes ($wr_{r_i}(act_j, t)$). Aucune mémoire n'est nécessaire pour effectuer un comportement nominal. La table 1 montre les valeurs à modifier dans $wp_{r_i}(act_j, t)$ pour atteindre les deux buts disjoints : *reach claim zone* ($\neg car$) et *reach laying zone* (car).

Les valeurs réflexes sont indépendantes du but courant, mais dépendent des perceptions et plus précisément de la direction. Comme pour les fourmis, il est plus facile pour un robot de se déplacer vers l'avant que de faire demi-tour [8]. Par exemple, les valeurs de $wr_{r_i}(act_j, t)$ peuvent être fixées comme suit :
- $wr_{r_i}(forward, t) = 50$;
- $wr_{r_i}(left, t) = 10$;
- $wr_{r_i}(right, t) = 10$;
- $wr_{r_i}(backward, t) = 0$;

Ainsi, même si un but mène un robot vers un mur, le robot peut se déplacer vers un côté, comme les fourmis le font pour éviter les obstacles. Néanmoins, ce mécanisme n'est pas suffisant pour éviter les blocages dans de long couloirs étroits dans lesquels les robots ne peuvent pas se croiser. Le but est plus influent que les réflexes. En conséquence, définir des règles de coopération est nécessaire pour permettre aux robots d'atteindre leur but commun sans blocage.

2.5 Attitude coopérative des agents

La définition de la coopération telle qu'elle est utilisée va au-delà du partage de tâches, de ressources ou de contraintes. Elle est définie en trois points : un agent est coopératif si : tous les signaux perçus sont compris sans ambiguïté (c_{per}) et l'information reçue est utile au raisonnement (c_{dec}) et le raisonnement induit des actions utiles aux autres agents et à l'environnement (c_{act}) [6]. Ainsi

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

une situation non coopérative (NCS) arrive lorsque $\neg c_{per} \vee \neg c_{dec} \vee \neg c_{act}$ est vraie. De plus, nous identifions sept sous-types de NCS :

- l'*incompréhension* ($\neg c_{per}$) : l'agent ne peut extraire le contenu sémantique d'un stimulus reçu (ex : incapacité de reconnaissance de formes ou de lecture de messages riches) ;
- l'*ambiguïté* ($\neg c_{per}$) : l'agent extrait plusieurs interprétations pour un même stimulus (ex : imprécision de reconnaissance de formes ou d'une lecture de message) ;
- l'*incompétence* ($\neg c_{dec}$) : l'agent ne peut exploiter l'état courant de ses connaissances (représentations et compétences) lors de sa phase de décision (ex : l'agent manque de variables dans ces règles d'inférences, si le raisonnement est implémenté par un moteur d'inférences) ;
- l'*improductivité* ($\neg c_{dec}$) : l'agent ne peut proposer d'action à effectuer pendant la phase d'action (ex : l'agent manque de règles d'inférences, si le raisonnement est implémenté par un moteur d'inférences) ;
- la *concurrence* ($\neg c_{act}$) : l'agent perçoit qu'un autre agent est sur le point d'agir afin d'atteindre le même état dans l'environnement ;
- le *conflit* ($\neg c_{act}$) : l'agent, à partir de ses représentations et perceptions, pense que la transformation de l'environnement qu'il est sur le point d'opérer, sera incompatible avec l'activité d'un autre agent ;
- l'*inutilité* ($\neg c_{act}$) : l'agent, à partir de ses représentations et perceptions, pense que son action ne peut modifier l'état de l'environnement ou que ses résultats ne seront sans aucun intérêt pour les autres agents ;

L'attitude coopérative des agents les force à éviter ces situations ou à les résoudre le cas échéant. La conception d'agents coopératifs se concentre sur la spécification des NCS –comme une sorte de programmation orientée exceptions dans laquelle le programmeur se focalise sur les exceptions.

3 Auto-organisation coopérative

Dans la section précédente, les différents modules d'un robot et leurs composantes ont été détaillés, à l'exception du module de coopération. Cette section a pour but de discuter des règles de coopération à établir afin de permettre au système multi-robot d'être fonctionnellement adéquat avec son environnement.

3.1 Coopération réactive

À partir de deux robots effectuant la tâche de transport dans le même environnement, le comportement nominal peut ne plus être adéquat. En effet, un robot possède les compétences pour remplir sa tâche, mais pas pour travailler avec les autres. Dans cet environnement, fortement contraint à cause des couloirs étroits, des zones d'interférences spatiales apparaissent. Si deux robots, l'un portant une boîte et se dirigeant vers la zone de dépôt et un autre robot, se dirigeant vers la zone de retrait pour retirer une boîte, se rencontrent face-à-face dans un couloir, ils restent bloqués car il ne peuvent poser les boîtes que dans la zone

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

TAB. 2. Spécification de la NCS "*un robot se retourne*".

Condition	Action
$ret \wedge fR$	$\nearrow V_{r_i}^{coop}(t, right)$
$ret \wedge fL$	$\nearrow V_{r_i}^{coop}(t, left)$
$ret \wedge \neg(fL \vee fR) \wedge ant \wedge toGoal \wedge cGoal$	$\nearrow V_{r_i}^{coop}(t, backward)$
$ret \wedge \neg(fL \vee fR) \wedge ant \wedge toGoal \wedge \neg cGoal$	$\nearrow V_{r_i}^{coop}(t, forward)$
$ret \wedge \neg(fL \vee fR) \wedge ant \wedge \neg toGoal$	$\nearrow V_{r_i}^{coop}(t, backward)$
$ret \wedge \neg(fL \vee fR) \wedge \neg ant$	$\nearrow V_{r_i}^{coop}(t, forward)$

Où :
- ret : r_i est en retournement ;
- fR : la case de droite est libre ;
- fL : la case de gauche est libre ;
- ant : est face à un robot antinomique ;
- $toGoal$: r_i se déplace en direction de son but ;
- $cGoal$: r_i est plus proche de son but que d'un but antinomique.

de dépôt. Il devient alors nécessaire de pourvoir les robots de comportements coopératifs. Nous distinguons alors deux situations non coopératives pouvant être résolues de manières réactives :

Un robot est bloqué. Un robot r_1 ne peut plus avancer à cause d'un mur, ou d'un robot r_2 allant dans le sens opposé[1]. Dans ce cas, s'il en a la possibilité, r_1 se déplacera sur les côtés. Ceci correspond à une augmentation des valeurs du vecteur d'actions coopératives correspondant aux actions de déplacements latéraux : $V_{r_1}^{coop}(t, droite)$ et $V_{r_1}^{coop}(t, gauche)$. Dans le cas où r_1 n'a pas de choix de déplacements latéraux, deux autres possibilités sont ouvertes. Si r_2 est un robot ayant un but antagoniste, c'est le robot qui est le plus éloigné de son but[2] qui se retourne (augmentation de $V_{r_i}^{coop}(t, arriere)$) pour laisser place à celui qui est le plus proche de son but (qui augmente $V_{r_i}^{coop}(t, avant)$ quitte à attendre un tour). Dans le cas où r_2 a le même but, sauf si r_1 est suivi d'un robot ayant un but antagoniste ou que r_1 s'éloigne de son but (visiblement il se déplace vers une zone à risque), r_1 se retourne ; sinon r_1 avance et r_2 recule.

Un robot se retourne. Un robot r_1 se retourne[3] à cause d'un blocage. S'il en a la possibilité, r_1 se déplacera sur les côtés. Sinon, r_1 continue à avancer jusqu'à ce qu'il n'y ait plus de marge de manœuvre ou bien qu'il rencontre un autre robot r_2 qui se retourne et qui est plus près de son but que lui. La table 2 résume le comportement du robot face à cette situation. Si une queue se forme, le premier robot se retournant sera vu par le second qui ce retournera à son tour. Ensuite, le troisième se retournera pour la même raison et ainsi de suite, jusqu'à ce qu'il n'y ait plus d'obstacle.

Ces deux règles correspondent à des *conflits* de ressources (les couloirs) ou bien à de l'*inutilité* lorsque le robot se retourne et s'éloigne de son but. Dans le cas des robots, nous ne spécifierons pas de situations d'incompréhension, car

[1] Dans le cas où r_2 se déplace sur les côtés ou dans le même sens que r_1, il n'est pas considéré comme bloquant car il ne gênera plus au pas suivant.

[2] Un robot est capable de connaître la distance qui le sépare de son but, car il connaît sa position et celle de son but. Comme il peut avoir à remplir un autre but, il est capable de connaître la distance qui sépare un autre robot de son but.

[3] Un robot sera considéré comme se retournant tant qu'il n'a pas le choix de déplacements latéraux.

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

les robots sont incapables de communiquer directement. Ces deux règles, relativement simples à spécifier, assurent que les robots ne s'interbloquent pas dans les couloirs. Par contre, ces situations ne font que résoudre les problèmes sur l'instant, créant des mouvements de retournement et donc des pertes de temps pour rapporter ou aller chercher des boîtes.

3.2 Coopération par anticipation

Pour éviter de répéter leurs erreurs de coopération passées, les robots doivent mémoriser les positions des zones à risque. Il est possible de spécifier des règles de coopération permettant d'anticiper les situations de blocage afin que le collectif soit plus efficace. Dans ce cas, nous parlerons de règles de coopération d'*optimisation*. Les règles précédentes permettent à un robot de résoudre une situation de blocage. Un robot se retrouve dans une telle situation parce qu'il est entré dans une zone empruntée par des robots remplissant un but antinomique. Afin d'anticiper cette situation, il faut donc que les robots soient capables d'éviter de rentrer dans une zone à risque : zone d'où sortent des robots ayant un but antinomique. Ainsi, une règle d'anticipation peut être définie :

Un robot voit un robot antinomique. Si un robot r_1 voit un robot r_2 ayant un but antinomique (un état de transport différent), si r_1 peut se déplacer sur les côtés, il le fera, sinon il continue à avancer.

Cependant, cette anticipation réactive présente un problème majeur : une fois qu'un robot a évité une zone à risque, rien n'assure qu'il n'y revienne à nouveau, guidé par son but. Afin de pallier cet oubli, notre robot est muni d'une mémoire (dans le module de représentations) des zones à risque. Chaque fois qu'un robot r_i rencontre une NCS d'anticipation face à un robot r_j, il ajoute à sa mémoire un tuple (ou balise virtuelle) $\langle posX(r_j,t), posY(r_j,t), goal(r_i,t), w\rangle$ où $posX(r_i,t)$ et $posY(r_i,t)$ représentent les coordonnées de r_j au moment de la NCS. $goal(r_i,t)$ représente le but que r_i essayait de remplir au moment de la situation. w représente une valeur de répulsion. Plus la valeur est élevée, plus le robot évitera la zone lorsqu'il essaie de remplir un but autre que $goal(r_i,t)$. Ainsi, le robot analyse toutes les balises présente dans sa mémoire[4] dont la distance est inférieure à la valeur de perception (pour conserver le principe de localité). Une balise de poids w située dans la direction dir à une distance de d cases induira une augmentation de $V_{r_i}^{coop}(t, dir_{opp})$ de w avec dir_{opp} étant la direction opposée à dir.

Comme la mémoire est limitée, les tuples ajoutés doivent pouvoir disparaître au fur et à mesure de la simulation. Par exemple, le poids w doit pouvoir diminuer d'une certaine valeur δ_w (appelée *facteur d'oubli*) à chaque pas de simulation. Une fois que $w = 0$ le tuple est enlevé de la liste. Cette méthode correspond à une utilisation de phéromones *virtuelles* et *personnelles*. Enfin, comme chez les fourmis, les robots renforcent leurs balises : un robot passant en position d'une de ces balises ayant un but autre que le but courant, réinitialise la valeur. En

[4] Les robots ne peuvent partager leur mémoire comme ils ne peuvent communiquer.

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

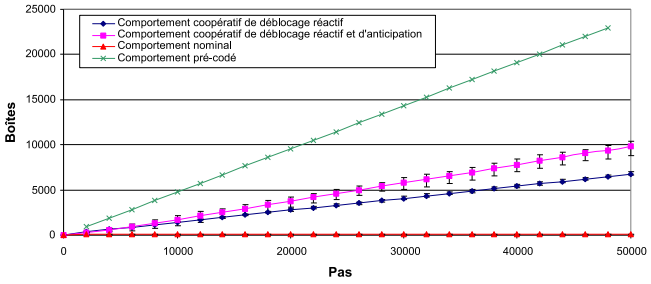

Fig. 2. Comparaison du nombre moyen de boîtes rapportées en fonction du temps de simulation et déviation standard pour 15 simulations pour différents comportements : individualiste, coopératif débloquant, coopératif par anticipation, et pré-codé.

effet, si le robot se trouve sur cette position, c'est bien que cette zone doit être évitée lorsqu'il remplit un autre but. Mais contrairement aux algorithmes fourmis, aucune stigmergie ni support physique n'est nécessaire, et l'environnement n'est pas physiquement marqué.

4 Expérimentations

Afin de valider notre approche incrémentale de l'attribution de comportements coopératifs aux robots transporteurs, le modèle a été implémenté et simulé dans un environnement présenté dans la figure 1. L'environnement de simulation correspond à deux salles (25 x 30 cases) séparées par deux couloirs longs et étroits (30 x 1 cases). 300 robots sont placés aléatoirement dans la salle de retrait. Ces robots possèdent une capacité de perception de 5 cases. Ils peuvent se déplacer d'une case par tour. S'ils possèdent la capacité à anticiper les conflits, leur mémoire peut contenir 400 tuples avec $w = 1500$ et $\delta_w = 1$.

Comparaison réaction/anticipation. La figure 2 présente une comparaison des résultats pour les comportements coopératifs présentés ci-dessus. Il montre le nombre moyen de boîtes rapportées pour 15 simulations (300 robots, 2 couloirs, perception à 5 cases de distance), correspondant au comportement nominal (individualiste) et à deux autres comportements coopératifs : un comportement débloquant réactif (voir section 3.1) et un comportement coopératif par anticipation (voir section 3.2). Le comportement de déblocage permet d'obtenir une efficacité linéaire permettant aux robots de remplir leur tâche de transport sans être bloqués, tandis que les robots avec comportement nominal individualiste le sont systématiquement. En ajoutant l'anticipation de blocage, le collectif gagne en efficacité (au moins 30% de boîtes rapportées en plus). Ceci revient à avoir défini une optimisation du comportement de déblocage. Conformément aux AMAS, nous observons expérimentalement que la résorption des NCS locales conduit à l'adéquation fonctionnelle collective. Plus les NCS sont prises

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

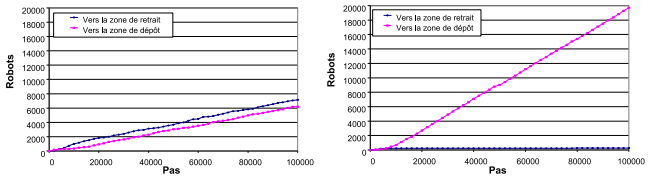

Fig. 3. Comparaison du nombre de robots entrant dans le couloir du haut pour deux comportements différents : débloquant réactif (à gauche) et, débloquant et anticipation (à droite).

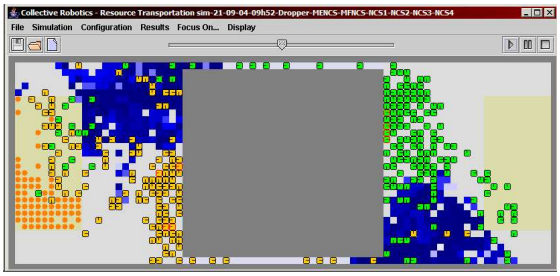

Fig. 4. Positionnement de tous les marqueurs virtuels (carrés sombres) pour tous les robots et les deux buts.

en compte localement, meilleures sont les performances. Enfin, le comportement global est insensible à l'état initial comme le montre la déviation standard qui est négligeable (voir figure 2).

Néanmoins, fournir des comportements coopératifs aux agents n'est pas si efficace que de pré-coder une affectation de sens de circulation aux couloirs –ce qui n'est clairement pas adaptatif– comme le montre la figure 2. Ici, on fixe le parcours des robots : zone de retrait, couloir 1, zone de dépôt, puis couloir 2. Mais dans les sections suivantes, les propriétés d'adaptation et de robustesse apportées par l'auto-organisation coopérative devient plus évidente.

Émergence de circulation. La figure 3 présente la fréquentation d'un couloir, c'est-à-dire le nombre de robots entrant dans un couloir, pour les deux comportements coopératifs : débloquant réactif (à gauche) et débloquant par anticipation (à droite). Dans le cas du comportement coopératif avec anticipation nous pouvons observer l'émergence de sens de circulation. Les robots dédient collectivement les couloirs à des buts particuliers. En effet, les marqueurs sont positionnés uniquement devant l'entrée d'un couloir pour une seule direction, comme le montre la figure 4. De plus, la mémoire n'est pas saturée, et connaît un remplissage maximal uniquement en début d'apprentissage.

Nous pouvons ici attribuer le caractère émergent à ce phénomène car les robots n'ont aucune notion de couloir –contrairement à de précédents travaux

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

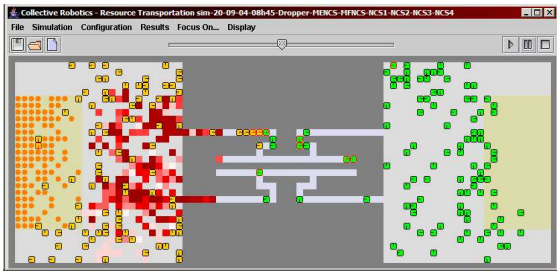

Fig. 5. Positionnement de tous les marqueurs virtuels (carrés sombres) pour le but *atteindre la zone de dépôt* dans un environnement difficile (avec impasses).

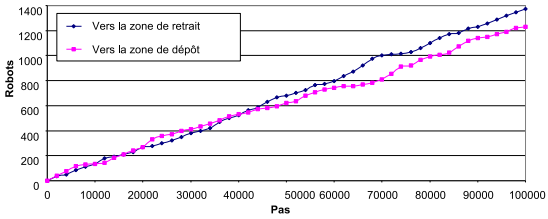

Fig. 6. Nombre de robots entrant dans le couloir du haut dans un environnement difficile pour les deux buts en fonction du temps de simulation.

[9]. Cette vue est uniquement utilisée à des fins d'observation : les robots ne peuvent la prendre en compte. Ainsi, uniquement grâce aux informations de bas niveau dont ils disposent, ils ont établi un comportement de circulation ce qui conduit à une *optimisation* de leur rendement et à un *apprentissage distribué*. De plus, le sens de circulation peut varier à chaque simulation : une petite variation initiale de quelques robots va rapidement en décider. Ici, d'après la distinction faite par [10], nous parlerons d'émergence dite *faible* car les robots n'ont pas la connaissance (globale) du nombre de boîtes qu'ils rapportent et ceci ne les motive pas à rapporter plus de boîtes ou à changer leur organisation.

Robustesse dans des environnements difficiles. L'environnement de la figure 1 est très simple : couloirs droits et distants, pas d'impasse. Pour vérifier la robustesse du collectif, des simulations ont été faites dans l'environnement avec impasses de la figure 5. Les robots sont capables de gérer de telles difficultés en utilisant les directions à un niveau d'abstraction supérieur. Par exemple, la direction *forward* n'est pas simplement interprétée comme "aller droit devant". Si le robot n'a qu'une seule direction libre, *left* par exemple, *forward* sera interprété comme *left*. En conséquence, le problème de blocage dans les impasses est facilement résolu. Mais les robots ne distinguent plus les couloirs, car ils sont trop proches et les marqueurs ne sont plus assez précis. Dans notre exemple, les couloirs sont distants de 9 cases, ce qui est inférieur à deux fois la distance de perception des robots, qui est de 5 cases. Par conséquent, les zones marquées se

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

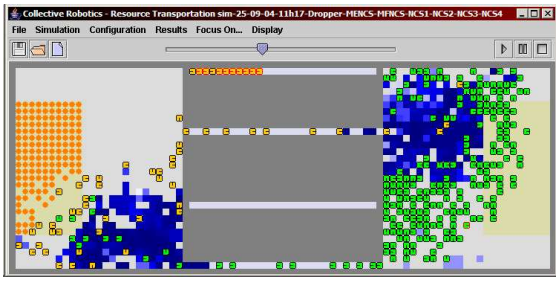

Fig. 7. Positionnement de tous les marqueurs virtuels (carrés sombres) dans un environnement dynamique avec deux couloirs fermés.

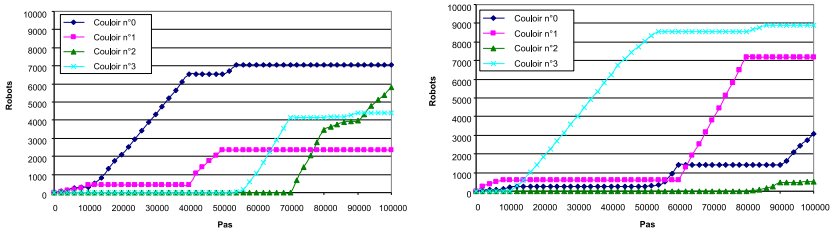

Fig. 8. Nombre de robots entrant, dans un environnement dynamique, pour les deux buts : *atteindre la zone de retrait* ($\neg car$, en haut) et *atteindre la zone de dépôt* (car, en bas).

chevauchent et les robots ne peuvent plus précisément détecter quel couloir éviter. Ainsi, il n'y a plus d'affectation de sens de circulation, comme le montrent les figure 5 et 6. Toutefois, le comportement collectif est assez robuste pour qu'aucun blocage n'apparaissent, même si le système devient moins efficace, car le processus d'apprentissage est inutile. Résoudre ce problème implique de soit modifier des paramètres comme la distance de perception ou les forces de répulsion des marqueurs en fonction de l'environnement (ce qui est trop dépendant de l'environnement), soit ajouter des capacités d'apprentissage sur ces paramètres en définissant de nouvelles règles de résolution coopérative –en considérant ces situations comme des NCS.

Adaptation à la dynamique. Dans les deux cas précédents, l'environnement est statique (exceptés les robots qui se déplacent). Pour tester l'adaptation du comportement collectif, des simulations ont été effectuées dans un environnement avec 4 couloirs. Tous les 10.000 pas, deux couloirs sont choisis aléatoirement et fermés pour ajouter de l'indéterminisme et de la dynamique à l'environnement. Ici encore, le collectif est moins efficace, mais il n'y pas de *blocage*. Bien sûr, si la fermeture des couloirs est trop fréquente, les robots n'ont pas le temps apprendre et d'affecter des sens de circulation aux couloirs. En effet, 300 robots ont besoins d'environ 2.000 pas pour s'adapter. Néanmoins, l'effet de la dynamique de l'environnement peut devenir négligeable avec un grand nombre de pas.

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

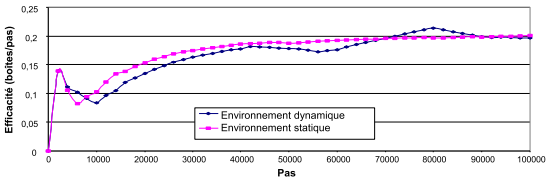

Fig. 9. Efficacité (nombre de boîtes rapportées/temps de simulation) comparée entre une simulation dans un environnement statique avec 2 couloirs et une simulation dans un environnement dynamique avec 4 couloirs dont 2 sont choisis aléatoirement tous les 10.000 pas pour être fermés.

La figure 7 montre le positionnement des marqueurs de tous les agents (mais un agent ne perçoit que les siens) dans l'environnement. Seules les entrées des couloirs ouverts sont marquées et certains robot peuvent être capturés dans les couloirs fermés, et deviennent ainsi inutiles. Le collectif s'adapte efficacement comme le montre la figure 8. Pendant chaque période de 10.000 pas, les deux couloirs ouverts sont exploités et un sens de circulation apparaît. La figure 9 montre le nombre de boîtes rapportées divisé par le nombre de pas de simulation pour deux environnements différents : un statique et un dynamique. L'efficacité du collectif dans l'environnement statique est plus stable contrairement au collectif évoluant dans l'environnement dynamique. Mais, finalement, les deux atteignent la même efficacité.

5 Discussion

Au regard des résultats précédents, l'approche AMAS par agents coopératifs est pertinente pour plusieurs raisons : (*i*) Contrairement aux approches par algorithmes fourmis [3], les robots *ne marquent pas physiquement* leur environnement grâce à des phéromones, mais mémorisent des marqueurs virtuels et personnels ; (*ii*) Contrairement aux approches compétitives [11] ou altruistes [12], les robots *ne communiquent pas directement* pour informer d'autres robots proches ou échanger des requêtes et des intentions, ce qui est un avantage en faveur de l'homogénéité physique des robots ; (*iii*) Le codage du comportement coopératif est *insensible* au nombre de robots, à la topologie et aux dimensions de l'environnement ; (*iv*) *Aucun feedback global* n'est nécessaire pour mener le système à l'adéquation fonctionnelle, ce qui empêche d'atteindre des minima locaux induits par une fonction de fitness, par exemple ; (*v*) Enfin, les collectifs obtenus sont *robustes et adaptatifs*, malgré des *capacités de perception et de communication très limitées*.

Néanmoins, des choix ont dû être établis concernant l'affectation de valeurs qui peuvent modifier drastiquement le comportement global. Pour le moment, la méthode ADELFE, qui nous a fourni le modèle d'agent coopératif, ne propose aucun guide pour trouver les NCS et les valeurs des différents paramètres ; les concepteurs doivent le faire d'eux-mêmes. Par exemple, les poids initiaux pour

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

les marqueurs et le facteur d'oubli ont été ajustés grâce au temps théorique mis par des robots pour faire un aller-retour dans l'environnement. Ceci peut complètement changer dans un environnement différent avec plus ou moins de couloirs pouvant s'ouvrir et se fermer lors du fonctionnement. Des simulations ont été faites dans de tels environnements, et les valeurs affectées semblent correctes à moins que les couloirs soient trop proches les uns des autres ou que la fréquence de fermeture soit trop élevée. Ces valeurs pourrait cependant être apprise en cours de fonctionnement, ce qui est une de nos perspectives. De plus, à partir du problème de transport collectif de ressources, nous nous somme focalisés sur les NCS particulières : conflit et inutilité. Si les robots sont capables de communications à haut niveau (pour échanger des données comme des marqueurs afin de partager leurs expériences), des situations d'incompréhension ou d'ambiguïté peuvent apparaître. Dans ce cas, ADELFE propose d'analyser les protocoles entre agent pour identifier les points d'entrée d'échanges non coopératifs.

Enfin, l'application que nous avons choisie et la solution que nous avons considérée sont des exemples typiques de systèmes où aucune hiérarchie *a priori* n'est définie. L'auto-organisation mène la société d'agents à l'adéquation fonctionnelle sans avoir défini d'organisation statique ; ce qui est en partie dû à l'homogénéité du collectif : tous suivent un code de conduite coopérative commun. Si le collectif est hétérogène (vitesses différentes, fonctions différentes, complémentaires ou non, etc.), les notions de hiérarchie et de priorité deviennent pertinentes. Cependant, afin d'assurer les propriétés d'extensionnalité [5] et d'irréductibilité des systèmes émergents, prédéfinir une organisation est prohibé ; le cas échéant, la fonction du système est intentionnellement définie, et par conséquent est non adaptative. Ainsi, l'organisation peut être vue comme un phénomène émergeant des relations entre agents, donc un résultat et non un schéma prédéfini. Cependant, il nécessite de définir complètement les comportements coopératifs locaux.

6 Conclusion

Dans cet article, nous avons montré la pertinence de l'usage de la coopération comme critère local de réorganisation dans des collectifs afin d'être plus adaptés à des tâches spécifiques. Compte tenu de la non-connaissance de la tâche globale et de l'environnement, le collectif atteint par auto-organisation un comportement émergent cohérent et est donc plus robuste aux perturbations environnementales. Notre plate-forme de simulation porte sur un problème simple de transport collectif de ressources dans des environnements statiques ou dynamique pouvant présenter des difficultés, comme des impasses, par exemple. Les simulations dans de tels environnements difficiles confirment l'intérêt des collectifs auto-organisateurs par coopération.

Enfin, cette application a été développée pour confronter le modèle d'agent coopératif proposé par la méthode ADELFE à une tâche nécessitant adaptation et robustesse. Cependant, cette application est très spécifique, dans le sens où

[5] c'est-à-dire que la fonction du système est définie par des relations entre les entrées et les sorties, et non pas par un algorithme.

Published in *7ème Congrès de la Société Française de Recherche Opérationnelle et d'Aide la Décision (ROADEF'06), 6, 7 et 8 Février 2006, Lille, France*, pages 385-400, Presses Universitaires de Valenciennes, 2006.

elle ne concerne que des agents quasi-réactifs et non communicants. Dans le cas d'agents capables d'actes de langage, d'autres schémas de comportements coopératifs devront être définis. Un premier pas a été fait par l'utilisation de ce modèle à une application de résolution d'emplois du temps distribués, dans laquelle des agents négocient et cherchent des positions dans un espace de constraintes [13].

Références

1. Heylighen, F. In : The Science of Self-organization and Adaptivity. EOLSS Publishers (1999)
2. Prigogine, I., Nicolis, G. : Chapters III & IV. In : Self Organization in Non-Equilibrium Systems. Wiley and Sons (1977)
3. Bonabeau, E., Theraulaz, G., Deneubourg, J.L., Aron, S., Camazine, S. : Self-organization in social insects. Trends in Ecology and Evolution **12** (1997) 188–193
4. Van Dyke Parunak, H. : "Go to the Ant" : Engineering Principles from Natural Agent Systems. Annals of Operations Research **75** (1997) 69–101
5. Maturana, H., Varela, F. : The Tree of Knowledge. Addison-Wesley (1994)
6. Capera, D., Georgé, J., Gleizes, M.P., Glize, P. : The AMAS theory for complex problem solving based on self-organizing cooperative agents. In : 1^{st} International TAPOCS Workshop at 12^{th} IEEE WETICE, IEEE (2003) 383–388
7. Bernon, C., Camps, V., Gleizes, M.P., Picard, G. : Designing Agents' Behaviors and Interactions within the Framework of ADELFE Methodology. In : 4^{th} International Workshop on Engineering Societies in the Agents World (ESAW'03). Volume 3071 of LNCS., Springer-Verlag (2003) 311–327
8. Topin, X., Fourcassié, V., Gleizes, M.P., Theraulaz, G., Régis, C., Glize, P. : Theories and experiments on emergent behaviour : From natural to artificial systems and back. In : European Conference on Cognitive Science, Siena, Italy. (1999)
9. Picard, G., Gleizes, M.P. : An Agent Architecture to Design Self-Organizing Collectives : Principles and Application. In : AISB'02 Symposium on Adaptive Multi-Agent Systems (AAMASII). Volume 2636 of LNAI., Springer-Verlag (2002) 141–158
10. Müller, J.P. : Emergence of collective behaviour : simulation and social engineering. In : 4^{th} International Workshop on Engineering Societies in the Agents World (ESAW'03). (2004)
11. Vaughan, R., Støy, K., Sukhatme, G., Matarić, M. : Go ahead make my day : Robot conflict resolution by aggressive competition. In : 6^{th} International Conference on Simulation of Adaptive Behaviour. (2000)
12. Lucidarme, P., Simonin, O., Liégeois, A. : Implementation and Evaluation of a Satisfaction/Altruism Based Architecture for Multi-Robot Systems. In : IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA. (2002) 1007–1012
13. Picard, G. : Résolution d'emploi du temps dynamique et distribuée par auto-organisation coopérative. In : 7èmes Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA'05), Plate-forme AFIA, Nice. (2005)