Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

Modélisation et expérimentations d'une décision locale basée sur l'auto-organisation coopérative

Application au problème des N reines

Gauthier Picard — **Pierre Glize**

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

RÉSUMÉ. *L'auto-organisation par coopération permet de concevoir des systèmes multi-agents capables de réagir et de s'adapter collectivement aux perturbations environnementales –ouverture et dynamique, par exemple. C'est une approche par recherche locale qui utilise la notion de coopération comme un critère local de réorganisation permettant aux parties –appelées agents coopératifs– de changer leurs interactions et leurs positions dans l'organisation. De tels systèmes produisent alors des fonctions émergeant des interactions locales. Cette approche est illustrée grâce à un problème classique de décision distribuée : le problème des N reines.*

ABSTRACT. *Cooperative self-organization is a means to design adaptive multi-agent systems able to collectively react to environmental disturbances –openness or dynamics, for example. This local search approach proposes to use the cooperation notion as the local reorganization criterion enabling parts –then called cooperative agents– to change their interactions and their locations within the organization. Such systems then perform functions emerging from internal interactions. This approach is illustrated with a classical distributed decision problem: the N queens problem.*

MOTS-CLÉS : *Coopération, Auto-organisation, Contraintes, Adaptation*

KEYWORDS: *Cooperation, Self-organisation, Constraints, Adaptation*

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

1. Introduction

Concevoir des systèmes décisionnels automatiques peut devenir de plus en plus difficile lorsque les participants sont distribués –logiquement, géographiquement ou temporellement– comme dans la prévision de crue, la génération d'emploi du temps ou la conformation de molécules [GEO 03]. Les CSP (*Constraint Satisfaction Problems*) distribués et dynamiques sont des formalismes classiques pour s'attaquer aux problèmes décisionnels. Mais ces approches classiques manquent souvent d'efficacité ou d'adaptation lorsque l'environnement (qui modifie les contraintes) est ouvert et dynamique. Pour répondre à ce manque, des approches alternatives, venant de la recherche locale ou de l'optimisation métaheuristique peuvent être emphasées. Les algorithmes fourmis sont un exemple d'implémentation de modèles inspirés de la nature pour résoudre des problème de résolution d'emploi du temps [SOC 02] ou pour optimiser des processus de résolution difficiles [DOR 96]. D'autre part, les systèmes multi-agents (SMA) ont montré leur pertinence dans des domaines variés et sont un moyen alternatif de modéliser les problèmes de décision distribuée dans lesquels des agents autonomes doivent coopérer pour trouver une solution collective.

Dans cette article, les SMA sont utilisés pour modéliser et résoudre des CSP en utilisant la notion d'*auto-organisation coopérative*, inspirée des phénomènes biologiques et sociologiques. Cette approche est d'abord introduite par la reformulation des problèmes de décision distribuée dans le cadre des CSP puis dans un paradigme multi-agent orienté organisation, dans la section 2. L'auto-organisation coopérative, qui est un moyen d'obtenir l'adaptation à la dynamique environnementale est présentée dans la section 3. Pour illustrer cette approche, le problème des N reines est modélisé dans la section 4 et expérimenté dans la section 5. Plus loin, nous discutons le potentiel de cette modélisation en la généralisant à d'autres problèmes, dans la section 6, avant de conclure dans la section 7.

2. Contraintes et systèmes multi-agents

Les problèmes de décision basés sur les contraintes peuvent être exprimés en utilisant le cadre des CSP. Un CSP est un triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ tel que $\mathcal{X} = \{x_1, \ldots, x_n\}$ est un ensemble de variables à instancier. $\mathcal{D} = \{D_1, \ldots, D_m\}$ est l'ensemble des domaines de ces variables. Chaque variable x_i est relative à un domaine de valeur. $C = \{c_1, \ldots, c_k\}$ est l'ensemble des contraintes, qui sont des relations entre certaines variables de $\mathcal{X}$ qui contraignent les valeurs que des variables peuvent avoir simultanément. Par conséquent, décider revient à trouver une solution, c'est-à-dire une affectation complète et consistante de $\mathcal{X}$.

Dans les problèmes décisionnels distribués basés sur les contraintes (DCSP), la distribution peut affecter les variables ou les contraintes. La plupart des approches considèrent le premier type de distribution en définissant une fonction ϕ qui lie les variables aux participants (les agents par exemple) : $\phi(c_i) = j$ signifie que la contrainte c_i appartient à j [YOK 98]. Enfin, comme dans un environnement complexe et dynamique, toutes les contraintes peuvent ne pas être complètement satisfaites ou insatisfaites, les CSP distribués sont souvent abordés en termes d'optimisation du degré

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

général de contraintes non vérifiées. Trouver une solution optimale revient donc à trouver une solution pour laquelle la somme totale des contraintes (pondérées) non satisfaites est minimale. Cette formulation s'appelle un problème d'optimisation de contraintes distribuées, ou DCOP [MOD 03].

2.1. *Point de vue multi-agent*

Afin de proposer une approche de modélisation alternative, inspirée des insectes sociaux et de l'intelligence collective, nous pouvons analyser les CSP d'un autre point de vue. Une notion primordiale lorsque l'on modélise des SMA est l'*environnement*. Dans les CSP, en considérant les contraintes comme possédées par les agents, l'environnement est l'espace des possibilités d'affectations des variables, c'est-à-dire une grille à n dimensions –le produit cartésien de tous les domaines de $\mathcal{D}$– composée de cellules à l'intersection des domaines de valeurs. Les *agents* doivent explorer la grille pour y trouver une position –une affectation de variables qui vérifie leurs contraintes. Par conséquent, le principal but d'un agent est de réserver des cellules de la grille afin d'indiquer les affectations de variables, en s'y déplaçant de cellule en cellule. Comme les agents n'ont que des perceptions limitées (quelques cellules, et non la grille dans son intégralité), leur degré de satisfaction est déterminé de manière complètement locale. De plus, comme les agents sont des entités situées, ils ne peuvent occuper qu'une cellule à la fois, mais une cellule peut contenir plusieurs agents. Nous distinguons la cellule qui est occupée de celle qui est réservée. Cette forme de modélisation a déjà été appliquée avec succès en utilisant les algorithmes fourmis [SOC 02] –dans lesquels les agents-fourmis déposent des phéromones pour réserver des créneaux horaires dans un problème d'emploi du temps– ou dans l'algorithme ERA de résolution de CSP –dans lequel des agents-variables explorent leur domaine de valeur– [LIU 02]. Enfin, la solution est obtenue en exécutant le système multi-agent, c'est-à-dire en lançant de manière concurrente les agents dans la grille afin de trouver un positionnement correct. Cela permet de changer les agents (mise à jour, suppression, ajout), les contraintes ou environnement, en ajoutant des dimensions ou des cellules.

2.2. *Point de vue organisationnel*

Maintenant que les CSP ont été reformulés en utilisant la terminologie agent, nous pouvons exprimer les problèmes de décision collective grâce à une notion orientée multi-agent : l'*organisation*. Une organisation est une description de toutes les relations inter-agents. Dans notre approche, ces relations ne sont pas directement les relations définies par les contraintes partagées par les agents, mais les relations spatiales. En fait, comme les agents ne perçoivent qu'un nombre limité de cellules, les interactions d'un agent sont définies par sa position à un instant donné. Ainsi, un problème de décision distribué est équivalent à trouver une organisation adéquate, dont le positionnement de tous les agents vérifie au mieux (en terme d'optimisation, si le problème ne possède pas de solution complète) les contraintes de tous les agents. Si le problème initial est surcontraint, le système ne peut trouver de solution à moins que les contraintes soient relâchables. La relaxation est la possibilité pour un agent

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

de ne pas vérifier certaines contraintes. Elle doit être mise en œuvre uniquement pour obtenir une solution approchée, optimisée en terme de degré global de relaxation. Ce degré peut simplement être le nombre de contraintes relâchées, ou bien la somme des pondérations des contraintes relâchées dans le cas de CSP pondérés [MOD 03].

Un SMA étant considéré comme un système dynamique, il peut passer par plusieurs organisations avant d'atteindre la solution. Compte tenu de l'autonomie des agents, il n'y a pas de contrôleur pour forcer la réorganisation. Les agents se déplacent dans la grille suite à des décisions internes locales. Par conséquent, les concepteurs doivent fournir des capacités de micro-niveau pour changer les interactions locales et ainsi changer l'organisation globale du système. Le problème majeur de l'auto-organisation est de définir le déclencheur de la réorganisation. Dans les cartes auto-adaptatives de Kohonen, l'organisation est représentée par les poids affectés à chaque neurone et à son voisinage. Les neurones changent leur poids en fonction d'une fonction DOG (*Difference of Gaussians*) donnée [KOH 01] . Dans les fourmilières, les fourmis réagissent uniquement à l'attraction stochastique des phéromones, des ressources et des positions des nids. Ainsi, nous pouvons définir l'auto-organisation seulement en fournissant des capacités locales, comme la fonction DOG ou les règles stochastiques de réaction des fourmis, aux agents. Ce critère doit être le plus générique possible. Comme les agents sont des entités décisionnelles de haut niveau, le critère que nous définissons est lui aussi de haut niveau : la *coopération*.

3. Auto-organisation coopérative

Dans les cartes de Kohonen, par exemple, un réseau de neurones formels s'auto-organise pour trouver une configuration adéquate, uniquement en fonction des entrées du système [KOH 01]. Les neurones ont uniquement une perception limitée de leur voisinage. De manière similaire, les algorithmes fourmis sont basés sur des comportements naturels basiques qui peuvent mener à l'apparition de macro-structures complexes comme des chemins, du nid vers les ressources [DOR 96]. Dans cette section, nous présentons une notion inspirée par la sociologie et l'éthologie, la *coopération*, pour obtenir des modélisations similaires pour les SMA.

3.1. *Définition de la coopération*

Afin de fournir des comportements auto-organisateurs à des agents ayant à résoudre un CSP, nous proposons d'utiliser la notion de coopération de [CAM 99]. Ici, la coopération n'est pas seulement le partage de tâches ou de ressources, mais surtout un guide comportemental pour concevoir les agents. La coopération peut aussi devenir un critère local de réorganisation lorsqu'elle est considérée de manière proscriptive : les agents doivent se réorganiser lorsqu'ils ne sont plus coopératifs. Voici nos définitions de coopération et de non coopération. En s'inspirant des définitions sociologiques, la coopération est le juste milieu entre l'altruisme et l'égoïsme. L'altruisme caractérise des entités qui préfèrent aider les autres plutôt que d'atteindre leur propre but, contrairement aux égoïstes qui vont privilégier leur but au dépend des autres. Ainsi, un agent coopératif doit essayer de satisfaire ces buts locaux et ceux des autres aussi égalitai-

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

rement que possible, compte tenu des éventuels coûts ou bénéfices en résultant. Une deuxième définition de la coopération provient de systèmes naturels dans lesquels elle est souvent synonyme de relations symbiotiques entre plusieurs entités. Du point de vue des agents, nous définissons la coopération par trois méta-règles, en fonction de chaque phase du cycle de vie "perception-décision-action" qu'un agent suit :

Definition 1 *Un agent est coopératif s'il vérifie les conditions suivantes :*

– c_{per} : les signaux reçus sont compris sans ambiguïté ;

– c_{dec} : les informations reçues sont utiles au raisonnement de l'agent ;

– c_{act} : le raisonnement mène à des actions utiles aux autres agents et à l'environnement.

La première méta-règle (c_{per}) concerne les agents dialoguants qui échangent des messages ayant une sémantique riche ou les agents ayant une vue complexe de l'environnement et nécessitant de la reconnaissance de forme, par exemple. Pour être aussi coopératif que possible, un agent doit connaître et apprendre les langages d'interaction, si nécessaire. La seconde méta-règle (c_{dec}) concerne tous les types d'agents comme ils possèdent des capacités décisionnelles. Ceci implique que les agents coopératifs doivent être capables de produire des raisonnements à partir de leur compétences (connaissances sur leur tâche), de leurs représentation (connaissances sur l'environnement) et de leur aptitudes (qui permettent de déduire des faits à partir des compétences et des représentations). La dernière méta-règle (c_{act}) est la plus utilisée. Elle nécessite de mesurer l'impact d'une action sur l'environnement social ou physique, en définissant une mesure de coopération comme dans la section 4.

La non coopération est l'exact opposé : ($\neg c_{per} \vee \neg c_{dec} \vee \neg c_{act}$). De telles situations sont appelées *situation non coopératives* (ou NCS) et sont le déclencheur de la réorganisation. Pour résumer, la coopération est le critère local de réorganisation qui permet aux agents de changer leurs interactions lorsque le système ne produit pas une fonction adéquate. D'un point de vue global, la coopération peut être considérée comme une métaheuristique pour explorer l'espace des possibilités en coupant les branches menant à des situations non coopératives. Mais le principal problème de cette approche réside dans le haut niveau de définition de la coopération. Même si [CAM 99] identifie plusieurs types de NCS, ils ne proposent aucun modèle générique de bas niveau. Par conséquent, nous illustrons comment instancier ces méta-règles à un problème classique dans la section suivante.

3.2. *Implémenter la coopération*

Un comportement coopératif peut être spécifié comme des exceptions pour réparer les situations non coopératives [BER 03]. Le principal enjeu de la conception d'un agent coopératif est de lui fournir un comportement nominal et des buts, et ensuite d'y ajouter des capacités de gestion des NCS comme des exceptions en programmation orientée objet –mais à un niveau plus haut. Un autre algorithme a aussi été spécifié pour exprimer les différents types de NCS qu'un agent peut expérimenter [CAP 03].

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

Dans cet article, comme les exemples sont simples, l'idée principale est de considérer les agents comme des objets autonomes en terme de décision suivant un cycle "perception-décision-action". Ce cycle peut être interrompu lorsque des NCS sont détectée par l'agent lui-même. Par conséquent, définir le comportement des agents est équivalent à :

1) spécifier un comportement nominal en attribuant des buts, des compétences et des capacités, comme précisé dans la méthode ADELFE, dédiée à la conception de systèmes multi-agents adaptatifs (ou AMAS) [PIC 04b] ;

2) spécifier des paires *condition-action* décrivant les exceptions à la coopération.

4. Exemple : le problème des N reines

Ce problème consiste à positionner N reines du jeu d'échecs, dans un échiquier de $N \times N$ cases, de telle sorte qu'aucune reine ne puisse en attaquer un autre. Classiquement, ce problème est modélisé en utilisant le formalisme des CSP où les variables à affecter sont les coordonnées des reines ; les domaines de valeurs sont des ensembles de taille N ; et les contraintes impliquent que les reines ne peuvent être sur la même ligne, colonne ou diagonale. Ici, les agents sont les reines.

4.1. *Comportement nominal*

Comme précisé dans la section 3.2, le comportement nominal d'un agent consiste en ses buts, compétences et capacités. Un agent-reine ($q_i \in \mathcal{A}$) peut percevoir la case (ou cellule) qu'il occupe ($cell(q_i)$) et toutes celles qu'il peut attaquer ($pCells(q_i)$). Concernant les actions, une reine peut se déplacer sur une cellule perçue ($moveTo(c_j)$), même si une autre reine l'occupe déjà, réserver les cellules qu'elle perçoit pour informer les autres reines et identifier les conflits ($reserve(c_j)$ ou $reserve(\{c_j, \ldots, c_k\})$), ou simplement rester sur la cellule courante ($rest$). Les marques utilisées pour la réservation affectent l'environnement commun –la grille– et non le système multi-agent qui est composé des reines autonomes. Pour décider de la prochaine case à occuper, les reines doivent être capables de percevoir les autres reines ($pAgents(q_i)$[1]). De plus, lors de la réservation de cellule, une reine pose des données supplémentaires concernant la cellule la moins conflictuelle qu'elle perçoit (voir section 4.3), et par conséquent une reine peut connaître le nombre de reines qui perçoivent une cellule donnée ($cost(c_j)$ ou $cost(q_i)$ avec $cost(q_i) \equiv (cost(c_j)|(c_j = cell(q_i)))$) et leur degré de contrainte.

Durant sont cycle de vie, un comportement nominal d'une reine est décrit dans l'algorithme 1 qui ne mène la reine qu'à un minimum local sans prendre en compte les autres reines. Afin d'éviter de rester sur un minimum local, ce comportement égoïste peut être enrichi par des règles de coopération.

---

1. $q_j \in pAgents(q_i) \equiv (\exists k((c_k \in pCells(q_i)) \wedge (c_k = cell(q_j))))$

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

Algorithme 1 – Comportement nominal pour un agent *égoïste* (q_i)

```
while alive do
  updatePerceptions() ;                              //mettre à jour les données perçues
  if (cost(q_i) == 0) then                                          //pas de conflit
    rest()                                                            //ne rien faire
  else
    moveTo(minConflictCell(q_i)) ;    //bouger sur la cellule perçue la moins conflictuelle
    reserve(pCells(q_i))                                    //marquer les cellules perçues
  endif
done
```

4.2. *Comportement coopératif*

Comme précédemment souligné, les règles d'auto-organisation coopérative pour éviter les NCS sont spécifiées comme des exceptions. Les gardes sont des instanciations des méta-règles présentées dans la section 3.1. Les actions pour réparer les NCS doivent être choisies parmi les actions que les agents peuvent effectuer aussi coopérativement que possible. Ceci peut impliquer de définir un mesure de coopération pour trier les multiples actions. Dans le problème des N reines, en considérant les perceptions et les action données, nous pouvons identifier deux NCS :

– la *concurrence* ($\neg c_{act}$) : deux reines occupent la même cellule ;

– le *conflit* ($\neg c_{act}$) : deux reines peuvent s'entre-attaquer.

Comme les reines ne peuvent pas directement communiquer en utilisant une sémantique complexe, il n'y a pas de cas de $\neg c_{per}$. De la même manière, nous considérons que les reines peuvent toujours choisir au moins une action à effectuer, et au pire rester sur place. Par conséquent il n'y a pas de cas de $\neg c_{dec}$. Les deux NCS identifiées sont différentes instanciations de la méta-règle c_{act}. Les actions pour les résoudre sont les suivantes :

Nom : **Concurrence (pour la reine q_i)**
Condition : $\exists j((j \neq i) \wedge (cell(q_i) = cell(q_j)))$
Actions : `moveTo(`$mostCooperativeCell(q_i)$`)`

La NCS précédente est assez simple : si deux reines sont sur la même case, la première à le détecter se déplace vers la cellule la moins conflictuelle (voir section 4.3), c'est-à-dire $mostCooperativeCell$.

Nom : **Conflit (pour la reine q_i)**
Condition : $\exists j, k((j \neq i) \wedge (q_j \in pQueens(q_i)))$

Actions :
`let` q_j `= lessConstrainedAgent(`$pAgents(q_i)$`) ;`
`if ((`$q_i = q_j$`) ∨ (cost(`c_i`) > cost(`q_j`))`
`then rest`
`else moveTo(`$mostCooperativeCell(q_i)$`)`

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

L'action de résolution de cette NCS dépend des autres agents. Pour être aussi coopératif[2] que possible les agents agissent uniquement s'ils sont moins contraints que leurs voisins (les agents perçus). En d'autres mots, si un agent est moins contraint qu'un autre, il attendra que l'autre se déplace pour faire décroître ses contraintes. Comme l'autre agent est coopératif, il détectera cette situation et se déplacera en conséquence.

4.3. *Mesure de coopération*

Afin d'évaluer la cellule la plus "coopérative" sur laquelle se déplacer, les agents doivent être capable de mesurer localement le degré de coopération des actions de déplacement. Cette mesure doit prendre en compte les degré de contrainte des autres agents. L'idée est de limiter l'impact des mouvements sur les autres en analysant l'agent le plus contraint (le pire) percevant une cellule donnée, de tel sorte que le coût de contrainte pour une reine n'augmente pas.

Definition 2 *Soit l'ensemble $\mathcal{P}^{q_i}$ des cellules perçues par l'agent q_i :*

$$\mathcal{P}^{q_i} \equiv pCells(q_i) \cup \{cell(q_i)\}$$

Definition 3 *Soit la mesure de coopération $coop : \{c_1, \ldots, c_n\} \rightarrow \mathcal{N}$:*

$$coop(c_i) \equiv \max_{q \in \{q' | c_i \in \mathcal{P}^{q'}\}} cost(q)$$

Definition 4 *Soit $\mathcal{C}_{min}^{q_i}$ l'ensemble des cellules ayant un coût minimum du point de vue de l'agent q_i :*

$$\mathcal{C}_{min}^{q_i} \equiv \{c \in \mathcal{P}^{q_i} | \nexists c' \neq c, cost(c') < cost(c)\}$$

Definition 5 *Soit $\mathcal{C}_{coop}^{q_i}$ l'ensemble des cellules les plus coopératives du point de vue de l'agent q_i :*

$$\mathcal{C}_{coop}^{q_i} \equiv \{c \in \mathcal{C}_{min}^{q_i} | \nexists c' \in \mathcal{C}_{min}^{q_i}, c' \neq c, coop(c') < coop(c)\}$$

Ainsi, l'ensemble des cellules les plus coopératives à choisir est l'ensemble des cellules ayant un coût minimum et un impact minimum pour les autres agents. Une première remarque peut être exprimée sur le contenu de $\mathcal{C}_{coop}^{q_i}$, qui ne peut être vide : il contient au moins la cellule courante (voir définition 2). Une seconde remarque concerne les données que nécessite un agent pour évaluer une cellule : le nombre de marqueurs sur une cellule, et le coût de chaque agent ayant marqué la cellule (voir les définitions 3 et 4). Par conséquent, un marqueur pour une cellule doit contenir les coûts courants de son possesseur. Les agents coopératifs sont ainsi capable de déterminer

2. moins égoïste que possible.

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

l'ensemble des cellules les plus coopérative vers lesquelles se déplacer. Mais comment choisir la prochaine cellule sans tomber dans un minimum local ? Voici le principal enjeu décisionnel pour un agent coopératif. À présent, Capera et al ne donne aucun guide [CAP 03]. Donc, nous pouvons choisir n'importe quelle méthode : random, tabou, recuit simulé, etc. Dans les expérimentations suivantes, de bons résultats sont obtenus avec un critère de sélection très simple (voir section 5).

5. Expérimentations

Les deux principaux choix à faire avant d'implémenter les comportements des agents et de lancer le processus de résolution sont :

– la *position initiale* : tous les agents sont initialement positionnés sur le bord gauche de la grille, comme sur la figure 1, afin de pouvoir interpréter les résultats sans y injecter d'aspect indéterministe ;

– la *fonction de sélection* pour choisir la cellule la plus coopérative ($mostCooperativeCell(q_i)$) : comme l'implémentation implique de choisir un ordre dans les cellules perçues (de la plus proche à plus éloignée, d'est en ouest, dans le sens antitrigonométrique), cet ordre est utilisé pour choisir la prochaine cellule. De plus, nous avons ajouté une mémoire limitée d'une cellule, pour éviter de retourner sur une cellule précédemment visitée (implémentation simpliste d'un tabou).

5.1. *Simple déroulement avec 4 reines*

La figure 1 montre un déroulement pour un problème de taille 4. Au début de la résolution, tous les agents sont placés à gauche de l'échiquier et l'environnement n'est *pas marqué*. Le nombre dans les cellules représente le nombre de marqueurs, i.e. le nombre d'agent percevant la cellule. Le système trouve une solution collective en 4 pas (durant lesquels chaque agent agit), qui sont délimités par des rectangles pointillés. Seulement 6 mouvements sont effectués pour atteindre la solution. Certains agents n'agissent pas lors de certains pas, comme ils sont en situation coopérative, et non sur des cellules conflictuelles ou concurrentes. Contrairement aux approches classiques pour la résolution de CSP, les agents changent de ligne et de colonne, comme sur la figure lorsque q_3 se déplace vers la position $(1, 3)$. Même si ce n'est pas une solution, cela représente une étape intermédiaire vers un état plus adéquat. Ceci souligne l'aspect *non informé* de cet algorithme.

5.2. *Résultats pour différentes tailles de problèmes*

La figure 2 montre les résultats de résolutions pour différentes tailles n de problèmes (de 50 à 1500 agents) :

– le *nombre total de déplacements* (pour tous les agents) durant la résolution augmente linéairement avec le nombre d'agents ($\mathcal{O}(4n/3)$) ;

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

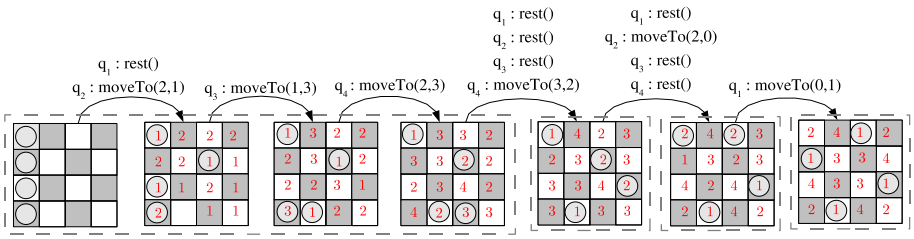

Figure 1. *Résolution pour 4 reines en 4 pas (lignes pointillées) et 6 déplacements*

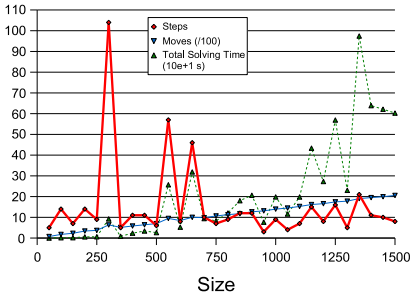

Figure 2. *Résultats de résolutions pour différentes tailles de problèmes (de 50 à 1500)*

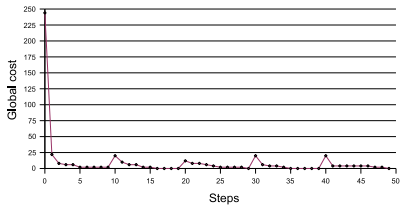

Figure 3. *Coût de contrainte global en fonction du pas de résolution pour un problème à 512 reines*

– le *temps de résolution* (en secondes) augmente significativement en fonction du nombre d'agents. Ce résultat montre la complexité de l'algorithme global, $\mathcal{O}(n^2 s)$: n agents percevant et analysant au pire $4n$ cellules pendant s pas de résolution ;

– le *nombre s de pas* semble ne pas dépendre de la taille du problème. Informellement, il doit dépendre de la définition de la coopération. Comme cette valeur est le lien entre le micro-niveau (les agents) et le macro-niveau (le système), nous ne pouvons, pour le moment, le définir formellement. Mais expérimentalement, nous pouvons le surestimer en $\mathcal{O}(n)$.

Ainsi, la complexité globale du processus de résolution peut être expérimentalement borné en $\mathcal{O}(n^3)$, ce qui est une complexité commune pour résoudre ce genre de problème avec des heuristiques informées avec *hill-climbing* [RUS 95].

5.3. *Adaptation et robustesse*

Une résolution a été effectuée avec la même mise en place qu'auparavant, mais avec uniquement 512 reines et des perturbations aléatoires : chaque 10 pas, 10% d'agents choisis au hasard sont déplacés à des positions aléatoires quelconques de la grille. La figure 3 montre le coût global ($\sum_{q \in \mathcal{A}} cost(q)$) en fonction du nombre de pas. Le collectif, après chaque perturbation, répare rapidement l'organisation et trouve une solution : le système est adaptatif et robuste à la dynamique de l'environnement.

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

De plus, ces résultats montrent qu'une solution est trouvée même avec un positionnement initial aléatoire des reines. Enfin, les premiers pas du processus de résolution (voir figure 1) sont un cas particulier de perturbation car le degré de contrainte initial, associé aux cellules, est erroné. Le résultat du processus de résolution est une organisation spatiale du système, trouvée sans aucune connaissance globale ni *feedback* explicite (uniquement le marquage de l'environnement) comme dans les cartes auto-adaptatives de Kohonen [KOH 01]. C'est donc un processus auto-organisé, au sein duquel la coopération garantit l'évolution vers une solution collective.

6. Discussion

Un premier point de discussion concerne la généralisation de cette approche à d'autres CSP. Comme l'algorithme ne donne aucun guide sur le critère de sélection, il demeure générique. Cependant, ce critère, et les cellules perçues dont il a besoin, peuvent devenir plus difficiles à définir et à implémenter pour des problèmes plus complexes –ayant plus de deux dimensions, par exemple. Dans le cas d'emplois du temps académiques, un prototype a été développé, qui nécessite de fournir aux agents de la mémoire sur les autres agents et sur les cellules occupées [PIC 05]. De plus, nous avons aussi appliqué cette approche à un problème de transport de ressources multi-robot, dans lequel des robots évoluaient dans un environnement contraint. Les robots devaient établir, sans communication ni marquage physique, des sens de circulation dans des couloirs pouvant se fermer dynamiquement afin de ne pas former de bouchons. La simple utilisation d'une mémoire des emplacements des expériences de NCS ainsi que des mécanismes d'attraction-répulsion ont permis d'obtenir des comportements collectifs émergents [PIC 04a].

Un autre point de discussion important est la propriété de terminaison de l'algorithme proposé. Pour le moment, il n'a pas été prouvé formellement que le collectif peut trouver une solution adéquate en un temps fini, contrairement aux approches distribuées et dynamiques classiques [YOK 98, BES 01]. Ceci est principalement dû au large gap entre les entités de micro-niveau et le système de macro-niveau. Mais cette propriété de terminaison peut être jugée comme secondaire et nous pouvons l'inspecter de manière expérimentale dans la plupart des cas, comme dans les domaines de l'optimisation locale ou les approches par algorithmes fourmis.

Enfin, notre approche peut être brièvement comparée à des approches exitantes choisies. Les plus populaires pour résoudre les DCSP sont les algorithmes AWCS et ABT [YOK 98] ; mais elles possèdent certains points faibles comme le souligne [BES 01]. Finalement, les CSP distribués et dynamiques ont été définis pour ajouter des possibilités de mises à jour en cours d'exécution. Dans ce cadre, la prise de décision est effectuée par une exploration de graphes de contraintes. Plusieurs algorithmes de recherche ont été définis, comme le DisDB, qui est complet et qui termine [BES 01]. Néanmoins, ces approches, et les algorithmes de résolution associés ne considèrent que la distribution de variables et leur principale priorité est la complétude et la correction.L'autre famille de méthodes de résolution de CSP est influencée par l'optimisation et les métaheuristiques et considère les problèmes d'un point de vue local. Mais cette notion de *local* est relative à l'espace de recherche

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

et non aux parties composant le système. La recherche tabou considère les états du système (une affectation donnée de valeurs aux variables) et leur voisinage, i.e. les états qui sont directement accessibles en appliquant une action. L'idée est d'explorer l'état qui optimise la fonction du système. Cette recherche est restreinte en évitant les états passés (les tabous) [GLO 97]. Néanmoins, ces méthodes restent globales, dans le sens où elles considèrent le système comme un tout et calculent l'énergie globale du système pour explorer l'espace des possibilités.Enfin, une méthode prometteuse pour résoudre des CSP d'un point de vue réellement local que sont les approches par algorithmes fourmis ou multi-agent à la ERA, qui fournissent un processus de résolution uniquement grâce à des comportements locaux affectés aux parties du système [DOR 96, LIU 02]. En utilisant des mécanismes comme la stigmergie et en simulant le collectif, le système global est capable d'atteindre un état optimal –à condition d'avoir bien définis les nombreux paramètres entrant en jeuNotre approche est assez proche des dernières, mais, dans le cas présenté, ne nécessite pas de comportements stochastiques : la réaction face aux marqueurs est déterministe. Pour comparer relativement à une appproche réactive multi-agent, il est aussi possible de considérer basiquement la coopération comme un comportement d'agression-fuite : plus une reine est "agressée" par ses congénères, plus elle va chercher à se placer dans une nouvelle case où elle l'est moins, et tenter de faire "fuir" celles qui peuvent l'attaquer [FER 95]. Mais cet algorithme aboutit à des solutions sous-optimales (bien connues et testées), où les reines retombent cycliquement dans des configurations identiques. La coopération passe outre la sous-optimalité, car elle impose à une reine d'aider sa congénère la plus contrainte localement ; mais aussi d'augmenter ce nombre de contraintes si aucune action ne peut le réduire.

7. Conclusion

Dans cette article, nous avons présenté une approche basée sur l'auto-organisation coopérative pour résoudre des problèmes de décision distribuée, et plus précisément des CSP. Ces problèmes ont été reformulés en utilisant une terminologie multi-agent. Pour illustrer ce travail, l'exemple du problème des N reines a été implémenté et commenté. Cette implémentation montre des résultats prometteurs en termes d'adaptation et de robustesse lorsque le système est sujet à des perturbations environnementales. Cette implémentation soulève aussi des points de discussion comme la généralisation de l'algorithme proposé ou le besoin de mémoire, lorsque les agents nécessitent des critères de décision plus complexes pour trouver l'organisation adéquate. Ces problématiques traversent les domaines de la recherche locale et des métaheuristiques pour implémenter des processus de décisions optimisés. La coopération semble être un critère pertinent de méta-niveau, mais qui requiert plus d'études concernant la charge de mémoire, la complexité, voire la terminaison. Ce dernier point –même si obtenu expérimentalement– peut devenir un bon argument d'utilisation des systèmes multi-agents adaptatifs par auto-organisation et représente notre perspective principale.

8. Bibliographie

[BER 03] BERNON C., CAMPS V., GLEIZES M.-P., PICARD G., « Designing Agents' Behaviors and Interactions within the Framework of ADELFE Methodology », *4^{th} International*

Published in *Journées Francophones sur les Systèmes Multi-Agents (JFSMA'05)*, à Calais du 23 au 25 novembre 2005, pages 161-174. Hermès-Lavoisier, 2005.

Workshop on Engineering Societies in the Agents World (ESAW'03), vol. 3071 de *LNCS*, 2003, p. 311-327.

[BES 01] BESSIÈRE C., MAESTRE A., MESEGUER P., « Distributed Dynamic Backtracking », *Workshop on Distributed Constraints at IJCAI'01*, 2001.

[CAM 99] CAMPS V., GLEIZES M.-P., GLIZE P., « A Theory of Emergent Computation Based on Cooperative Self-Organization for Adaptive Artificial Systems », *4^{th} European Congress of Systems Science, Valencia*, 1999.

[CAP 03] CAPERA D., GEORGÉ J., GLEIZES M.-P., GLIZE P., « The AMAS theory for complex problem solving based on self-organizing cooperative agents », *1^{st} International TAPOCS Workshop at IEEE 12th WETICE*, IEEE, 2003, p. 383-388.

[DOR 96] DORIGO M., MANIEZZO V., COLORNI A., « Ant System : Optimization by a Colony of Cooperating Agents », *IEEE Transactions on Systems, Man and Cybernetics- Part B : Cybernetic*, vol. 26, n° 1, 1996, p. 29-41.

[FER 95] FERBER J., *Les système multi-agents : Vers une intelligence collective*, InterEditions, 1995.

[GEO 03] GEORGÉ J.-P., GLEIZES M.-P., GLIZE P., RÉGIS C., « Real-time Simulation for Flood Forecast : an Adaptive Multi-Agent System STAFF », *AISB'03 symposium on Adaptive Agents and Multi-Agent Systems(AAMAS'03)*, 2003.

[GLO 97] GLOVER F., LAGUNA M., *Tabu Search*, Kluwer, 1997.

[KOH 01] KOHONEN T., *Self-Organising Maps*, Springer-Verlag, 2001.

[LIU 02] LIU J., JING H., TANG Y. Y., « Multi-agent oriented constraint satisfaction », *Artificial Intelligence*, vol. 136, n° 1, 2002, p. 101-144.

[MOD 03] MODI P. J., SHEN W., TAMBE M., YOKOO M., « An Asynchronous Complete Method for Distributed Constraint Optimization », *2^{nd} International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'03)*, 2003, p. 161–168.

[PIC 04a] PICARD G., « Cooperative Agent Model Instantiation to Collective Robotics in ADELFE », *5^{th} International Workshop on Engineering Societies in the Agents World (ESAW'04), 20-22 October 2004, Toulouse, France*, 2004.

[PIC 04b] PICARD G., GLEIZES M.-P., « The ADELFE Methodology – Designing Adaptive Cooperative Multi-Agent Systems », *Methodologies and Software Engineering for Agent Systems (Chapter 8)*, Kluwer Publishing, 2004, p. 157-176.

[PIC 05] PICARD G., « Résolution d'emploi du temps dynamique et distribuée par auto-organisation coopérative », *7^{emes} Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA'05), Plate-forme AFIA, Nice*, 2005.

[RUS 95] RUSSEL S., NORVIG P., *Artificial Intelligence : a Modern Approach*, Prentice-Hall, 1995.

[SOC 02] SOCHA K., KNOWLES J., SAMPELS M., « A MAX-MIN Ant System for the University Timetabling Problem », *3^{rd} International Workshop on Ant Algorithms, ANTS'02*, vol. 2463 de *LNCS*, 2002, p. 1-13.

[YOK 98] YOKOO M., DURFEE E., ISHIDA Y., KUBAWARA K., « The Distributed Constraint Satisfaction Problem : Formalization and Algorithms », *IEEE Transactions on Knowledge and Data Engineering*, vol. 10, 1998, p. 673-685.