La criticité comme moteur de la coopération dans les systèmes multi-agents adaptatifs

Teddy Bouziat, Stéphanie Combettes, Valérie Camps, Pierre Glize

La criticité comme moteur de la coopération dans les systèmes multi-agents adaptatifs

T. Bouziat S. Combettes V. Camps P. Glize
Teddy.Bouziat@irit.fr Stephanie.Combettes@irit.fr Valerie.Camps@irit.fr Pierre.Glize@irit.fr

Institut de Recherche en Informatique de Toulouse,
Université Paul Sabatier - Toulouse III, France

Résumé

Dans l'approche par AMAS (Adaptive Multi-Agent System), la coopération est définie comme attitude sociale de l'agent et permet de détecter, de résoudre, voire d'anticiper les contraintes subies par le système au travers du concept de Situations Non Coopératives. Nous proposons ici une stratégie de décision pour l'agent coopératif, basée uniquement sur la "criticité", afin d'aider ses voisins les plus contraints. Cette métrique permet de prioriser les actions des agents et de répartir de manière coopérative la charge pesant sur le système, évitant ainsi qu'un ou plusieurs agents ne puisse(nt) accomplir sa/leur tâche et donc nuire à la performance et à la robustesse du système. Afin de montrer l'apport de notre approche, cette stratégie de décision est évaluée au travers de différentes expérimentations sur un système de transport de ressources par plusieurs robots.

Mots-clés : *Adaptation, Coopération, Emergence, Criticité*

Abstract

In the Adaptive Multi-Agent System (AMAS) approach, the cooperation is defined as a social attitude of an agent and enables to detect, to solve or even to anticipate constraints undergone by the system through the concept of Non Cooperative Situations. We propose here a decision strategy for the cooperative agent, only based on the "criticality", which enables to help its most constrained neighbours. This metric enables to prioritize agent actions and to distribute in a cooperative way the load of the system in order to avoid that one or several agents accomplish its/their tasks, and so to compromise performance and robustness of the system. In order to show the contribution of our approach, this decision strategy is evaluated through different experimentations on a resource transportation system by several robots.

Keywords: *Adaptation, Cooperation, Emergence, Criticality*

1 Introduction

De nos jours les systèmes à concevoir sont de plus en plus complexes, souvent plongés dans un environnement dynamique et ouvert ; ils doivent alors faire preuve de robustesse face aux situations imprévues induites par cette dynamique et survenant en cours de fonctionnement. Ils doivent donc être dotés de capacités à s'(auto-)-adapter aux conditions imprévues. La coopération est une notion importante qui a fait l'objet de nombreuses études dans les systèmes multi-agents et a montré sa pertinence pour la résolution de problèmes mettant en oeuvre de nombreuses entités. L'approche par AMAS a introduit une vision originale de la coopération : la coopération en tant qu'attitude sociale. Cette originalité provient du théorème de l'adéquation fonctionnelle [4] [12] qui énonce que pour faire converger un système artificiel alors que ses attracteurs ne sont pas connaissables *a priori* (du fait de leur complexité), il est nécessaire de donner aux parties (agents) qui le composent un comportement coopératif. Ces agents doivent, de leur point de vue local détecter, supprimer voire anticiper toute situation non coopérative [1]. Ce comportement coopératif doit subsumer leur comportement local et en ce sens devient une attitude sociale. A des fins de généralisation de l'approche mais aussi pour mieux mettre en oeuvre la coopération, nous proposons de définir la coopération à l'aide d'une nouvelle et unique métrique, appelée "criticité". Celle-ci permet à l'agent de juger localement la pertinence de ses actions en terme de coopération afin qu'il puisse aider ses voisins les plus contraints. Après un état de l'art et un rappel de la coopération selon l'approche par AMAS, nous introduisons puis définissons la notion de criticité. Nous instancions cette notion au problème de transports de ressources par plusieurs robots. Afin de mettre en évidence l'apport de la criticité, nous proposons d'étudier son influence sur trois versions du système : 1) les

agents n'utilisent pas de mécanisme de coopération, 2) les agents se servent de la criticité comme moteur de la coopération et 3) les agents sont capables d'apprendre leur criticité. Nous montrons enfin, avant de conclure, qu'un agent peut apprendre à déterminer sa criticité grâce aux feedbacks que l'environnement lui retourne suite aux actions qu'il a réalisées, ces retours indiquant à l'agent s'il est réellement plus (ou moins) critique que ce que lui-même pensait être.

2 Etat de l'art

La coopération est une notion étudiée depuis longtemps dans les SMA. Davis et Smith [7] ont analysé les différentes manières dont les agents coopèrent pour atteindre un objectif commun en faisant du partage de tâches ou de résultats. L'aspect dynamique du rôle que peut jouer un agent au cours du processus de résolution donne une grande importance au modèle du réseau contractuel proposé. Cammarata [3] s'est intéressée aux problèmes de répartition de tâches, de choix des agents et de mise en oeuvre de la communication. Dans son modèle pour le trafic aérien elle a comparé différents moyens de résolution de conflits éventuels. Cohen et Levesque [6] et Galliers [10] travaillent sur la coopération en termes d'intentions et d'adoption de buts. Les agents ont des croyances et des savoirs. Ils coopèrent intentionnellement en adoptant les objectifs des autres. Thierry Bouron [2] a défini un ensemble d'indices de coopération permettant de caractériser et de quantifier la coopération. Ils concernent la coordination d'actions, le degré de parallélisation, le partage de ressources, la robustesse, la non redondance des actions et la non persistance des conflits. Hogg et al. [13] ont également montré que les problèmes, notamment distribués, sont résolus plus rapidement quand les agents travaillent en coopération. Les auteurs ont défini la coopération comme un ensemble d'agents qui interagissent entre eux en s'échangeant des informations de communication (ou "*hints*") pour résoudre le problème. Cependant ces informations peuvent être parfois incorrectes et par conséquent modifier le comportement de l'agent qui reçoit l'information. Les auteurs traitent plus particulièrement des problèmes discrets de satisfaction de contraintes et montrent l'influence bénéfique de la coopération : "*The power of cooperation*". De même les travaux de Sekaran et al. [14] ont mis en évidence le fait que si les agents coopèrent, la performance générale du système est meilleure. Leur but est d'étudier si, dans une situation particulière, un agent doit accepter ou refuser une demande de coopération. Leur système est basé sur le concept de comportement réciproque. Ferber [9] considère la coopération selon deux perspectives : la coopération comme attitude intentionnelle et la coopération du point de vue de l'observateur. Dans la première perspective, la coopération est une attitude des agents, qui après avoir identifié et adopté un but commun décident de travailler ensemble. Selon la seconde perspective, la coopération est une qualification de l'activité des agents par un observateur extérieur qui n'aurait pas accès aux états mentaux des agents. Il considère enfin que plusieurs agents coopèrent (ou sont dans une situation de coopération) si l'une des deux conditions suivante est vérifiée : (*i*) l'ajout d'un nouvel agent permet d'accroître différentiellement les performances du groupe ; (*ii*) l'action des agents sert à éviter ou à résoudre des conflits potentiels ou actuels.

Dans l'approche par AMAS (Adaptive Multi-Agent System), contrairement à [9], la coopération est vue comme une **attitude sociale** à laquelle l'agent ne peut déroger. La définition proposée, repose sur trois métarègles locales à l'agent, que le concepteur doit instancier selon le problème qu'il doit résoudre [4] [12]. Lorsqu'un agent est dans un état de coopération, il est situé au bon endroit dans l'organisation du système. La réalisation de sa fonction locale est pertinente pour lui et pour son voisinage avec qui il entretient des interactions pertinentes et utiles. Dans le cas contraire, il est face à une situation non coopérative (SNC) et doit agir pour revenir à un état de coopération. La suppression de SNC constitue le moteur auto-organisationnel de l'approche, qui repose sur trois mécanismes ("*tuning*", " *réorganisation*", "*l'évolution*") [5] qui entraînent une modification de la fonction globale du système en s'adaptant aux situations imprévues survenant en cours de fonctionnement. Le concepteur d'un AMAS doit alors doter les agents de son système, d'un comportement nominal mais aussi (et surtout) de règles de comportement (mécanismes) leur permettant de détecter, réparer, voire anticiper des SNCs. Il peut s'aider pour cela de la méthodologie de conception ADELFE [1].

Les évolutions technologiques sans cesse croissantes rendent nécessaire la conception de systèmes de plus en plus complexes, ouverts, plongés dans un environnement dynamique et pour

lesquels il est difficile (voire impossible) de connaître et définir, dès la phase de spécification, l'objectif global attendu (comme dans les systèmes ambiants, ou l'internet des objets). Dans ce cadre-là, plusieurs systèmes (éventuellement conçus par différents concepteurs) peuvent être amenés à interagir pour répondre à un problème particulier. Ce nouveau besoin commence à apparaître au niveau des AMAS ; répondre à un problème peut nécessiter la conception de plusieurs AMAS conçus indépendamment mais interdépendants, formant alors un meta-AMAS. Ces AMAS doivent alors coopérer entre eux conformément à l'approche. Nous pensons que la métrique de criticité peut être utilisée pour mettre en oeuvre la coopération entre AMAS. Chaque AMAS ayant sa propre définition de la criticité, l'échange de cette métrique sans traitement préalable peut ne pas être pertinent. Il s'agit alors d'apprendre à la "normer". Ce papier n'abordera pas ce point mais en première étape nous souhaitons montrer ici l'intérêt de la criticité, pour mettre en oeuvre la coopération entre agents d'un AMAS.

3 La criticité : moteur de la coopération

La criticité d'un agent peut être définie comme la distance qui sépare son état courant de l'état dans lequel son objectif local est atteint. Ainsi plus une criticité est grande, plus l'agent est loin d'atteindre son but et donc plus il est en difficulté. Cette métrique permet notamment à un agent coopératif de choisir l'action la plus coopérative à chaque instant. L'action la "plus coopérative" signifie celle qui aide prioritairement le voisinage de l'agent, si ce voisinage est en plus grande difficulté que l'agent devant agir, sans pour autant que cet agent se retrouve dans une situation plus critique que son voisinage une fois son action réalisée [11]. Par exemple, lorsque deux agents sont en conflit, la résolution de ce conflit doit être la plus coopérative pour les deux agents. Elle doit prioritairement aider celui qui est le plus en difficulté - et ce parfois au détriment ponctuel de l'autre agent. Le problème consiste alors à savoir comment déterminer l'agent qui est en plus grande difficulté.

La notion de criticité est donc introduite pour permettre à un agent de prendre localement la meilleure décision pour lui et pour le collectif. Notre définition de la criticité repose sur la distance entre l'état courant d'un agent et son objectif local. Le terme *distance* est ici volontairement générique pour permettre au concepteur de l'AMAS de l'instancier à son problème. Il peut en effet s'agir d'une distance temporelle, spatiale ou encore logique. La criticité est une fonction définie à l'aide des différentes perceptions de l'agent ; il est donc nécessaire de (*i*) choisir les paramètres les plus pertinents issus des perceptions de l'agent ; (*ii*) définir l'influence de ces paramètres dans le calcul de sa criticité. Ces deux tâches ne sont pas aisées et demandent une phase de *tuning* afin que le calcul de la criticité d'un agent réponde au mieux à son objectif mais permette aussi à cet agent de juger au mieux son état de difficulté par rapport à celui de ses voisins. Une criticité mal calculée peut conduire en effet à des systèmes qui ne sont pas coopératifs mais plutôt égoïstes ou altruistes.

3.1 Définition de la criticité

Plus formellement, supposons un agent r_i devant résoudre une fonction partielle f_i pour atteindre son objectif local. Nous définissons la criticité C de l'agent r_i à l'instant t comme une fonction F dépendante d'un certain nombre n ($n \in \mathbb{N}$) de paramètres p_i ($0 < i < n$) issus des perceptions de r_i :

$$C_{r_i}(t) = F(p_1(t), ..., p_n(t))$$

Le concepteur doit alors déterminer pour r_i les différents paramètres p_i et définir la fonction F qui modélise l'influence des p_i sur l'obtention de f_i. Malheureusement, l'influence des différents paramètres peut être inconnu du concepteur. C'est pourquoi nous proposons un mécanisme d'apprentissage de F. Cet apprentissage est réalisé en cours de fonctionnement et tient compte des feedbacks reçus de l'environnement suite aux actions réalisées par l'agent.

3.2 Apprentissage de la criticité

F pouvant être une fonction difficile à trouver ou analytiquement inconnue, nous proposons de définir un algorithme permettant d'approximer localement linéairement cette fonction. Cet algorithme s'appuie sur l'approche par AMAS et a pour objectif de trouver l'influence de chaque paramètre (issu des perceptions de l'agent r_i) sur le calcul de la criticité C_{r_i} de l'agent r_i. L'influence d'un paramètre p_i est représenté par un poids ω_i dans la formule ci-dessous.

$$C_{r_i}(t) = \sum(\omega_i * p_i(t))$$

L'AMAS permettant de déterminer ces influences, est composé d'agents *critère*, chacun

rattaché à un poids ω_i. Le rôle d'un agent *critère* est de faire varier le poids ω_i auquel il est rattaché en fonction des feedbacks de l'agent r_i. Chaque agent *critère* a pour objectif d'ajuster progressivement son poids jusqu'à tendre vers une valeur correspondant à son influence réelle. Les feedbacks transmis à l'AMAS par le biais de r_i (qui les a lui même reçu de son voisinage suite à une action réalisée) peuvent être de deux types : '+' si la criticité a été sous-estimée ou '-' si elle est surestimée par l'agent *critère*. Une fois le feedback reçu, l'agent r_i indique aux différents agents *critère* leur *implication*. Cette *implication* est un paramètre pris en compte pour augmenter/diminuer le poids d'un agent *critère* en fonction de l'importance qu'a cet agent dans le calcul de la criticité. L'*implication* permet de personnaliser les comportements des agents : leurs dynamiques seront potentiellement différentes.

Le comportement d'un agent *critère* tient compte d'une *tendance*, qui a l'initialisation de l'agent vaut 0. La *tendance* exprime une motivation pour augmenter ou diminuer son poids ; elle tient compte des expériences passées qui lui indiquent s'il vaut mieux qu'il augmente ou diminue son poids, autrement dit elle représente la force avec laquelle l'agent va tenir compte du prochain feedback. Si un agent a eu fortement tendance à baisser son poids aux cycles précédents, alors il lui faudra plusieurs feedbacks positifs avant qu'il ne décide d'augmenter son poids. Ceci est détaillé dans l'algorithme 1.

Algorithme 1 : Algorithme d'apprentissage de la criticité pour un agent *critère*

```
Calculer son implication ;
si feedback =' +' alors
    si tendance ≥ 0 et tendance ≥ seuil alors
        Augmenter ω_i en fonction de son implication
    fin
fin
si feedback =' −' alors
    si tendance ≤ 0 et | tendance | ≥ seuil alors
        Diminuer ω_i en fonction de son implication
    fin
fin
Mettre à jour sa tendance ;
```

4 Instanciation de la criticité

Nous avons instancié la notion de criticité et mis en oeuvre son apprentissage dans le cadre d'un problème de transport de ressources par des robots. Ce système, nommé CoCaRo (Color Carrier Robot), a été développé à l'aide de GAMA [8]. GAMA est une plateforme permettant de modéliser et simuler des SMA de grande échelle, facile d'utilisation et intégrant des outils d'analyse et de visualisation performants. Pour évaluer l'apport la criticité en terme de coopération, trois versions du système ont été développées : une première où les agents n'utilisent pas de mécanisme de coopération, une seconde où les agents se servent de la criticité comme moteur de la coopération et enfin une troisième où les agents sont coopératifs et apprennent eux-mêmes leur criticité. Les parties 4.2 et 4.3 présentent les modules communs aux trois systèmes et les parties 4.4, 4.5 et 4.6 décrivent le système respectivement sans coopération, avec coopération et avec apprentissage de la criticité.

4.1 CoCaRo : Système de transport de ressources

CoCaRo est un AMAS pour le transport de boites de couleurs. Dans ce système, l'objectif des robots consiste à rapporter toutes les boites de couleurs (rouges, vertes, ou bleues) dans des zones spécifiques de l'environnement, appelées « nids ». Chaque nid ne peut accueillir qu'une couleur de boite (rouge, verte ou bleue). Le système est composé de robots de couleur (rouge, vert, ou bleu) qui possèdent les compétences nécessaires à la résolution du problème (perceptions, actions, communication, mécanisme de coopération, etc.). Chaque robot est doté d'une quantité initiale d'énergie qu'il consomme à chaque pas de déplacement. Chaque robot doit trouver, transporter et déposer une boite dans le nid ayant la même couleur que la boîte. A chaque dépôt de boite, le robot reçoit une récompense sous forme d'énergie lui permettant de rester efficace plus longtemps. La valeur de la récompense dépend de la couleur de la boite ramenée par rapport à la couleur du robot. Nous allons maintenant détailler l'environnement de CoCaRo et les agents composant le système.

4.2 L'environnement

L'environnement est composé de 3 types d'objets différents :

1. **La grille de déplacement** : Elle est composée arbitrairement de 2500 cellules carrées (50x50) qui peuvent contenir plusieurs composants (boite, robot, nid) ou être vides.
2. **Les nids** : Ce sont les objets dans lesquels les robots déposent les boites. Il en existe 3 : un rouge, un vert, un bleu et sont disposés à équidistance les uns des autres afin de ne pas introduire de biais lié à la proximité de deux nids dans les simulations.
3. **Les boites :** Elles sont transportées et déposées par les robots dans les nids. Elles sont de trois types distincts : rouge, vert et bleu, et apparaissent de manière aléatoire sur la grille à intervalle de temps régulier.

4.3 Perceptions et attributs des agents

CoCaRo est composé de 3 types d'agents différents : les robots rouges, les robots verts et les robots bleus. Chaque type d'agent est capable de transporter tous types de boites mais un robot obtient une meilleure récompense s'il ramène une boite de sa couleur. Nous décrivons maintenant les perceptions et attributs communs aux différents cas d'utilisation étudiés.

– **Perceptions** : Les agents *robots* ont la capacité de percevoir l'ensemble des objets (autres robots et/ou boites) qu'il perçoit autour de lui et les coordonnées de sa position sur la grille. Ils connaissent les positions des différents nids à tout instant.
– **Attributs** : L'agent possède les attributs suivants : (*i*) un niveau d'énergie, (*ii*) une vitesse de déplacement, (*iii*) une criticité et (*iv*) une criticité anticipée.

Le niveau d'énergie : l'agent *robot* se déplace sur la grille grâce à l'énergie qui lui est fournie initialement ou qu'il récupère lors du dépôt des boites dans les nids. Ce niveau baisse d'une valeur $conso$ à chaque pas de temps par déplacement. Le niveau d'énergie Ne_{r_i} est une fonction définie pour chaque agent de la manière suivante : Soit B l'ensemble des boites de l'environnement, soit l'agent *robot* r_i ayant rapporté le sous-ensemble $B_{r_i} = \{b_k \in B\}$ de boites entre les temps 0 et $t \in \mathbb{N}^+$,

$$Ne_{r_i}(t) = \begin{cases} N_{r_i}(t) & \text{si } 0 < N_{r_i}(t) < Max_{Ne} \\ Max_{Ne} & \text{si } N_{r_i}(t) \geqslant Max_{Ne} \\ 0 & \text{sinon} \end{cases}$$

avec Max_{Ne} représentant le niveau maximal d'énergie, $N_{r_i}(t) = N_i + \sum_{B_{r_i}} rec(b_k) - conso*t$. N_i représente le niveau d'énergie au début de la simulation. rec_{r_i} est la fonction récompense de l'agent r_i. La valeur de récompense dépend de la couleur de la boite rapportée par l'agent *robot*. Si ce dernier rapporte une boite de sa couleur ($c(b_k) = c(r_i)$), alors la récompense sera plus importante :

$$rec_{r_i}(b_k) = \begin{cases} \frac{2}{3} * Max_{Ne} & \text{si } c(b_k) = c(r_i) \\ \frac{1}{3} * Max_{Ne} & \text{sinon} \end{cases}$$

$conso$ représente la consommation d'énergie par pas de temps. Elle est multipliée par le temps de façon à évaluer la dépense d'énergie jusqu'au temps t.

La vitesse de déplacement : à chaque pas de temps, le niveau d'énergie est décrémenté de $conso$ unités. C'est sur cette métrique que se base le calcul de la vitesse de déplacement. Celle-ci permet de représenter l'efficacité avec laquelle l'agent *robot* se déplace en fonction de son niveau d'énergie :

$$VitesseDplct_{r_i}(t) =$$

$$\begin{cases} 1 & \text{si } \frac{2}{3} * Max_{Ne} \leq Ne_{r_i}(t) \leq Max_{Ne} \\ \frac{1}{2} & \text{si } \frac{1}{3} * Max_{Ne} \leq Ne_{r_i}(t) < \frac{2}{3} * Max_{Ne} \\ \frac{1}{3} & \text{si } 0 < Ne_{r_i}(t) < \frac{1}{3} * Max_{Ne} \\ 0 & \text{sinon} \end{cases}$$

La criticité : la criticité, indiquant le niveau de difficulté d'un agent *robot*, est définie en fonction de son niveau d'énergie. En effet, l'agent est moins efficace lorsqu'il est moins chargé et donc se dégrade plus rapidement.
Soit l'agent *robot* r_i, sa criticité C_{r_i} est une fonction temporelle calculée de la manière suivante :

$$C_{r_i}(t) = Max_{Ne} - Ne_{r_i}(t)$$

La criticité est donc un entier variant de 0 (agent non critique) à Max_{Ne} (agent très critique).

La criticité anticipée : C'est une fonction permettant à l'agent de connaître la criticité qu'il obtiendra une fois la boite b_k déposée dans son nid. L'agent peut ainsi choisir la meilleure boite la pour lui.

Soit l'agent *robot* r_i, la criticité anticipée $CA_{r_i}(b_k)$ de r_i pour la boite b_k est calculée de la manière suivante :

$$CA_{r_i}(b_k, t) = Max_{Ne} - Ne_a(b_k, t)$$

avec $Ne_a(b_j, t) =$

$$\begin{cases} N_{r_i}(t+t_d) + rec_{r_i}(b_k) & \text{si } 0 < . < Max_{Ne} \\ Max_{Ne} & \text{si } . \geqslant Max_{Ne} \\ 0 & \text{sinon} \end{cases}$$

t_d correspond au temps qu'il faut à l'agent r_i pour aller de sa position courante à la boite b_k, puis au nid associé pour y déposer cette boite :

$t_d = \frac{distance(r_i,b_k)+distance(b_k,Nid)}{VitesseDplct_{r_i}(t)}$

4.4 Système 1 : Agents non coopératifs

Dans ce système, aucun mécanisme de coopération **entre** les agents *robot* n'est utilisé. Grâce à la criticité, un agent cherche la boite la plus intéressante pour lui de sorte à avoir une quantité d'énergie maximale pour rapporter un maximum de boites. Les agents ne sont pas coopératifs entre eux et ne cherchent pas à s'aider en s'échangeant des boites, même si cela pourrait s'avérer avantageux pour eux.

Actions Un agent *robot* peut *se_déplacer*, *déposer*, *prendre*, *aller_nid*. Tant qu'il n'a pas détecté de boites autour de lui, il se déplace à l'aide d'un tirage de Monte-Carlo. L'agent peut être dans plusieurs états :
– $carried$: l'agent porte une boite ;
– $target$: l'agent a ciblé une boite ;
– $onPosBox$: l'agent se trouve sur la même case que sa boite cible ;
– $onPosNest$: l'agent se trouve sur la même case que le nid de la boite qu'il porte.
L'agent choisit les différentes actions en fonction de son état courant et de ses perceptions. Le tableau 4.4 montre la correspondance entre actions et perceptions.

Perceptions	Actions
$\neg target \wedge \neg carried$	$se\_deplacer$
$target$	$aller_nid(targeted\_box)$
$target \wedge onPosBox$	$prendre(targeted\_box)$
$carried \wedge onPosNest$	$deposer(carried\_box)$

TABLE 1 – Actions des agents

Algorithme de décision L'objectif d'un agent *robot* étant d'avoir la plus grande quantité d'énergie au cours du temps, nous détaillons l'algorithme de décision lui permettant de maximiser son niveau d'énergie. L'algorithme 2 utilise la criticité et la criticité anticipée. Un agent peut changer de boite si lors du retour au nid, il en trouve une lui permettant d'avoir une quantité d'énergie supérieure à celle qu'il aurait avec la boite qu'il transporte.

Algorithme 2 : Décision pour un agent r_i non-coopératif

```
1  tant que Ne_{r_i}(t) ≠ 0 faire
2      si carried alors
3          CA_courante = CA_{r_i}(boite_portee, t)
4      fin
5      si target alors
6          CA_courante = CA_{r_i}(boite_ciblee, t)
7      fin
8      Mettre à jour boites_visibles ;
9      pour tous les b_k ∈ boites_visibles faire
10         si CA_{r_i}(b_k, t) < CA_courante alors
11             CA_courante = CA_{r_i}(b_k, t) ;
12             boite_ciblee = b_k ;
13             si carried alors
14                 deposer(boite_courante) ;
15             fin
16         fin
17     fin
18 fin
```

Algorithme 3 : Décision pour l'agent *robot* coopératif r_i

```
1 si possesseur(b_k) ≠ null alors
2     EnvoiRequeteCoop(r_j, C_{r_i}(t), CA_{r_i}(b_k, t))
3 sinon
4     comportement_nominal() ;
5 fin
```

4.5 Système 2 : Agents coopératifs

Les agents *robot* sont à présent coopératifs. Le mécanisme de coopération leur permet de s'échanger des boites en fonction de leurs criticités instantanée et anticipée.

Algorithme de décision Lors de la détection d'une boite b_k, l'agent *robot* r_i vérifie si celle-ci appartient déjà à un agent *robot* r_j. Si tel est le cas, r_i envoie un message à r_j en lui fournissant sa criticité et sa criticité anticipée pour la boite b_k. Ce processus est introduit après la ligne 10 de l'algorithme 2 et détaillé dans l'algorithme 3. Ces informations servent de paramètres d'entrées aux mécanismes de coopération pour r_j qui choisira soit de garder la boite soit de l'échanger. Ce processus est introduit

après la ligne 18 de l'algorithme 2 et est détaillé ci-après.

Mécanisme de coopération Le mécanisme de coopération d'un agent *robot* est mis en oeuvre dans la fonction **traitement_requete_coop()** détaillée dans l'algorithme 4. L'agent r_j possédant une boite b_k traite à chaque pas de temps t, la requête de coopération qu'il a reçue à $t-1$. Etant coopératif, r_j doit déterminer si l'émetteur r_i de la requête est plus critique que lui, si c'est le cas, r_j lui donne sa boite, sinon il la garde. Pour cela, r_j compare sa criticité à celle de r_i et vérifie à l'aide de leur criticité anticipée respective si l'échange ne provoque pas sur le long terme une élévation de la criticité.

Algorithme 4 : *traitement_requete_coop()* de l'agent r_j transportant b_k

```
1  si C_rj(t) < C_ri(t) alors
2  |  si CAr_i(b_k, t) < C_ri(t) alors
3  |  |  accepter_echange();
4  |  |  deposer(b_k);
5  |  sinon
6  |  |  refuser_echange() ;
7  |  fin
8  sinon
9  |  si CAr_j(b_k, t) < C_rj(t) alors
10 |  |  refuser_echange() ;
11 |  sinon
12 |  |  accepter_echange() ;
13 |  |  deposer(b_k);
14 |  fin
15 fin
```

4.6 Système 3 : Apprentissage de la criticité

Nous avons vu que la criticité est directement liée au niveau d'énergie, lui-même lié au niveau de récompense pour chaque type de boite (cf. section 4.3). Nous avons instancié l'algorithme d'apprentissage présenté en section 3.2 (algorithme 1) à CoCaRo, avec des agents ne possédant pas les bonnes valeurs de récompenses. Les agents ont alors une mauvaise connaissance de leur niveau d'énergie et donc de leur criticité.

Cet algorithme permet l'apprentissage par approximation linéaire locale de la fonction criticité qui, dans notre système, est calculée à partir du niveau d'énergie, lui-même étant une fonction linéaire : $\sum_{B_{r_i}} rec(b_k) = \sum_{c \in C} nb\_boites_{r_i}(c) * rec_{r_i}(c)$ avec $nb\_boites_{r_i}(c)$ le nombre de boites de couleur c rapportées par le robot r_i et $rec_{r_i}(c)$ la récompense associée au dépôt d'une boite de couleur c. Cet algorithme est instancié pour chaque robot r_i sachant que (*i*) les différents poids w_i à apprendre sont les récompenses $rec_{r_i}(c)$ et (*ii*) une perception $p_i(t)$ représente le nombre de boites $nb\_boites_{r_i}(c)$ d'une même couleur rapportées par r_i. Les trois niveaux de récompense à apprendre impliquent d'instancier trois agents *critère* par agent r_i.

Les feedbacks sont envoyés à l'agent *critère* lorsque l'agent r_i change (ou pense changer) de vitesse de déplacement (cf. 4.3). Par exemple, s'il passe d'une vitesse de déplacement de 1 à $\frac{1}{2}$ alors qu'il ne s'y attendait pas, cela signifie qu'il a surestimé jusqu'à présent les récompenses et doit donc les diminuer. Inversement, s'il passe à un niveau supérieur, il doit les augmenter (sous-estimation).

L'*implication* représente l'importance d'un agent *critère* par rapport aux autres. Il est défini comme le produit de la récompense associée à sa couleur par le nombre de boites rapportées de cette couleur. Cela permet de personnaliser les comportements des agents *critère*.

La constante DYN représente la dynamique du système. Si l'environnement est très évolutif, un agent *critère* doit accorder une grande importance au dernier feedback envoyé par r_i. Elle correspond à la longueur de l'historique des feedbacks. Cette constante a été empiriquement fixée à $0,5$, notre système n'ayant pas une grande dynamicité.

La constante $SEUIL$, fixée empiriquement à $0,1$, est la valeur limite à partir de laquelle l'agent *critère* tient compte du feedback courant. Si l'agent r_i a rapporté peu de boites vertes, l'agent *critère* associé aux boites vertes ne tient pas compte du feedback (inférieur au seuil).

5 Expérimentation

Nous avons implémenté puis simulé les 3 systèmes décrits précédemment afin d'évaluer notre approche. Avant de présenter les résultats obtenus, nous expliquons les conditions de réalisation de ces simulations rejouées 10 fois chacune. Chaque système est ensuite évalué en terme d'efficacité et de robustesse à l'aide de 3 métriques : le nombre de robots fonctionnels, la moyenne du niveau d'énergie des agents et le nombre de boites présentes dans l'environnement.

Algorithme 5 : Algorithme d'apprentissage de la criticité pour l'agent *critère* w_i

$PLUS = 1$; $MOINS = -1$; $SEUIL = 0,1$;
$DYN = 0,5$; $PAS = 10$; $tendance = 0$;
$impl = rec_{ri}(c) * nb\_boites_{ri}(c)$;
si $feedback = PLUS$ **alors**
  si $tendance > 0$ ***et*** $tendance > SEUIL$ **alors**
    $rec(c) = rec(c) + implication * PAS$
  fin
sinon si $feedback = MOINS$ **alors**
  si $tendance < 0$ ***et*** $|tendance| > SEUIL$ **alors**
    $rec(c) = rec(c) - implication * PAS$
  fin
fin
$tendance = feedback * impl * DYN + (1 - DYN) * tendance$;

Conditions initiales Les simulations réalisées pour les 3 systèmes sont soumises aux mêmes conditions initiales :

- le nombre de robots est fixé à 90, 30 par couleur ;
- le niveau maximal d'énergie (Max_{N_e}) d'un robot est fixé à 300 ;
- le niveau initial d'énergie (N_i) est fixé à 300 ;
- la consommation d'énergie par pas de temps ($conso$) est fixée à 1 ;
- la perception et la portée de communication d'un robot sont fixées à un rayon de 3 cases tout autour de sa position courante ;
- le nombre de boites apparaissant dans l'environnement est fixé à 1 tous les 3 pas de temps ;
- le placement initial des boites et des robots est le même pour l'ensemble des simulations ;

Dans toutes les figures illustrant les résultats, les courbes de couleur noir représentent le système sans coopération, les courbes de couleur gris clair le système avec coopération et les courbes de couleur gris foncé le système avec coopération et apprentissage de la criticité.

5.1 Environnement sans perturbation

Dans cette simulation, la couleur des boites apparaissant dans l'environnement est choisie aléatoirement et permet de représenter un environnement sans perturbation : tous les types de robots (rouges, verts et bleus) ont la même probabilité de survie au cours du temps. Cette simulation permet d'évaluer l'efficacité des 3 systèmes.

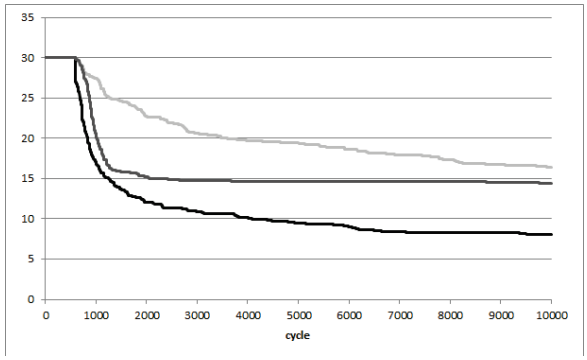

FIGURE 1 – Nombre de robots rouges fonctionnels

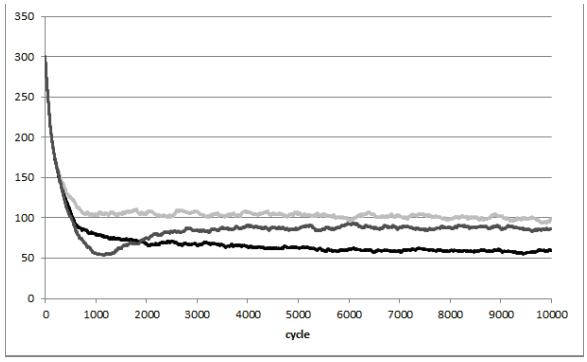

FIGURE 2 – Niveau moyen d'énergie

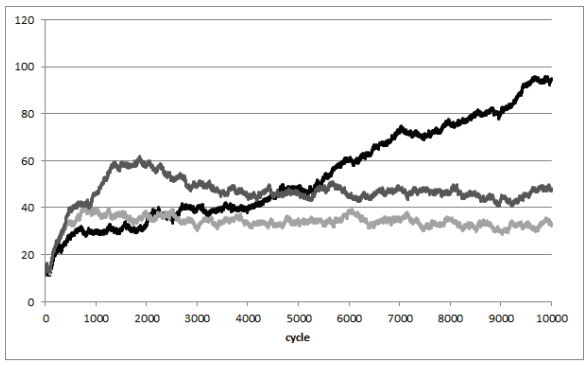

FIGURE 3 – Nombre de boites dans l'environnement

Discussion : La comparaison des 3 systèmes montre clairement que l'échange de criticité permet d'améliorer l'efficacité du système. En effet, la figure 1, présentant seulement le nombre de robots rouges par souci de clarté, montre que le mécanisme de coopération sous-jacent à cet échange permet à un nombre plus important d'agents de survivre. Ainsi le système avec coopération couvre une plus grande partie de la grille et améliore donc directement la capacité du système à détecter les boites. En revanche, l'apprentissage de la criticité entraîne

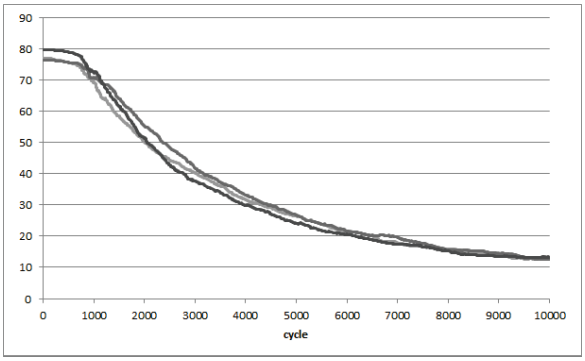

FIGURE 4 – Erreur moyenne sur les récompenses par couleur

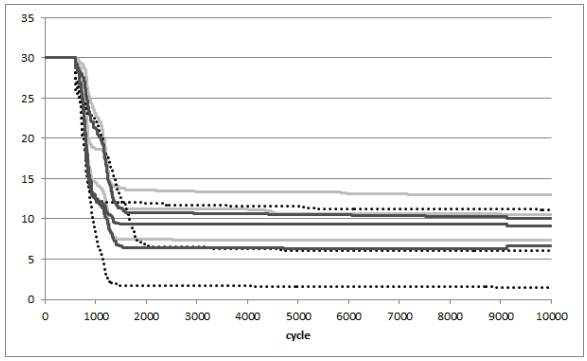

FIGURE 5 – Nombre de robots fonctionnels

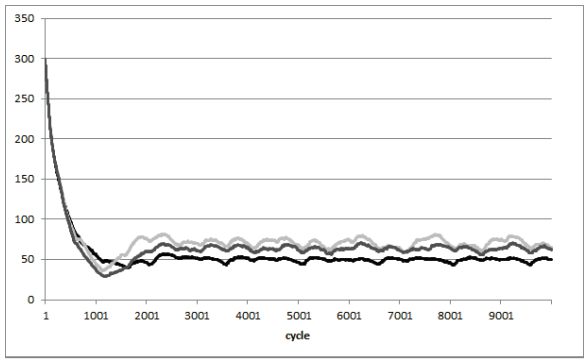

FIGURE 6 – Moyenne des niveaux d'énergie

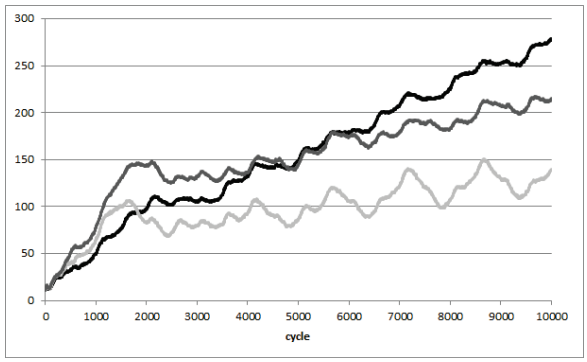

FIGURE 7 – Nombre de boites dans l'environnement

une plus grande perte d'agents que le système avec coopération. Ce point vient tout simplement du fait qu'en début de phase d'apprentissage, le système est loin d'être efficace mais il réussit peu à peu à tendre vers l'optimal. La figure 2 montre que pendant les 2000 premiers cycles de simulation, le niveau d'énergie moyen des robots en apprentissage est en dessous du système sans coopération. La phase d'apprentissage a donc un coût non négligeable pour le système même si l'erreur moyenne sur les récompenses diminue au cours du temps (fig. 4). Le système arrive cependant à stabiliser le nombre de boites dans l'environnement (fig. 3).

5.2 Environnement avec perturbations

La couleur des boites apparaissant dans l'environnement n'est à présent plus aléatoire ; sur une période fixée à 500 cycles, seules les boites d'une même couleur apparaissent et ceci se répète par alternance. Cette simulation permet d'évaluer la robustesse du système car le nombre initial d'agents de chaque type étant le même, le système n'est pas prévu pour être efficace dans ce cas. En effet, l'environnement ne faisant apparaitre des boites que d'une certaine couleur et les robots ayant des récompenses différentes en fonction du type de boites rapportées, certains seront en difficulté sur une période donnée pendant que d'autres ne le seront pas.

Discussion : L'utilisation de la criticité fait clairement apparaitre la coopération entre les agents. Le système avec agents coopératifs ainsi que le système avec apprentissage arrivent à préserver leur diversité en terme de type de robots (fig. 5). Cela a pour effet de garder une efficacité plus grande sur tous les types de boites (fig. 7), contrairement au système sans coopération (courbes en pointillés) qui voit dès le 1er cycle sa population en robots rouges (et donc son efficacité à ramener des boites rouges) chuter. Le système avec apprentissage de la criticité perd davantage de robots que le système avec coopération mais moins que le système sans coopération. Cela vient du fait que pendant la phase d'apprentissage, les agents ont tendance à se tromper sur les récompenses attendues et donc à prendre des boites pour lesquelles ils sont moins efficaces. Cela explique le rapprochement des courbes du niveau d'énergie (fig. 6 et fig. 2). Enfin l'erreur moyenne diminue plus rapidement que dans l'environnement sans perturbation (fig.

8) car lors des ajouts de boites, les robots ne ramassent qu'un type de boite particulier, rendant alors l'apprentissage plus aisé.

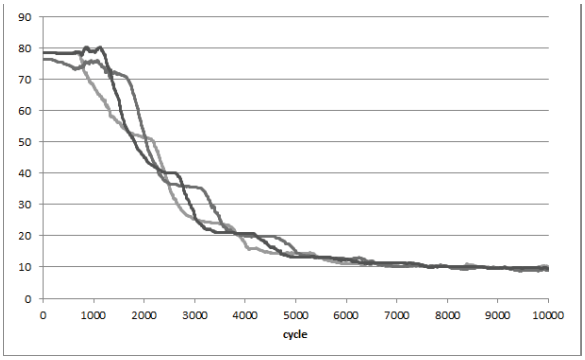

FIGURE 8 – Erreur moyenne sur les récompenses par couleur

6 Conclusion et perspectives

Ce papier introduit une nouvelle métrique, la criticité. Elle permet à un agent de juger localement la pertinence de ses actions en terme de coopération de sorte à aider ses voisins les plus contraints et ainsi garantir l'adéquation fonctionnelle du système (conformément à l'approche par AMAS). Cette criticité peut être difficile voire impossible à définir analytiquement dans le cadre de systèmes complexes. Nous avons donc proposé un algorithme permettant à chaque agent d'apprendre sa propre criticité grâce aux feedbacks retournés par l'environnement suite aux actions qu'il a réalisées. Cet apprentissage a certes un coût mais il permet au système de tendre vers un état de coopération. Les résultats obtenus dans le cadre du système CoCaRo sont encourageants et sont un premier pas vers l'utilisation de cette notion pour la conception de plusieurs AMAS interdépendants mais conçus indépendamment. Si des agents sont capables d'apprendre à coopérer entre eux au travers de l'apprentissage et de l'utilisation de leur criticité respective, alors nous pouvons envisager de permettre à plusieurs AMAS d'apprendre à coopérer sur ce même modèle et ainsi former un AMAS d'AMAS fonctionnellement adéquat.

Références

[1] C. Bernon, V. Camps, M.P. Gleizes, and G. Picard. Engineering Adaptive Multi-Agent Systems : The ADELFE Methodology . In *Agent-Oriented Methodologies* .

[2] T. Bouron. *Structures de communication et d'organisation pour la coopération dans un univers multi-agent*. PhD thesis, Univ. Paris VI, 1992.

[3] S. J. Cammarata, D. McArthur, and R. Steeb. Strategies of cooperation in distributed problem solving. In A. Bundy, editor, *IJCAI*, pages 767–770, 1983.

[4] V. Camps. *Vers une théorie de l'auto-organisation dans les systèmes multi-agents basée sur la coopération : application à la recherche d'information dans un système d'information répartie*. PhD thesis, Univ. Paul Sabatier, Toulouse, 1998.

[5] D. Capera. *Systèmes multi-agents adaptatifs pour la résolution de problèmes : Application à la conception de mécanismes*. PhD thesis, Univ. Paul Sabatier, Toulouse, 2005.

[6] Ph. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42 :213–261, 1990.

[7] Randall D. and Reid S. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20 :63–109, 1983.

[8] A. Drogoul, E. Amouroux, P. Caillou, B. Gaudou, A. Grignard, N. Marilleau, P. Taillandier, M. Vavasseur, D. A. Vo, and J.D. Zucker. GAMA : multi-level and complex environment for agent-based models and simulations. In *AAMAS*, pages 1361–1362, Saint-Paul, MN, USA, 2013.

[9] J. Ferber. *Les systèmes multi-agents : vers une intelligence collective*. InterÉditions, 1995.

[10] J. R. Galliers. A strategic framework for multi-agent cooperative dialogue. In *ECAI*, pages 415–420, 1988.

[11] J.-P. Georgé. *Résolution de problèmes par émergence : étude d'un environnement de programmation émergente*. PhD thesis, Univ. Paul Sabatier, Toulouse, 2004.

[12] P. Glize. L'adaptation des systèmes à fonctionnalité émergente par auto-organisation coopérative. HDR, Univ. Paul Sabatier, Toulouse, 2001.

[13] T. Hogg and B. A. Huberman. Better than the best : The power of cooperation. In *LNCS*, pages 165–184, 1993.

[14] Sen S. Sekaran M. To help or not to help. In *7th Annual Cognitive Sciences Conference*, Pittsburg, July 1995.