Heuristique pour l'apprentissage automatique décentralisé d'interactions dans des systèmes multi-agents réactifs.

Heuristics for automatic decentralized interaction learning in reactive multi-agent systems.

Vincent Thomas [1] Christine Bourjot [1] Vincent Chevrier [1]

[1] équipe MAIA - LORIA

LORIA - Campus Scientifique - BP 239
54506 Vandoeuvre-lès-Nancy Cedex
vthomas@loria.fr

Résumé

Cet article propose une heuristique pour la construction automatique d'interactions dans un système multi-agent réactif. Il décrit le formalisme interac-DEC-POMDP qui permet de représenter dans un cadre homogène interactions et actions puis développe un algorithme fondé sur des échanges de récompenses et des techniques d'apprentissage par renforcement pour construire automatiquement et de manière entièrement décentralisée des organisations dans une sous-classe des interac-DEC-POMDPs. Ces techniques permettent en outre de produire à moindre coûts des comportements collectifs adaptatifs basés sur la notion d'interaction directe.

Mots Clefs

Système multi-agents réactifs, interaction, apprentissage par renforcement, conception de SMA

Abstract

This article proposes an heuristics for the automatic computation of interactions in reactive multi-agent system. It describes first the Interac-DEC-POMDP formalism and then focuses on a algorithm based on reinforcement learning techniques and distribution of individual rewards among agents to build interaction and collective behaviour in a sub-class of DEC-POMDP. These techniques produce adaptative collective behaviour based on the use of direct interactions.

Keywords

Reactive multi-agent system, interaction, reinforcement learning, SMA construction

1 Introduction

Nos travaux ont pour objectif de construire des systèmes multi-agents réactifs : c'est-à-dire des systèmes caractérisés par un contrôle distribué réparti entre des entités régies par des règles simples de type stimulus-réponse. Nous cherchons en outre à construire des systèmes capables de s'adapter de manière autonome pour produire à l'exécution, par l'agencement des comportements individuels, une réponse collective à un problème. L'aspect coopératif de ces systèmes provient du fait que les performances sont évaluées par l'intermédiaire d'une fonction globale et non pas au niveau individuel.
Concevoir un système multi-agents consiste alors [6] :

- à proposer une architecture interne aux agents
- à décrire les formes d'interactions possibles
- à trouver des lois comportementales et des lois d'adaptation des agents
- à implémenter ces éléments.

Il s'agit d'un problème complexe puisque la simplicité des comportements des agents s'oppose à la complexité des problèmes que nous envisageons, que le contrôle du système s'effectue au niveau individuel alors que son évaluation est effectuée au niveau collectif et que nous souhaitons tirer parti de la multiplicité des agents pour faire apparaître des comportements collectifs qualitativement différents des comportements individuels.
Plusieurs démarches peuvent être envisagées pour la construction de tels systèmes. Certaines se fondent sur la conception de méthodologies permettant de "faciliter le processus d'ingénierie des systèmes par un ensemble de méthodes appliquées tout au long du cycle de développement d'un logiciel, ces méthodes étant unifiées par une certaine approche philosophique générale" [5]. De nombreuses méthodologies ont ainsi vu le jour comme GAIA [17], mais la construction des comportements reste à la

charge du concepteur. Une autre démarche possible est de s'inspirer de systèmes collectifs observés dans la nature comme métaphore pour proposer de nouveaux mécanismes [11] et de nouvelles formes d'interaction comme la stigmergie. Ces approches se sont montrées fructueuses (cf [2]) mais nécessitent un travail important de la part du concepteur pour adapter les comportements naturels à un problème donné a priori.
Comme nous souhaitons intégrer le moins possible le concepteur dans la boucle de construction du système, nous nous concentrons sur une approche fondée sur des cadres formels. Elle consiste à proposer un cadre formel opératoire dans lequel il est possible d'exprimer un problème collectif, de représenter des agents réactifs et de manipuler facilement leurs comportements afin d'y élaborer des algorithmes de construction. Du fait de son expressivité, un formalisme propose déjà un certain nombre de réponses quant aux architectures internes des agents et aux formes d'interactions possibles dans le système. Proposer un cadre relègue la conception de systèmes aux algorithmes de production des comportements individuels mais constitue déjà une réponse partielle au problème de conception.
Certains formalismes permettent de représenter des problèmes multi-agents comme les DEC-POMDPs (Decentralized Markov Decision Process), cependant leur résolution reste complexe et nécessite de les reconsidérer. Notre approche consiste à proposer un nouveau formalisme proche des DEC-POMDPs mais intégrant explicitement la notion d'interaction directe.

2 DEC-POMDPs

2.1 Définition

Les DEC-POMDPs constituent un cadre formel permettant de représenter des problèmes de prise de décision séquentielle multi-agents dans l'incertain (cf [1]). Un DEC-POMDP est donné par un tuple $< \alpha, S, A_i, T, \Gamma_i, O, R >$:
- α désigne le nombre d'agents du système
- S désigne l'ensemble fini des états possibles du monde
- A_i désigne l'ensemble des actions possibles pour l'agent $i \in [0, \alpha]$. Une action jointe est définie par le tuple des actions des agents. L'espace d'action jointe A en désigne son ensemble $A = \times A_i$
- $T : S \times A \times S \rightarrow [0, 1]$ est la matrice de transition du système et caractérise les lois d'évolution du monde. $T(s, a, s')$ fournit la probabilité d'arriver dans l'état s' lorsque les agents émettent l'action jointe a en partant de l'état s
- Γ_i définit les observations possibles pour l'agent i
- $O : S \times A \times \Gamma_i^* \rightarrow [0, 1]$ définit la fonction d'observation du système qui associe à un état et une action donnés l'ensemble des observations à fournir aux agents
- $R : S \times A \rightarrow \mathbb{R}$ désigne la fonction de récompense globale. Elle traduit la motivation des agents et permet de caractériser la tâche globale à résoudre.

Le système est caractérisé par une exécution décentralisée et peut être compris comme un système doté d'une dynamique propre influencée par les actions émises par les agents.

2.2 Résoudre un DEC-POMDP

Chaque agent est représenté par une politique individuelle π_i. En toute généralité, cette politique associe à tout historique d'observation-action $passe_n = (o_0, a_0, o_1, a_1 ... o_n)$ l'action effectuée par l'agent : $\pi_i : \Gamma_i^* \times A_i^* \rightarrow A_i$. Résoudre un DEC-POMDP $< \alpha, S, A_i, T, \Gamma_i, O, R >$ donné consiste à trouver un tuple de politiques individuelles $\pi = (\pi_1, ... \pi_n)$ qui maximise à l'exécution la récompense globale reçue par le système. Comme ce problème a été prouvé être un problème NEXP-complet (cf [1]), nous sacrifierons l'optimalité et nous limiterons par la suite à des agents sans mémoire à court terme dont la fonction de décision peut s'exprimer sous la forme : $\pi_i : \Gamma_i \rightarrow A_i$. La politique π_i d'un agent peut alors être considérée comme l'ensemble des règles comportementales stimulus-réponse régissant son comportement. Résoudre un DEC-POMDP consiste donc à construire les lois comportementales des agents pour résoudre un problème collectif caractérisé par une fonction de récompense globale.

2.3 Difficultés des approches décentralisées

Plusieurs approches de résolution peuvent être envisagées. Les approches centralisées utilisent un agent qui est supposé disposer de l'ensemble des informations des agents pour construire les politiques individuelles [3] ou [8].
Cependant, comme nous cherchons à construire des systèmes pouvant s'adapter à l'exécution et comme il n'est pas toujours possible d'avoir une vision globale du système, nous allons nous concentrer sur des approches décentralisées. Elles consistent à mettre à jour localement les politiques de chaque agent en fonction des informations dont il dispose. De telles approches doivent faire face à un certain nombre de difficultés :
- la complexité du problème en lui-même qui rend illusoire la construction automatique des comportements optimaux dans le cas général.
- le problème de co-évolution des agents : chaque agent est libre de modifier ses règles comportementales. Ainsi, les lois d'évolution de l'environnement perçu par un agent peuvent être modifiées par le changement de politique des autres agents, ce qui introduit des problèmes de coordination importants entre les agents.
- le problème du credit assignment consistant à répartir de manière adéquate la récompense globale au sein des agents et le problème de la tragédie des communs qui fait que la maximisation égoïste par chaque agent de ses récompenses locales peut conduire à la minimisation de la performance globale du système [9].

Des approches ont cependant montré :
- qu'il était possible de construire des systèmes multi-agents à partir d'apprentissages par renforcement décentralisés [4], les agents apprennent par expérience à se coordonner pour résoudre une tâche. La coordination des

comportements entre agents se fait grâce à l'utilisation d'un signal de récompense commun qui permet de renforcer certaines actions jointes. De telles solutions présentent de grandes capacités de passage à l'échelle mais nécessitent que le concepteur guide l'apprentissage des agents.
– qu'il était possible de tirer parti de la structure du problème pour construire des solutions approchées. Ces solutions peuvent de plus être implantées de manière distribuée en tenant compte des relations de dépendance exprimées explicitement par la représentation du modèle en MDP factorisé [7]. Néanmoins de telles approches se fondent sur une communication gratuite et sans limite entre les agents du système.

3 Notre proposition

3.1 Objectifs

Nous cherchons à construire de manière entièrement décentralisée les politiques des agents. Nous souhaitons en outre que nos systèmes présentent de fortes contraintes de localité qui leur permettront d'être utilisées dans des applications réelles caractérisées par des communications limitées entre agents (en terme de nombre d'agents et de distance de communication au sens d'une topologie définie dans le cadre du problème), par des observations partielles, par des actions aux conséquences locales (au sens d'une topologie dépendante du problème) et sans aucune étape de centralisation aussi bien à l'exécution et qu'à la construction. Pour ce faire, nous avons proposé un nouveau cadre formel dans lequel la notion d'interaction directe va pouvoir être représentée. [15]

3.2 Intérêt de l'interaction directe

La caractéristique principale d'un système multi-agents réside dans la notion d'interaction : les agents d'un même système s'influencent mutuellement pour construire une solution au problème posé. Ces interactions qui constituent l'essence d'un système multi-agents peuvent se faire de deux manières : de manière indirecte par l'environnement ou de manière directe par communication.
Nous pensons que dans beaucoup de problèmes, les agents ne sont pas constamment en interaction et il n'est pas nécessaire de raisonner constamment au niveau global. Par contre, des décisions impliquant un nombre réduit d'agents sont nécessaires pour obtenir des solutions satisfaisantes. Si l'on souhaite pouvoir construire des prises de décision de manière automatique, il est souhaitable de pouvoir évaluer les conséquences d'une décision et de déterminer les agents localement en interaction. Or l'interaction entre agents n'apparaît pas explicitement dans un DEC-POMDP. En outre, seules les interactions indirectes sont possibles dans ce formalisme mais il est difficile de les considérer puisque les agents en interaction ne sont pas déterminés a priori et dépendent de l'évolution globale du système.
Nous proposons donc d'intégrer la notion d'interaction directe dans le cadre DEC-PODMP. L'interaction directe présente alors plusieurs intérêts :
– Tout d'abord, elle permet de représenter explicitement la présence d'autres agents dans le système et permet aux agents de raisonner sur la multiplicité des prises de décision.
– Ensuite, en considérant des actions jointes, elle permet de résoudre un certain nombre de problèmes de coordination auxquels le concepteur a déjà pensé en définissant les interactions qu'il souhaite mettre en œuvre.
– Elle définit de nouvelles entités constituées par les agents interagissant. Il est alors possible d'effectuer des calculs sur cette entité comme des transferts de récompense entre agents pour répondre à la tragédie des communs et construire des systèmes à partir de récompenses individuelles locales.
– Enfin, elle définit une structure au problème à partir d'un concept facilement interprétable fondamental dans les systèmes multi-agents.

3.3 Caractérisation

Une interaction directe est définie comme une action mutuelle réciproque. Elle implique plusieurs agents et aboutit à l'émission d'une action jointe décidée collectivement par concertation. Cette action jointe modifie l'état global du système dans le voisinage des agents impliqués dans l'interaction.
Une interaction directe est caractérisée par
– un type d'interaction
– un émetteur
– un receveur (potentiellement plusieurs)
– un ensemble d'actions jointes comme conséquences possibles de l'interaction
– une décision prise collectivement à partir de communications locales

Le formalisme interac-DEC-POMDP intègre cette notion d'interaction directe et permet de représenter explicitement dans un même cadre actions et interactions directes entre agents.

3.4 Interac-DEC-POMDP

Un interac-DEC-POMDP est constitué de deux modules : un module d'action et un module d'interaction.

Module d'action. Le module d'action est défini par un DEC-POMDP $< \alpha, S, A_i, T, \Gamma_i, O, R >$. Les comportements des agents sont caractérisés par leurs politiques individuelles $\pi_i : \Gamma_i \times A_i \rightarrow [0, 1]$

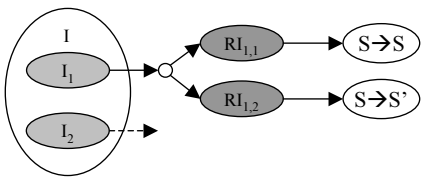

FIG. 1 – Structure des interactions

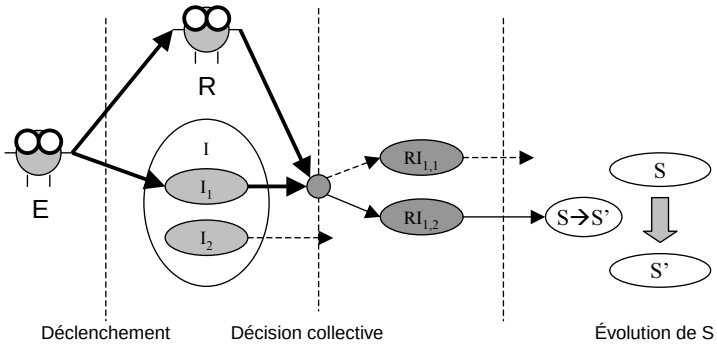

FIG. 2 – Exécution d'une interaction : a) déclenchement, b) décision collective, c) exécution du résultat

Module d'interaction. Le module d'interaction est défini par des interactions structurées de la manière suivante (cf figure 1) :
- un ensemble de types d'interactions $I = \{I_k\}$
- pour chaque type d'interaction I_k, un ensemble de résultats possible $RI_k = \{RI_{k,l}\}$
- pour chaque résultat $RI_{k,l}$, un ensemble de matrice de transition $TR_{k,l} : agent^* \times S \rightarrow S$ qui définit l'état d'arrivée du système après application de l'interaction en fonction de l'état de départ et des agents impliqués dans l'interaction.

L'exécution d'une interaction se fait en plusieurs étapes (cf figure 2) :
- Dans un premier temps, un agent décide de déclencher une interaction vers un ensemble d'agents particuliers en fonction de sa politique individuelle de déclenchement $\pi_{decl,i} : \Gamma_i \rightarrow I \times agent^*$ (cf fig 2 a)
- Ensuite, un résultat est décidé collectivement à partir d'une politique jointe sur l'ensemble des agents impliqués : $\Pi_{agent^*} : \gamma_{agent^*} \rightarrow RI_k$. Cette politique est définie pour chaque ensemble d'agents et chaque interaction et fournit en fonction des observations des agents impliqués le résultat décidé. Nous verrons dans les parties suivantes comment représenter ces politiques de manière distribuée. (cf fig 2 b)
- Enfin, le résultat choisi est exécuté et modifie l'état du système en fonction de $TR_{k,l}$. (cf fig 2 c)

Agencement. Plusieurs agencements possibles des actions et interactions sont envisageables :
- Il est possible d'avoir des actions et interactions concurrentes, un agent pouvant émettre soit une action soit une interaction
- Il est possible d'avoir des actions et des interactions sérialisées : les agents émettent leur actions, émettent ensuite leurs interactions, et le processus se répète

Nous envisageons pour le moment, le cas le plus simple où actions et interactions sont sérialisées.

Résolution. Résoudre un interac-DEC-POMDP consiste alors à déterminer :

Quoi faire par les politiques d'actions individuelles π_i

Quand comment et avec qui interagir par les politiques de déclenchement individuelles $\pi_{decl,i}$

Comment résoudre l'interaction par les politiques jointes de résolution Π pour toutes les interactions et tous les ensembles d'agents

Résoudre un interac-DEC-POMDP est un problème plus complexe qu'un DEC-POMDP car cela nécessite de calculer plus de politiques et car le formalisme interac-DEC-POMDPs inclut les DEC-POMDPs. Comme résoudre un tel problème est irréalisable pour le moment, nous nous limiterons à la recherche de solutions approchées. Afin de tester notre approche, nous nous sommes concentrés sur une sous-classe des interac-DEC-POMDP qui permet d'isoler la problématique de conception des interactions.

4 Une sous classe de problème

4.1 Description

Cette partie a pour objectif de présenter la sous-classe d'interac-DEC-POMDP sur laquelle nous allons nous pencher. Cette sous-classe est caractérisée par le fait que les interactions directes constituent les seules influences possibles entre agents.

Dans cette sous-classe, les actions entre agents sont indépendantes et le module d'action peut se décomposer en des modules d'action individuels :
- L'espace d'état peut se partitionner en espaces d'états individuels S_i. Chaque agent i est caractérisé par $s_i \in S_i$ et l'état global du système est défini par $S = \times S_i$.
- La matrice de transition T peut se décomposer en matrices individuelles $T_i : S_i \times A_i \times S_i \rightarrow [0, 1]$
- Chaque agent perçoit intégralement son état individuel
- La fonction de récompense globale R se décompose en fonctions locales additives r_i, $R = \sum r_i$

Le module d'interaction revêt lui aussi une forme particulière dans ce cadre. Les matrices de transitions peuvent s'exprimer en fonction des espaces individuels des agents impliqués : $TR_{k,l} : S_i \times S_j \times S_i \times S_j \rightarrow [0, 1]$ pour deux agents i et j. Les interactions constituent alors le seul moyen qu'a un agent i de modifier l'espace individuel de

l'agent j. De plus, du fait des contraintes de localité que nous souhaitons, nous supposerons en outre que les 'états d'interface' à partir desquels un agent peut tenter d'exercer une interaction sur un autre agent sont en faible nombre.

4.2 Problème des pompiers

Le problème sur lequel nous nous focalisons consiste à éteindre des feux répartis dans un environnement. Chaque agent-pompier i évolue dans une pièce caractérisée par un nombre d'états donné. Seuls quelques agents peuvent atteindre les feux à éteindre et seuls quelques agents peuvent accéder à des sources d'eau. Les agents ont en outre des seaux à leur disposition pour transporter de l'eau. Ainsi, on distingue plusieurs types de pompiers :

- les pompiers ravitailleurs qui ont accès à de l'eau et qui peuvent remplir un seau.
- les pompiers extincteurs qui ont accès au feu et peuvent l'éteindre s'ils ont de l'eau
- les pompiers couloir qui sont situés dans des pièces sans feu ni eau, ces pièces disposent de quatre portes menant à des pièces connexes.

L'état individuel d'un agent correspond à sa position dans la pièce et à une variable booléenne 'état de son seau' (rempli ou vide).

Un pompier peut émettre plusieurs actions : il peut se déplacer, il peut remplir son seau s'il est proche d'une case d'eau et vider son seau s'il est proche d'une case de feu. Lorsque cette dernière action a lieu et si le seau est rempli, l'agent qui en est à l'origine (et lui seul) reçoit une récompense individuelle positive ($+100$). Dans les autres cas, aucune récompense n'est distribuée.

Enfin, les interactions entre agents correspondent à des échanges de seaux. Une telle interaction implique deux agents sur des cases connexes dans des pièces séparées. Elle peut avoir deux résultats : soit l'échange est effectif et le seau passe d'un agent à l'autre, soit l'échange est refusé. Cette interaction ne peut avoir lieu que localement et respecte bien nos contraintes de localité tant au niveau de ses conditions d'utilisation que de ses conséquences.

L'objectif va être d'utiliser à bon escient les interactions et les actions pour permettre aux agents de chercher de l'eau, d'échanger les seaux dans la bonne direction et d'éteindre les feux. Nous cherchons donc à construire de manière entièrement décentralisée des organisations constituées d'agents formant une chaîne pour éteindre les feux.

5 Algorithmique

5.1 Objectif

L'objectif consiste à construire à partir d'apprentissages entièrement décentralisés et de récompenses individuelles un comportement collectif intéressant pour le système. Il est à noter que du fait de l'architecture interne des agents (pas de mémoire à court terme) le comportement optimal est en toute généralité inatteignable.

Malgré les caractéristiques du problème des pompiers, la résolution reste non triviale car :

- Tout d'abord, les agents sont guidés par des récompenses individuelles. Ces récompenses individuelles correspondent à l'avancement de la tâche en cours mais sont perçues localement par les agents. Ainsi un agent peut participer à l'avancement de la tâche sans recevoir directement une récompense et ne peut alors pas évaluer correctement son comportement.
- De plus, il y a une interdépendance entre les politiques d'actions et d'interactions des agents à laquelle s'ajoute un manque de connaissance sur l'état des autres ainsi que sur leurs comportements.

5.2 Présentation générale

L'approche originale que nous proposons se fonde sur une heuristique de partages de récompenses et consiste en deux étapes :

- Dans la première, nous montrons qu'il est possible de réduire les états individuels et les actions individuelles à considérer en extrayant les politiques potentiellement optimales.
- Dans la seconde, nous effectuons l'apprentissage collectif pour générer le comportement global à partir des politiques individuelles précédentes et des interactions du système.

Comme l'originalité de notre approche repose sur l'agencement de ces politiques, nous préférons mettre l'accent sur cette seconde partie dans la suite de l'article.

5.3 Réduction au niveau individuel

Le problème des pompiers est proche des MDPs faiblement couplés pour lesquels un problème mono-agent peut se décomposer en sous-problèmes reliés par un faible nombre d'états [10]. La principale différence réside dans le fait que dans les MDPs faiblement couplés les sous-problèmes sont organisés de manière séquentielle, alors que dans le problème des pompiers, ils sont organisés en parallèle, chaque agent évoluant dans son sous-espace **simultanément** aux autres agents.

Néanmoins, la première partie des travaux de [10] peut être utilisée. Elle consiste à extraire pour chaque sous-problème un cache de politiques constitué par l'ensemble des politiques potentiellement optimales reliant les états d'interface aux autres sous-problèmes. Pour le problème des pompiers, cela consiste pour chaque agent à construire les politiques potentiellement optimales pouvant mener aux états d'interface permettant d'interagir avec les autres agents.

Nous supposerons qu'il est possible d'extraire pour chaque pompier un ensemble de politiques potentiellement optimales (cf figure 3) qui sont :

- pour les agents pompiers ravitailleur, aller chercher de l'eau ou ne rien faire.
- pour les agents pompiers extincteur, aller éteindre un feu ou ne rien faire.
- pour les agents couloirs, se déplacer d'une porte vers une autre ou ne rien faire.

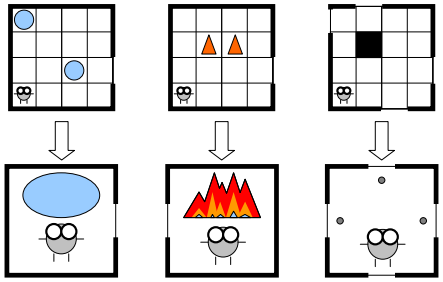

FIG. 3 – Réduction des différents sous-problèmes

Nous pouvons alors simplifier notre problème en ne considérant que ces politiques et les états d'interface au niveau de chaque agent. Par exemple, pour un agent couloir, il est inutile de considérer l'ensemble des états et des déplacements possibles au sein du couloir. Les seules politiques utiles pour le système correspondent aux comportements conduisant à un état permettant d'interagir avec un autre agent, et ce indépendamment de la taille du couloir. Ainsi, bien que nous nous intéressons par la suite uniquement à quelques états et quelques actions par agent, il faut garder à l'esprit que la portée est plus générale du fait des réductions envisageables.

5.4 Agencement collectif des politiques

Dans cette partie, nous cherchons à combiner de manière décentralisée toutes les politiques potentiellement optimales présentées dans la partie précédente. Il s'agit pour les agents d'apprendre simultanément quelle politique utiliser et quelle interaction déclencher.

Techniques employées. Puisque les agents n'ont pas accès aux comportements des autres agents, les techniques de planification sont difficilement envisageables. Nous avons donc opté pour des techniques d'apprentissage par renforcement [16] pour synchroniser l'exécution des politiques. A l'exécution du système, chaque agent effectue un apprentissage décentralisé simultanément avec les autres agents. Un cycle d'apprentissage est constitué d'une phase d'exécution des actions suivie par une phase d'exécution d'interactions et par une phase d'apprentissage.

Représentation. Chaque agent dispose de mémoire à long terme pour pouvoir synthétiser l'ensemble de son expérience et mettre à jour ses comportements individuels. Cette mémoire est constituée de Q-valeurs individuelles. Dans le Q-learning [16], ces Q-valeurs sont des fonctions $Q : S_i \times A_i \rightarrow \mathbb{R}$ qui associent à un état et une action donnés l'espérance de gain de l'agent. Dans notre cas, le sens de ces Q-valeurs est difficile à préciser puisqu'elles intègrent des récompenses d'origines diverses mais elles constituent néanmoins un indicateur permettant de construire les politiques. De manière analogue aux fonctions de valeurs classiques, on notera $v(s) = max_a Q(s,a)$.

5.5 Cycle d'apprentissage

Exécution des actions. Un agent décide de son action en fonction d'une politique déterminée par ses Q-valeurs. Afin de permettre l'exploration de nouvelles pistes, cette politique est stochastique et a tendance à privilégier l'action la plus prometteuse. Pour ce faire, nous avons utilisé des politiques ϵ-greedy caractérisées pour un ϵ donné par la formule classique suivante :

$$\pi_i(s_i) = \begin{cases} \text{aléatoire} \in A_i & \text{avec probabilité de } \epsilon \\ argmax_a(Q(s_i,a)) & \text{avec prob. de } (1-\epsilon) \end{cases}$$

Tous les agents émettent simultanément leur action choisie, le système évolue en fonction de l'action jointe. Chaque agent reçoit une récompense individuelle r_i et se trouve désormais dans l'état s'_i.

Exécution des interactions. Chaque agent à son tour évalue individuellement l'interaction qu'il souhaite déclencher (échange de seau ou non dans notre cas). Les interactions sont déclenchées et résolues séquentiellement.
La résolution d'une interaction consiste à choisir un résultat d'interaction parmi les résultats possibles (échange effectif ou non dans notre cas). Pour simplifier les notations, on utilisera pour un résultat $R_{k,l}$ la notation : $(TR_{k,l}(s'_1), TR_{k,l}(s'_2)) = TR_{k,l}(s'_1, s'_2)$. $TR_{k,l}(s'_1)$ désigne l'état de l'agent 1 après exécution du résultat $R_{k,l}$.
L'heuristique que nous proposons consiste à choisir le résultat d'interaction qui maximise (avec un facteur d'exploration ϵ) la somme des Q-valeurs individuelles après exécution de l'interaction selon la formule [1] :

$$R_c = argmax_{R_{k,l}}(V_1(TR_{k,l}(s'_1)) + V_2(TR_{k,l}(s'_2)))$$

En faisant cela, certains agents vont éventuellement renoncer à une partie de leur récompense pour permettre à d'autres agents d'augmenter leurs récompenses futures. On peut ainsi obtenir des comportements altruistes utiles pour le système qu'il n'est pas possible d'obtenir avec des apprentissages purement égoïstes. Cette formule suppose en outre que les conséquences des interactions sont connues par les agents : il est possible de s'en affranchir avec un second apprentissage [15] mais nous n'aborderons pas cet aspect dans cet article.
Afin de prendre en compte l'interaction dans les comportements individuels et d'inciter les agents à reproduire l'interaction si elle est bénéfique, le gain collectif obtenu par exécution de l'interaction est réparti entre les agents sous forme d'un transfert de récompense sociale. Ce gain collectif peut s'exprimer par la différence entre la somme des fonctions de valeurs des agents impliqués avant et après interaction. La seconde heuristique que nous proposons consiste à répartir ce gain de manière équitable pour inciter au mieux les agents à reproduire leurs actions.

[1] Ce résultat peut être évalué localement par les agents en se communiquant quelques Q-valeurs d'état individuel

$$gain_1 = V_1(TR_c(s'_1)) - V_1(s'_1)$$
$$gain_2 = V_2(TR_c(s'_2)) - V_2(s'_2)$$
$$r_{s,1} = -gain_1 + 0.5(gain_1 + gain_2)$$
$$r_{s,2} = -gain_2 + 0.5(gain_1 + gain_2) = -r_{s,1}$$

Apprentissage des actions. A l'issue de l'ensemble des interactions, chaque agent i met à jour sa Q-valeur de départ avant action $Qval(s_i, a_i)$ en fonction
– des récompenses reçues au cours de la phase d'action r_i
– des récompenses reçues au cours des interactions $r_{s,i}$
– de la Q-valeur de l'état d'arrivée s''_i aprés interactions
La formule de mise à jour des Q-valeurs est analogue à celle du Q-learning [16] ($\gamma \in [0, 1]$ est le discount factor et α le coefficient d'apprentissage)

$$Qval(s_i, a_i) = (1-\alpha)Qval(s_i, a_i)+\alpha(r_i+r_{s,i}+\gamma V(s''_i))$$

Stratégie d'apprentissage. Afin d'éviter que les agents ne modifient trop vite leurs politiques et que celles ci n'oscillent, le coefficient α est faible et le coefficient ϵ tend vers 0. Ceci permet de synchroniser les politiques des agents.

5.6 Bilan

Pour faire un bilan succinct, l'approche que nous proposons réside dans plusieurs propositions jointes :
– la première s'inspire des travaux de Parr et stipule qu'il est possible de construire séparément les comportements individuels de leur agencement collectif
– la seconde consiste à synchroniser les agents à partir d'une mémoire à long terme et de techniques d'apprentissage par renforcement [16]
– la dernière réside dans des heuristiques qui résolvent les interactions en fonction des Q-valeurs individuelles et qui permettent de répartir les récompenses dans le système au cours des interactions.

6 Résultats

Afin de mettre en valeur cette approche, plusieurs expériences ont été faites. Une expérience consiste à disposer plusieurs briques d'agents (couloir, feu, eau) en interaction et à effectuer un apprentissage dans ce système.

6.1 Résultats bruts

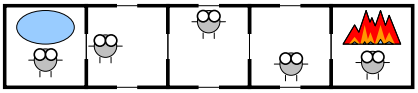

FIG. 4 – Exemple 1

Sur un exemple constitué de 5 agents dans un couloir (cf fig 4), nous avons effectué un apprentissage de 300000 pas de temps avec un facteur $\alpha = 0.02$ et ϵ convergeant de 1 à 0 pendant les 200000 premiers pas de temps.

Au départ, seul l'agent extincteur accède directement à des récompenses en éteignant le feu. Au cours des interactions qui ont lieu, on observe des transferts de récompenses qui permettent de distribuer la tâche au sein des agents. Chaque agent apprend
– à effectuer des échanges de seaux uniquement de la gauche vers la droite du fait de la maximisation de la somme des Q-valeurs
– à amener le seau rempli à l'agent qui se trouve à sa droite du fait des transferts de récompenses sociales
– à aller chercher le seau vers l'agent qui se trouve sur sa gauche qui est incité à en amener un du fait des transferts de récompenses sociales.
Ces apprentissages ne doivent pas être considérés séparément les uns des autres mais sont la conséquence des apprentissages couplés des actions et des interactions.
Cette approche permet de générer automatiquement une organisation (chaînage de déplacements et de transferts de seaux entre les agents) utile à la tâche globale à résoudre de manière entièrement décentralisée et sans que les agents n'aient de vue globale du système. Cette organisation permet d'éteindre le feu (comme l'illustre les récompenses reçues par le système au cours de son apprentissage cf figure 5) et le comportement collectif s'avère dans ce cas simple être optimal. En outre, les fonctions de Q-valeurs se stabilisent au cours du temps (cf figure 6) et même si elles évoluent encore à cause d'une re-répartition des récompenses dans le système, la politique reste inchangée. Les changements de vitesse d'évolution au pas de temps 200000 (cf fig 6) proviennent du fait qu'ϵ a alors pour valeur 0.

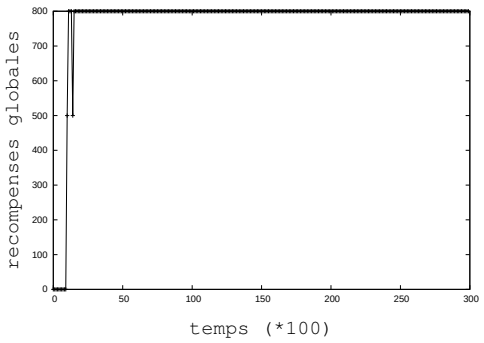

FIG. 5 – Récompenses reçues par exploitation des politiques pendant 20 pas de temps ($\epsilon = 0$) au cours de l'apprentissage en fonction de t

6.2 Résultats Comparés

Nous avons souhaité comparer ces résultats à ceux qu'il serait possible d'obtenir sur un problème analogue modélisé par un DEC-POMDP. Afin de pouvoir guider les agents, une contrainte de localité est levée : les agents peuvent percevoir la récompense globale du système, et les interactions sont remplacées par des zones communes dans lesquelles les agents peuvent entreposer et prendre des seaux (comme l'illustre la figure 7).

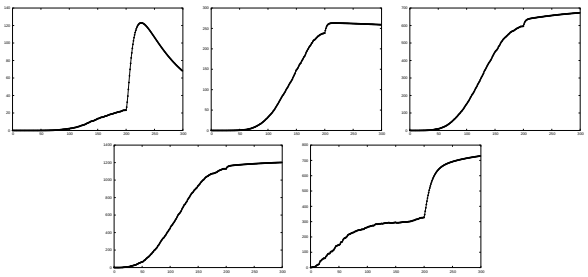

FIG. 6 – Évolution de la somme des Q-valeurs pour chaque agent durant l'apprentissage

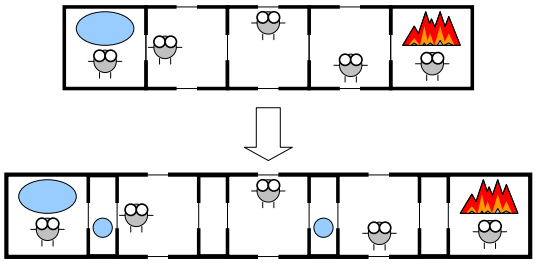

FIG. 7 – DEC-POMDP modélisant le problème de l'exemple 1

Du fait du problème du credit assignment, les comportements des agents oscillent et aucun comportement global n'émerge (cf fig 8 et 9). Les interactions directes permettent donc de guider l'apprentissage des comportements collectifs en explicitant les relations entre les agents et en effectuant automatiquement des transferts de récompense adéquat.

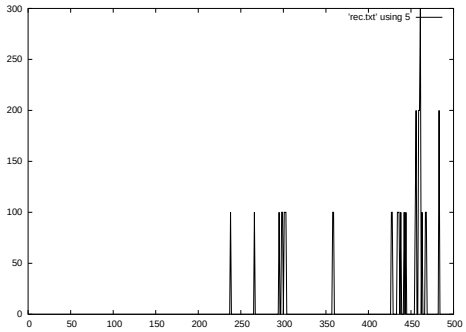

FIG. 8 – Récompenses reçues sans interaction par exploitation au cours de l'apprentissage

6.3 Augmentation du nombre d'agents

L'approche proposée du fait d'apprentissages décentralisés permet d'appréhender des problèmes avec un nombre d'agents élevé : par exemple, des systèmes avec 25 agents sont facilement concevables (cf figure 10 a)). Chaque politique individuelle est construite à partir de Q-valeur in-

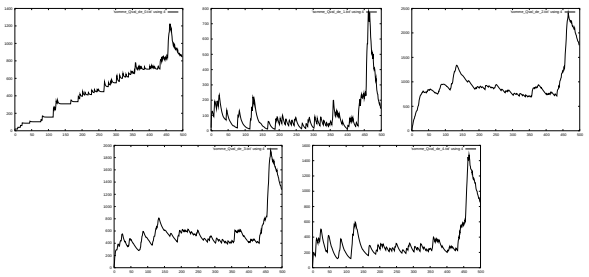

FIG. 9 – Somme de Q-valeurs sans interaction

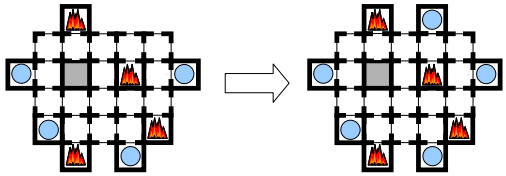

FIG. 10 – Exemple 2

dividuelles : il n'y a donc pas d'explosion combinatoire du nombre d'états ou d'actions à considérer. De plus la complexité de résolution globale est linéaire par rapport au nombre d'agents du système. Les apprentissages effectués permettent aux agents d'éteindre de manière optimale trois feux sur les quatre et fournit de bons résultats (cf figure 11).

Dans cet exemple, les résultats qualitatifs font apparaître la présence de plusieurs chaînes indépendantes. Les apprentissages permettent de mettre en interaction plusieurs agents uniquement en fonction de la tâche à résoudre et de générer simultanément plusieurs organisations si elles sont utiles.

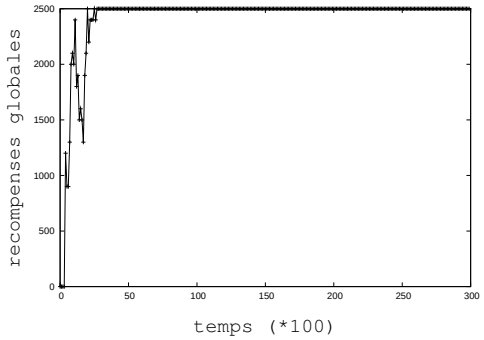

FIG. 11 – Récompense globales reçues pour l'exemple 2

6.4 Systèmes ouverts

Il est possible de maintenir un facteur d'exploration dans le système. Ceci se traduit par une diminution des performances globales (cf fig 12 pour l'exemple 2 avec un maintien de ϵ à 0.1 (qui constitue une valeur très importante)). Cependant, ceci permet d'ajouter à l'exécution de

nouveaux agents au pas de temps 250000 (cf figure 10 b et 12) et de faire converger vers un nouvel équilibre en temps réel. Il est donc possible d'observer des re-configurations automatiques du système avec un apprentissage constant et d'envisager des utilisations de ce processus dans des systèmes ouverts.

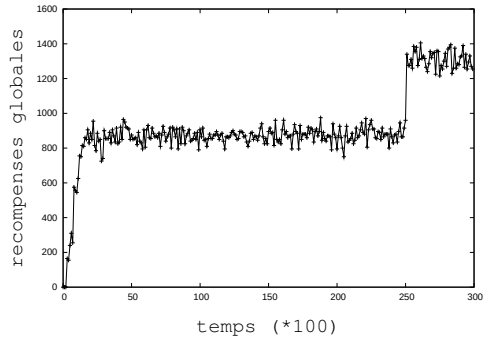

FIG. 12 – Récompense globale reçues avec exploration et re-adaptation

6.5 Limitation du nombre d'interactions

Enfin, d'autres expériences ont montré que l'apprentissage des politiques de déclenchements peut être effectué. Le déclenchement des interactions est alors limité aux interactions utiles (cf [15]) et permet de réduire les communications dans le système.

7 Discussion

7.1 Avantages de l'approche

Cette approche présente un certain nombre de points intéressants quant au formalisme :

– Nous avons présenté un cadre formel qui permet de représenter dans des formalismes markoviens l'interaction à la base des SMA et induit une structuration du système.
– Ce formalisme est opératoire. Il permet de définir des interactions, leurs utilisations, les actions des agents et leurs comportements à l'aide de fonctions facilement manipulables.

L'algorithmique que nous avons développé présente aussi un certain nombre de mérites :

– Elle est très simple et fondée sur des heuristiques entièrement décentralisées, ces heuristiques permettent d'envisager l'utilisation de notre approche dans des systèmes réels dotés de fortes contraintes de localité.
– Elle donne des résultats inaccessibles avec une autre algorithmique aussi naïve dans un cadre DEC-POMDP et prouve donc l'intérêt de représenter explicitement la notion d'interaction dans le système
– Elle présente de bonnes capacités de passage à l'échelle et présente des intérêts pour la construction de systèmes ouverts. Du fait d'apprentissages décentralisés, la complexité des calculs reste linéaire par rapport au nombre d'agents.
– Elle est guidée par un objectif et permet de construire automatiquement des comportements collectifs sans qu'il ne soit nécessaire de donner des lois comportementales a priori

7.2 Positionnement

Les travaux présentés dans cet article peuvent être mis en relation avec d'autres travaux connexes :

– Dans [14], Simonin présente le modèle satisfaction-altruisme qui se fonde sur une approche similaire : les agents évoluent dans l'environnement et peuvent émettre des interactions directes locales pour résoudre les situations de conflits. Notre approche utilise les même principes mais permet de construire automatiquement ces interactions par rapport à un objectif alors que les politiques d'action et d'interaction sont données a priori dans [14] mais les propositions de Simonin présentent en contrepartie des capacités d'adaptation plus importantes pour la tâche.
– Dans [13], Schneider présente des techniques fondées sur des heuristiques permettant à un groupe d'agents de se coordonner par apprentissage décentralisé à partir de récompenses locales. Ces heuristiques consistent pour un agent à maximiser sa fonction de valeur tout en intégrant les fonctions de valeur des agents voisins. Cependant, cette approche nécessite des communications constantes entre les agents et néglige un principe fondamental de l'interaction : la capacité de restructurer le réseau de relations entre les agents. Dans le problème des pompiers, l'ajout d'agents en temps réel ne pose pas de difficulté et les Interac-DEC-POMDP de manière générale permettent de représenter des interactions dépendantes des positions des agents et pouvant évoluer.
– [4] présente des techniques d'apprentissage pour synchroniser des comportements d'agents mais utilise une récompense globale alors que nous cherchons à répartir des récompenses locales dans un système.

7.3 Limites et perspectives

Ces travaux se heurtent néanmoins à un certain nombre de difficultés que nous souhaitons résoudre par la suite :

– Nous avons effectué une simplification en considérant qu'il était possible de réduire facilement les problèmes individuels. En effet, alors que les travaux de Parr s'intéressent à des sous-problèmes résolus séquentiellement, l'aspect simultanée des actions des agents rend le problème plus complexe : des problèmes de synchronisation peuvent apparaître puisque les politiques n'ont pas forcément des durées d'exécution égales. Résoudre ces problèmes nécessiterait de se placer dans un cadre plus complexe dans lequel les actions effectuées par un agent peut avoir des durées variables comme les SMDP (semi markov decision process) [12].
– L'introduction de récompenses négatives dans le système peut bloquer l'apparition d'organisation. En effet, les gains collectifs obtenus lors des interactions sont répartis équitablement entre les agents mais d'autres heu-

ristiques sont envisageables. En effet, s'il coûte beaucoup à un agent couloir particulier de traverser la pièce mais que l'apparition d'une chaîne reste globalement profitable, l'agent proche du feu peut avoir intérêt à donner une grande partie de sa récompense future à son voisin qui pourra motiver à juste mesure les agents à effectuer des actions intéressantes pour le système. Il nous semble ainsi intéressant de se concentrer par la suite sur l'apprentissage des récompenses à transmettre pour opérer à l'exécution une répartition adaptée des récompenses en fonction des besoins des autres agents.

8 Conclusion

Dans cet article, nous avons proposé :

– un cadre formel interac-DEC-POMDP permettant de représenter des actions et des interactions d'un système multi-agent dans un cadre homogène
– une classe de problème particulier dans lequel les seules influences possibles entre agents sont constituées par les interactions
– un algorithme basé sur des techniques d'apprentissage par renforcement et des heuristiques d'échanges de récompenses pour construire de manière entièrement décentralisée les politiques d'actions et d'interactions pour résoudre un problème.

Ces techniques permettent de produire à l'exécution, de manière automatique et décentralisée, une organisation dans le système utile à la tâche en cours. Ces techniques présentent en outre de nombreuses propriétés adaptatives utiles dans des systèmes ouverts et respectent des contraintes de localité concernant les communications, les observations des agents, les conséquences des actions et des interactions qui les rendent envisageables dans des approches concrètes.

A plus long terme, nous pensons qu'une telle approche consistant à représenter de manière explicite les interactions dans un système ouvre de nouvelles opportunités concernant :

– le calcul des politiques d'interactions
– la génération automatique d'interactions (détection d'actions jointes utiles ou définir des méta-actions consistant à forcer un autre agent à faire une action)
– les techniques à mettre en oeuvre pour réguler le niveau de centralisation (apprentissage hiérarchique par exemple)

Références

[1] D. S. BERNSTEIN, R. GIVAN, N. IMMERMAN et S. ZILBERSTEIN : The complexity of decentralized control of markov decision processes. *Math. Oper. Res.*, 27(4):819–840, 2002.

[2] E. BONABEAU et G. THERAULAZ : *Swarm intelligence*. Oxford university press, 1999.

[3] C. BOUTILIER : Sequential optimality and coordination in multiagent systems. *In IJCAI '99 : Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 478–485, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[4] O. BUFFET : *Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs*. Thèse de doctorat, Université Nancy I, 2003.

[5] J. C. CASTERAN, M. P. GLEIZES et P. GLIZE : Des méthodologies orientées multi-agent. *8ièmes Journées Francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, pages 191–207, 2000.

[6] J. FERBER : Les systèmes multi-agents : un apercu general. *Technique et science informatique*, 16:979–1012, 1997.

[7] C. GUESTRIN, M. LAGOUDAKIS et Ronald PARR : Coordinated reinforcement learning. *Nineteenth International Conference on Machine Learning(ICML 2002)*, pages 227 – 234, 2002.

[8] E. A. HANSEN, D. S. BERNSTEIN et S. ZILBERSTEIN : Dynamic programming for partially observable stochastic games. *In AAAI*, pages 709–715, 2004.

[9] G. HARDIN : the tragedy of the commons. *Science*, pages 1243–1248, 1968.

[10] R. PARR : Flexible decomposition algorithms for weakly coupled Markov decision problems. *Fourteenth Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 422–430, 1998.

[11] S. PICAULT : *Modèles de comportements sociaux pour les collectivités de robots et d'agents*. Thèse de doctorat, Université Paris 6, 2001.

[12] M. L. PUTERMAN : *Markov Decision Processes : Discrete Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[13] J. SCHNEIDER, W.K. WONG, A. MOORE et M. RIEDMILLER : Distributed value functions. *In Proceedings of the 16th International Conference on Machine Learning*, pages 371–378. Morgan Kaufmann, San Francisco, CA, 1999.

[14] O. SIMONIN : *Le modèle satisfaction-altruisme*. Thèse de doctorat, Université Montpellier II, 2001.

[15] V. THOMAS : Interac-dec-mdp : un premier formalisme pour l'utilisation d'interactions directes dans un mdp décentralisé. *Quatrième Journées Nationales sur Processus Décisionnel de Markov et Intelligence Artificielle*, 2004.

[16] C. WATKINS et P. DAYAN : Technical note q-learning. *Machine Learning*, 8:279–292, 1992.

[17] M. WOOLDRIDGE, N. R. JENNINGS et D. KINNY : The gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.