Construction de systèmes multi-agents par apprentissage collectif à base d'interactions

Vincent Thomas, Christine Bourjot, Vincent Chevrier

Construction de systèmes multi-agents par apprentissage collectif à base d'interactions

Vincent Thomas, Christine Bourjot, Vincent Chevrier *

* *Équipe MAIA - tranche C*
Laboratoire LORIA
Campus scientifique - P. 239
54506 Vandoeuvre-les-Nancy, France.
vincent.thomas@loria.fr, christine.bourjot@loria.fr, vincent.chevrier@loria.fr

RÉSUMÉ. *Cet article se focalise sur des approches formelles pour la construction de systèmes multi-agents. Ce travail a cherché à proposer des apprentissages décentralisés pour construire les comportements d'agents sociaux. Cet article propose un formalisme original, l'interac-DEC-POMDP inspiré des modèles markoviens au sein duquel les agents peuvent interagir directement et localement entre eux. A partir de ce formalisme, cet article propose aussi un algorithme d'apprentissage décentralisé fondé sur une répartition heuristique des gains des agents au cours des interactions. Une démarche expérimentale a validé sa capacité à produire automatiquement des comportements collectifs. Les techniques présentées pourraient alors constituer des moyens permettant aux agents de décider automatiquement et de manière décentralisée comment s'organiser avec les autres pour résoudre un problème donné.*

ABSTRACT. *This article deals with formal approaches to build multi-agent systems. The goal of the conducted works was to propose decentralized learning techniques to build the bejavior of social agents. This article presents an original formalism, the interac-DECPOMDP, in which agents can directly interact. On the basis of this formalism, this article proposes a decentralized learning algorithm based on a heuristic distribution of rewards during interactions. Experiments have validated its ability to automatically build collective behaviors. The presented techniques could then constitute a mean to operationalize self-organization in order to solve problems.*

MOTS-CLÉS : *système multi-agents, apprentissage collectif, interaction, modele markovien*

KEYWORDS: *Multi-agent systems, collective learning, interaction, markov models*

1. Contexte scientifique

Cette première partie présente l'objectif à long terme de nos travaux. Elle mettra en évidence les contraintes sur les systèmes que nous considérons et sur les moyens permettant de construire ces systèmes. A l'issue de cette partie, une fois notre objectif décrit, nous présenterons la structure de cet article.

1.1. *Conception de systèmes multi-agents*

L'objectif des travaux présentés dans cet article consiste à construire des systèmes multi-agents (SMA) réactifs, c'est-à-dire, des systèmes caractérisés par un contrôle distribué entre plusieurs entités appelées agents réactifs et régies par des règles simples de type stimulus-réponse.

Dans ce cadre, nous souhaitons construire des systèmes **coopératifs** dont la performance est évaluée à partir d'une fonction d'utilité globale définie a priori et caractéristique du problème à résoudre. Il s'agit alors aux agents d'exhiber des comportements individuels tels que l'agencement de ces comportements à l'exécution parvienne à construire un comportement collectif apte à optimiser cette fonction d'utilité globale.

De plus, nos travaux se concentrent sur des systèmes multi-agents dotés de contraintes fortes afin de pouvoir les utiliser à plus long terme dans des contextes applicatifs réels. Tout d'abord, les agents ne disposent que de **perceptions partielles** : ils ne peuvent accéder directement à la fonction d'utilité globale et n'ont qu'une perception locale de l'avancement de la tâche à résoudre. Ils ne peuvent pas non plus accéder aux comportements des autres agents du système. Les agents doivent alors **s'adapter au cours de l'exécution** du système pour prendre en compte de nouvelles perceptions possibles et les comportements évolutifs des autres agents.

Concevoir un système multi-agent consiste selon (Ferber, 1997) à déterminer

– Quelle architecture (senseurs, effecteurs, mémoire, ...) donner aux agents ?

– Quels comportements réactifs (fonction de décision) et quelles lois d'adaptation donner aux agents ?

– Quelles interactions introduire dans le système ?

– Comment implanter ces systèmes ?

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.

1.2. *Démarches de conception*

On distingue habituellement deux grands types d'approche de conception : les approches bottom-up et les approches top-down.

Les approches **bottom-up** consistent à partir de règles comportementales données a priori, à les transposer dans un cadre applicatif et à réutiliser les comportements émergents pour résoudre le problème posé. De nombreux travaux se sont ainsi inspirés de systèmes naturels comme (Dorigo *et al.*, 1999) qui s'inspire des phénomènes de fourragement dans les colonies de fourmis pour la résolution de problème d'optimisation combinatoire, (Bourjot *et al.*, 2003) qui s'inspire des phénomènes de construction collective de toile pour l'extraction de régions dans des images ou encore (Lumer *et al.*, 1994) qui s'inspire de phénomènes de tri de couvains pour proposer de nouvelles techniques de classification. Néanmoins, ces approches nécessitent souvent un travail de la part du concepteur qui doit analyser les performances du système, adapter les comportements individuels et régler les paramètres régulant ces comportements.

Les approches **Top-down** consistent au contraire à partir du problème et à en déduire les comportements individuels des agents pour que le système collectif parviennent à résoudre au mieux le problème posé. Parmi ces approches, on peut citer les méthodologies qui permettent 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" (Casteran *et al.*, 2000). De nombreuses méthodologies ont ainsi vu le jour comme GAIA (Wooldridge *et al.*, 2000), mais la construction des comportements reste à la charge du concepteur. D'autres approches cherchent à utiliser les techniques d'optimisation pour produire les comportements des agents. Certaines approches utilisent ainsi les algorithmes génétiques comme (Flacher *et al.*, 2006) pour produire des comportements collectifs dans des espaces continus, ou encore des techniques de programmation dynamique (Szer, 2004) ou de descente de gradient (Baxter *et al.*, 2001) dans des cas discrets. L'utilisation de techniques d'optimisation nécessite cependant de disposer d'un formalisme dans lequel les comportements des agents pourront être décrits à l'aide de paramètres d'optimisation.

1.3. *Positionnement*

Comme nous souhaitons intégrer le moins possible le concepteur dans la boucle de construction du système, nous nous concentrons sur une approche top-down fondée sur des cadres formels. L'objectif de nos travaux consiste donc à proposer un cadre 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 automatique. Un fois qu'un tel cadre sera posé, nous chercherons à proposer des techniques d'apprentissage entièrement décentralisées permettant aux agents d'adapter automatiquement leur comportement à la tâche collective à résoudre sans qu'ils ne disposent de vision globale du système.

Le formalisme DEC-POMDPs (Decentralized Markov Decision Process) permet déjà de représenter des problèmes de prise de décision multi-agents. Ce cadre est en outre une extension des modèles markoviens pour lesquels des techniques d'apprentissage par renforcement existent et permettent à un agent de mettre à jour son comportement pour résoudre une tâche. Ce cadre semble ainsi constituer un point de départ intéressant pour proposer des techniques de construction automatique de SMA. Cet article propose le formalisme Interac-DEC-POMDP, un nouveau formalisme inspiré des DEC-POMDP et des techniques d'apprentissage décentralisé en tirant parti.

1.4. *Plan de l'article*

Cet article se décomposera en 5 parties :

– Dans la première partie, nous présenterons les formalismes DEC-POMDPs qui ont l'intérêt de pouvoir représenter des problèmes de décision multi-agents. Nous montrerons cependant qu'il s'agit d'une extension des modèles markoviens initialement prévus pour représenter des problèmes de prise de décision mono-agent et qu'ils ne sont pas adaptés aux systèmes que l'on souhaite construire.

– Dans un second temps, nous présenterons un nouveau cadre formel original inspiré des DEC-POMDP, plus apte à représenter des systèmes multi-agents en intégrant explicitement le concept d'interaction manipulable directement par les agents.

– Dans la troisième partie, nous présenterons des techniques d'apprentissage décentralisé tirant parti de ce nouveau formalisme et permettant aux agents d'apprendre automatiquement à agir dans leur environnement mais aussi à interagir avec les autres agents pour résoudre un problème.

– La quatrième partie s'attardera sur un problème particulier et montrera que ces techniques d'apprentissage parviennent à résoudre à moindre coût des problèmes collectifs particulièrement difficiles à résoudre avec des DEC-PODMPs. Nous montrerons ainsi que ces approches ouvrent de nouvelles pistes dans le cadre de la construction automatique de systèmes multi-agents.

– Dans la cinquième partie, nous ferons un bilan de notre proposition et établirons des liens entre notre approche et des méthodologies existantes. Ces liens permettront de mettre en évidence l'intérêt de notre proposition et ses retombées potentielles.

2. Formalisme DEC-POMDP

Cette partie va se concentrer sur le cadre formel DEC-POMDP (DECentralized Partially Observable Markov Decision Process). Ce cadre formel proposé récemment (cf (Bernstein *et al.*, 2002)) permet de représenter des problèmes de prise de décision séquentielle multi-agents dans l'incertain. Nous présenterons le cadre formel puis les approches permettant d'en tirer parti pour produire des comportements. Nous montrerons ainsi que, bien que le formalisme soit bien établi, son utilisation pour construire des systèmes multi-agents se heurte à une complexité inhérente au formalisme lui-

même. Cette complexité implique que d'autres approches doivent être envisagées pour pouvoir utiliser ce type de formalisme dans un cadre multi-agents.

2.1. *Définition*

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

– $O : S \times A \times \{\Gamma_i\}_i \rightarrow [0, 1]$ définit la fonction d'observation du système qui associe à un état et une action donnés l'ensemble des observations des agents

– $R : S \times A \rightarrow \mathbb{R}$ désigne la fonction de récompense globale immédiate. Elle représente la motivation des agents et permet de caractériser la tâche globale à résoudre.

2.2. *Résolution d'un DEC-POMDP*

Le comportement de chaque agent est représenté par une politique individuelle π_i. En toute généralité, cette politique associe à tout historique d'observation-action $passe_n = (\Gamma_0, a_0, \Gamma_1, a_1 ... \Gamma_n)$ l'action effectuée par l'agent : $\pi_i : passe_n \rightarrow A_i$.

Un système représenté par un DEC-POMDP est caractérisé par une exécution décentralisée puisque chaque agent est doté de sa propre fonction de décision. Un tel système peut être compris comme un système doté d'une dynamique propre influencée par les actions émises par les agents.

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 un critère de performance défini à partir des récompenses immédiates globales reçue par le système. Nous nous concentrerons par la suite sur le critère gamma-pondéré ($\gamma \in]0, 1[$) qui est le plus simple à manipuler : $\sum_i \gamma^i r_i$ (cf (Sutton *et al.*, 1998))

Résoudre un DEC-POMDP consiste donc à construire les lois comportementales des agents pour résoudre un problème collectif caractérisé par une fonction globale à optimiser définie à partir des récompenses immédiates.

2.3. *Approches de résolution*

On peut distinguer deux grandes approches de résolution : les approches de résolution centralisées et les approches décentralisées. Chacune de ces approches possède ses propres limites que nous allons détailler dans les parties suivantes.

2.3.1. *Approches centralisées*

2.3.1.1. Description

Les approches de résolution centralisées se basent sur l'existence d'une entité chargée du contrôle de l'ensemble des agents. La politique jointe est calculée par cette entité qui redistribue ensuite les politiques individuelles aux agents. A l'exécution, chaque agent dispose alors de sa propre politique lui permettant de décider de manière autonome de l'action à effectuer.

2.3.1.2. Difficultés associées

Bernstein et al. se sont intéressés à la complexité de la résolution d'un DEC-POMDP constitué de deux agents dans (Bernstein *et al.*, 2002). Ils ont prouvé que le problème consistant à déterminer "si, étant donné un DEC-POMDP comprenant deux agents $< S, A_1, A_2, P, R, \Gamma_1, \Gamma_2, O, T, K >$, un horizon fini T et un réel K, il existe une politique jointe δ pour laquelle la performance gamme pondérée globale en s_0 $V_\delta^T(s_0) > K$?" est de complexité N-EXP.

2.3.1.3. Approches existantes

Du fait de cette complexité (déjà présente pour des systèmes constitués de deux agents), il est difficilement envisageable de chercher la politique jointe optimale (qui ne reste atteignable que pour des problèmes très simples). La problématique générale de construction d'un système multi-agents doit alors être reconsidérée sous un nouvel angle. Plusieurs approches peuvent être envisagées. Parmi celles-ci, on peut citer :

– celles consistant à se limiter à la **recherche de solutions sous-optimales** dans un temps raisonnable (en essayant de disposer de bornes concernant la qualité de la solution). Cela consiste à mettre en oeuvre des algorithmes permettant de trouver des politiques jointes sous-optimales en tirant éventuellement parti des caractéristiques du problème comme les approches proposées par Guestrin (Guestrin *et al.*, 2001)

– celles consistant à **caractériser des sous-classes de problèmes** pour lesquelles il est possible de trouver la solution optimale en exploitant les propriétés de ces cas particuliers (comme les *transition-independant DEC-MDP* (Becker *et al.*, 2004) ou les MMDPs pour lesquels chaque agent perçoit l'état global du système (Boutilier, 1999))

– celles consistant à **introduire des mécanismes ou de nouvelles capacités** aux agents pour simplifier la résolution. L'introduction de communication permet l'échange d'informations entre agents et peut réduire dans certains cas la complexité du problème de construction des comportements (Pynadath *et al.*, 2002).

2.3.2. *Approches décentralisées*

2.3.2.1. Description

Dans les approches décentralisées, chaque agent met à jour localement sa politique en fonction des informations dont il peut disposer. Ces approches sont particulièrement proches de nos considérations puisque nous souhaitons des agents capables de mettre à jour leur comportement sans aide extérieure.

2.3.2.2. Difficultés associées

La question fondamentale qui se pose dans le cadre d'approches décentralisées consiste à trouver des moyens permettant aux agents de mettre à jour localement leurs politiques tout en prenant en compte la présence des autres ou en d'autres termes, à déterminer comment intégrer une composante sociale à l'agent. 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 : comme chaque agent est libre de modifier ses règles comportementales, 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 entre les 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 (Hardin, 1968).

2.3.2.3. Approches existantes

Parmi les approches existantes on peut citer :

– **l'utilisation de réseaux bayésiens** (Guestrin, 2003) pour tirer parti de la structure du problème inhérente aux SMAs et construire des réponses à moindre coût et de manière décentralisée à partir de techniques d'élimination de variables.

– **les techniques fondées sur l'empathie** (Chades, 2002) qui cherchent à construire par planification itérative les politiques des agents. Bien que ces techniques permettent de construire des comportements à moindre coût, elles se heurtent à des équilibres de Nash dus à la présence d'interactions entre les agents. Ces équilibres nécessitent de modifier simultanément les comportements de plusieurs agents.

– **les fonctions de valeur distribuées** (Schneider *et al.*, 1999) qui permettent aux agents d'échanger leurs récompenses au cours de leur apprentissage et d'intégrer les récompenses des agents voisins pour développer un comportement social prenant en compte les satisfactions des autres. Bien que cette approche utilise des communications constantes entre agents, elle met en évidence des moyens de construire des comportements collectifs à moindre coût en considérant les satisfactions des autres agents.

– **certains MDP faiblement couplés** (Meuleau *et al.*, 1998) qui formalisent un problème d'allocation de ressource comme un problème multi-agent. L'intérêt de cette approche réside dans la représentation explicite des interactions entre agents (consistant en des attributions de ressources communes) et l'utilisation d'heuristiques fondées sur des variables individuelles pour résoudre ces situations d'interaction.

– **Les techniques d'apprentissage incrémental** (Buffet, 2003) qui cherchent à produire des comportements collectifs à partir d'apprentissages décentralisés en présentant aux agents des situations progressivement de plus en plus complexes.

2.3.3. *Bilan concernant le formalisme DEC-POMDP*

Face à la complexité inhérente au problème, nous avons montré que plusieurs types d'approches sont envisageables :

– soit en mettant en évidence des **sous-cadres des DEC-POMDPs** pour lequel il est envisageable de chercher et d'appliquer des techniques de résolution optimales (comme le MMDP (Boutilier, 1999)),

– soit en proposant des techniques permettant de trouver **des sous-optimaux** (comme les approches proposées par (Guestrin, 2003), (Chades, 2002), (Buffet, 2003))

– ou soit en proposant de nouveaux **mécanismes permettant de réduire cette complexité** (comme les COMM-MTDP (Pynadath *et al.*, 2002)).

Notre proposition appartient à ces deux dernières approches : nous serons amenés à proposer de nouveaux mécanismes d'organisation distribués pour construire de manière décentralisée des solutions approchées à partir d'heuristiques tirant parti de ces mécanismes. Pour ce faire, notre proposition consiste à tirer parti de la structure multi-agents des systèmes que nous considérons.

3. Notre proposition

3.1. *Hypothèses*

La caractéristique principale d'un système multi-agents réside dans la notion d'interaction pouvant être définie comme des couplages de comportements entre agents, ou "une mise en relation dynamique de deux ou plusieurs agents par le biais d'un ensemble d'actions réciproques" (Ferber, 1997). Les agents d'un même système s'influencent mutuellement pour construire une solution au problème posé.

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 et tirer parti de l'aspect collectif du système.

Comme on souhaite faire des apprentissages décentralisés, cela nécessite pour un agent d'avoir une bonne évaluation des conséquences à long terme d'une décision en particulier en ce qui concerne les conséquences de ses actions sur les autres agents.

Or l'interaction entre agents n'apparaît pas explicitement dans un DEC-POMDP et lorsqu'un agent effectue une action, il ne sait pas quelle en est sa portée. Il peut en effet influencer la trajectoire suivie par un autre agent et n'a aucun moyen de communiquer avec lui pour savoir quelles sont les conséquences collectives de son action.

Nous proposons donc d'intégrer la notion d'interaction directe dans le cadre DEC-PODMP pour permettre aux agents de raisonner collectivement sur certains couplages. L'interaction directe présente alors plusieurs intérêts :

– Tout d'abord, elle permet de représenter explicitement au niveau des agents la présence d'autres agents dans le système et permet aux agents de raisonner sur leurs moyens d'interagir avec les autres.

– 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 pour inciter d'autres agents à effectuer une action utile pour la collectivité.

– Enfin, elle définit une structure au problème à partir d'un concept facilement interprétable et fondamental dans les systèmes multi-agents : l'interaction.

3.2. *Aperçu du formalisme*

Le formalisme Interac-DEC-POMDP que nous allons présenter est ainsi issu des DEC-POMDPs auxquels nous avons ajouté le concept d'interaction. Ce formalisme se décompose ainsi en deux modules : un module d'action au sein duquel les agents agissent sur l'environnement et un module d'interaction au sein duquel les agents proches peuvent prendre une décision jointe locale par concertation.

Les agents ont donc plusieurs possibilités : ils peuvent agir directement sur l'environnement par des politiques d'action mais ils peuvent aussi interagir directement avec les autres agents grâce à des politiques d'interaction pour prendre une décision à plusieurs sans impliquer l'ensemble des agents du système.

Les sections suivantes décrivent de manière plus précise ce cadre formel. La question consistant pour les agents à apprendre comment agir et comment interagir avec les autres sera reléguée à la partie 4 qui traite de notre seconde proposition : un algorithme d'apprentissage décentralisé adapté au formalisme Interac-DEC-POMDP.

3.3. *Définition des interactions directes*

3.3.1. *Caractéristique d'une interaction directe*

Une interaction directe est définie comme une action mutuelle réciproque fondée sur des échanges d'information (Keil *et al.*, 2003). Alors que les conséquences d'une

interaction indirecte sont décidées par les lois de l'environnement (comme l'évaporation de phéromone dans le cadre d'algorithme fourmi), les conséquences d'une interaction directe sont décidées directement par concertation entre les agents impliqués.

L'interaction directe que nous considérerons

– est dirigée explicitement vers un destinataire dans le but de modifier son comportement sans que l'état de l'environnement ne soit affecté directement

– et ne fait pas intervenir directement l'environnement mais 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.

Pour respecter les principes de localité énoncés auparavant, nous souhaitons

– que cette interaction soit déclenchée par un agent pour disposer d'un système centré agent et pour respecter les localités des prises de décision.

– que cette interaction soit résolue à la suite de communications réduites entre agents proches

– que ces interactions n'impliquent que des communications ponctuelles et locales

Enfin, comme nous souhaitons pouvoir représenter des interactions qui peuvent restructurer les relations entre agents à l'exécution, une interaction n'implique pas toujours les mêmes agents, et un agent doit pouvoir décider avec qui il souhaite interagir.

3.3.2. *Structure des interactions*

Les interactions que nous considérons sont structurées selon la figure 1 :

– Les interactions possibles sont représentées par un ensemble de symboles d'interactions $I = \{I_k\}$. Une interaction I_k va être définie par les actions jointes que l'interaction propose et le nombre d'agents impliqués dans les interactions de ce type. Dans le cadre d'un jeu de sport collectif, l'interaction "demander une passe" sera représentée par un symbole $I_j \in I$. Les agents impliqués dans une interaction directe endossent implicitement des rôles par rapport à cette interaction (par exemple, l'agent qui demande la balle n'a pas un rôle symétrique par rapport à l'agent qui est sollicité).

– Un ensemble de résultats possibles $RI_k = \{RI_{k,l}\}$ est associé à chaque type d'interaction I_k. Ces résultats correspondent à des actions jointes définies de manière implicite et supposées connues. Par exemple, deux résultats sont associés à l'interaction "demander une passe" (I_j) : le résultat $I_{j,0}$ "effectuer la passe" et le résultat $I_{j,1}$ "la passe est refusée". Le résultat "effectuer la passe" peut être compris comme le déclenchement d'une action jointe entre les deux agents impliqués : l'agent receveur enverra la balle et l'agent émetteur déclenchera une action pour la réceptionner.

– Un ensemble de matrices de transition $TRI_{k,l} : agent^* \times S \rightarrow S$ [1] pour chaque Résultat $RI_{k,l}$. Cette matrice permet de déterminer l'état d'arrivée du système après application du résultat d'interaction. L'état d'arrivée dépend de l'état de départ et des agents impliqués dans l'interaction. Ainsi au résultat "effectuer la passe" est associée la matrice de transition consistant à faire passer le ballon du receveur à l'émetteur. Bien entendu, ces matrices modifient différemment l'état du système en fonction des agents émetteur et receveur.

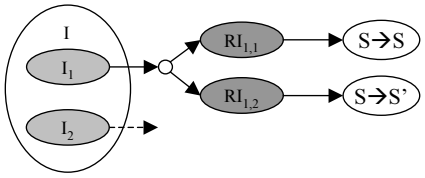

Figure 1. *Les interactions sont définies par des types (I) menant à plusieurs résultats possibles (RI) auxquels sont associés des matrices de transition (S->S')*

Lorsqu'un agent déclenche une interaction, en raison de ses perceptions partielles, il ne peut pas savoir si l'agent destinataire est en mesure de répondre et si l'interaction peut être exécutée. La fonction $possible$ est une fonction correspondant à la réponse du système à un déclenchement d'interaction. Cette fonction a pour objectif **de définir la structure des communications** . Il s'agit d'une fonction $possible : S \times I \times agents^* \rightarrow \{0, 1\}$ qui associe à toute configuration de l'environnement, toute interaction et tout tuple d'agents destinataires un booléen. Ce booléen détermine si l'interaction directe avec les autres agents est envisageable dans la situation donnée. Si le booléen est faux, les agents ne peuvent pas communiquer (par exemple parce qu'ils sont trop éloignés) et l'interaction déclenchée échoue systématiquement. Si le booléen retourné est vrai, les agents peuvent interagir, échanger de l'information et décider collectivement d'un résultat.

3.3.3. *Exécution d'une interaction*

Nous distinguerons par la suite deux phases dans une interaction :

– la phase de **déclenchement** consistant pour un agent à décider quelle interaction directe déclencher et avec quel(s) agent(s) interagir. Cette phase est la conséquence d'une décision purement individuelle et peut conduire à l'instanciation d'une interaction directe si les capacités de communication des agents sont suffisantes.

– la phase de **résolution** de l'interaction consistant, lorsqu'une interaction est instanciée, à choisir collectivement une action jointe parmi celles proposées par l'inter-

1. Le symbole $agent^*$ désigne l'ensemble des listes triées d'agents : la position d'un agent dans la liste désignant son rôle dans l'interaction.

action. Cette action jointe sera appelée 'résultat de l'interaction' et son exécution fera évoluer l'état global du système.

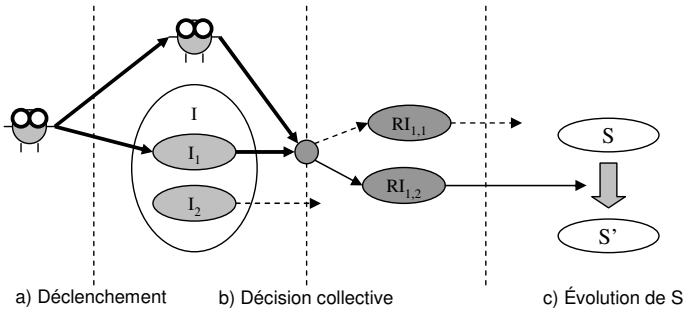

Figure 2. *L'exécution d'une interaction se fait en trois phases a) Déclenchement, b) Décision collective, c) Exécution du résultat de l'interaction*

L'exécution d'une interaction se fait en plusieurs étapes (fig 2 et algorithme 1) :

– 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)

– Un test est effectué pour vérifier si les agents peuvent communiquer et si l'interaction peut être instanciée. Ce test consiste à évaluer la valeur de la fonction $possible$ à partir de la configuration locale du système. Si ce test s'avère être négatif, aucune communication n'est possible et l'interaction échoue.

– Dans le cas contraire, 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 (cf fig 2 b). Nous verrons par la suite comment représenter ces politiques de manière distribuée.

– Enfin, le résultat choisi est exécuté et modifie l'état du système en fonction de $TR_{k,l}$. (cf fig 2 c)

3.3.4. *Intérêt de cette représentation de l'interaction*

En représentant l'interaction directe de cette manière, nous assurons plusieurs contraintes propres aux systèmes multi-agents :

– Les agents décident individuellement des déclenchement des interactions directes et toute évolution du système reste à l'initiative des agents ce qui leur préserve leur autonomie et leur pro-activité.

– Le fait d'avoir des déclenchements individuels permet de reconfigurer à l'exécution les relations entre agents puisqu'un agent va pouvoir décider à chaque instant avec qui il souhaite interagir.

Algorithme 1 Executer une interaction

Observation de l'agent i : $o_i \leftarrow O(s)_i$
Choix de l'interaction : $(I_k, agent_j) \leftarrow \pi_{decl,i}(o_i)$
si $possible(I_k, agent_j)$ **alors**
  Observation de l'agent j : $o_j \leftarrow O(s)_j$
  Échanges locaux et choix du résultat de l'interaction : $(RI_{k,l} \leftarrow \Pi_{i,j}(o_i, o_j))$
  Exécution du résultat : $s \leftarrow TRI_{k,l}(S, i, j)$
  Fin de l'interaction
fin si

– Le fait d'avoir des politiques collectives permet aux agents de prendre en compte les perceptions des autres agents.

– Enfin, les interactions sont basées uniquement sur des échanges locaux d'information lors de la résolution d'une interaction et ne nécessite pas des communications constantes ou à longue portée entre agents.

Maintenant que nous avons décrit la notion d'interaction directe et leur utilisation par les agents, nous pouvons décrire de manière globale ce qu'est le formalisme Interac-DEC-POMDP.

3.4. *Formalisme Interac-DEC-POMDP*

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

3.4.1. *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 d'action $\pi_i : \Gamma_i \times A_i \rightarrow [0, 1]$.

L'exécution du module d'action consiste à exécuter un cycle d'un DEC-POMDP.

3.4.2. *Module d'interaction*

Le module d'interaction est défini par $< I, RI, TRI >$ en complément au module d'action. Les comportements des agents sont définis par leur politique individuelle de déclenchement $\pi_{decl,i}$ et les politiques de résolution pour chaque interaction et chaque ensemble d'agents Π_{i,j,I_k}.

L'exécution du module d'interaction consiste à exécuter séquentiellement la politique de déclenchement (avec potentiellement l'instanciation et la résolution de l'interaction) de chacun des agents (dans un ordre aléatoire).

3.4.3. *Exécution d'un Interac-DEC-POMDP*

L'exécution d'un cycle d'un Interac-DEC-POMDP consiste à exécuter séquentiellement un cycle du module d'action suivi d'un cycle du module d'interaction. Cette exécution est décrite formellement par l'algorithme 2.

Algorithme 2 Execution d'un Interac-DEC-POMDP pendant t_{fin} pas de temps

$t \leftarrow 0$
répéter
  Module d'action
  Observation : $O = (o_0, ...o_n) \leftarrow O(s)$
  pour tout agents $i \in [0..n]$ **faire**
    choix de l'action i : $a_i \leftarrow \pi_i(o_i)$
  fin pour
  action jointe : $a \leftarrow (a_0...a_n)$
  exécution de l'action jointe : $s' \leftarrow T(s, a)$
  récompense : $R \leftarrow R(s, a, s')$
  modification de l'état du système : $s \leftarrow s'$
  Module d'interaction
  Liste aléatoire des agents : $liste \leftarrow ordre_{aleatoire}$
  pour tout agent $\in liste$ **faire**
    Observation de l'agent i : $o_i \leftarrow O(s)_i$
    Choix de l'interaction : $(I_k, agent_j) \leftarrow \pi_{decl,i}(o_i))$
    si $possible(I_k, agent_j)$ **alors**
      Début de l'exécution de l'interaction
      Observation de l'agent j : $o_j \leftarrow O(s)_j$
      Échanges locaux et choix du résultat de l'interaction : $(RI_{k,l} \leftarrow \Pi_{i,j}(o_i, o_j)$
      Exécution du résultat : $s \leftarrow TRI_{k,l}(S, i, j)$
      Fin de l'exécution de l'interaction
    fin si
  fin pour
  $t \leftarrow t + 1$
jusqu'à $t > t_{fin}$

3.4.4. *Résolution*

Les Interac-DEC-POMDPs permettent ainsi de représenter des actions individuelles et des interactions directes dans un même formalisme. Ils permettent de définir une nouvelle classe de problèmes dans laquelle les agents peuvent interagir directement. Résoudre une instanciation, consiste à trouver pour un tuple $< S, A_i, T, R, \Gamma_i, O, I, RI, TRI >$ l'ensemble des politiques individuelles π_i pour tout i, l'ensemble des politiques individuelles de déclenchement $\pi_{trig,i}$ pour tout i, l'ensemble des politiques de résolution d'interaction Π_{i,j,I_k} pour tout i, j, k permettant de maximiser la moyenne des sommes des récompenses globales reçues

par le système à l'exécution à partir d'un état s_0.

En d'autres termes, 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.

3.5. *Une sous classe de problème*

3.5.1. *Description*

Cette partie a pour objectif de présenter la sous-classe d'Interac-DEC-POMDP sur laquelle nous allons nous focaliser. 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 à partir desquels un agent peut tenter d'exercer une interaction sur un autre agent sont en faible nombre.

3.5.2. *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 attribuée.

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. Ainsi, si un agent souhaite interagir avec un autre, il va tout d'abord devoir se déplacer au cours de la phase d'action pour être côte à côte avec lui.

L'objectif de l'algorithme que nous proposerons dans la partie suivante 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.

3.6. *Bilan sur le formalisme Interac-DEC-POMDP*

En introduisant explicitement l'interaction comme élément de premier ordre du cadre Interac-DECP-POMDP, on dispose d'un nouveau cadre formel plus expressif et plus proche des systèmes multi-agents puisque certains couplages peuvent être définis par le concepteur comme le préconise les approches OCMAS (Ferber *et al.*, 2003).

Le cadre Interac-DEC-POMDP fournit en outre une structuration naturelle des systèmes multi-agents en utilisant ce concept d'interaction. Il se révèle ainsi particulièrement adapté à l'utilisation de techniques de résolution distribuée comme celles proposées par (Guestrin, 2003) et présentée dans la partie 2.3.2.3 tirant parti de la structure du problème pour construire les comportements. L'interac-DEC-POMDP nous semble en outre particulièrement adapté pour traiter des problèmes dans lequel les interactions directes prennent une place importante : comme les problèmes de *manufacturing control* où les machines vont pouvoir interagir pour échanger des produits intermédiaires et les problèmes de partage de tâches ou de ressources, un agent pouvant interagir avec un autre pour effectuer une redistribution partielle des ressources allouées. La prochaine partie de cet article va présenter comment utiliser l'interaction pour apprendre automatiquement les comportements des agents.

4. Algorithmique

4.1. *Objectif de l'algorithme*

Nous nous intéressons donc à un problème de coopération, dans un système perçu localement par chacun des agents, avec des récompenses partiellement observées et dont les seules influences possibles entre les agents du système résident dans les interactions. Il s'agit de 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. Il faut donc un mécanisme permettant de répartir les récompenses entre les agents à l'exécution du système.

– 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. Cette interdépendance nécessite des mécanismes permettant de synchroniser l'ensemble des politiques des agents.

4.2. *Présentation générale de l'algorithme d'apprentissage*

Notre objectif dans cette partie est de tirer parti d'algorithmes d'apprentissage individuels construisant des comportements d'agents égoïstes pour produire des comportements collectifs. Cet algorithme va donc devoir répondre à deux questions

– comment construire la résolution des interactions à partir des comportements individuels des agents ?

– comment adapter les comportements individuels aux interactions apprises dans le système ?

Ces deux questions auront leur réponse propre :

– la construction des politiques d'interaction à partir des politiques individuelles sera fondée sur une heuristique consistant à maximiser la somme des satisfactions des agents impliqués ce qui permet l'apparition de comportements altruistes utiles pour le groupe. Il s'agit d'une approche similaire aux heuristiques développées dans (Meuleau *et al.*, 1998) présentée dans le chapitre 2.3.2.3 mais qui peut désormais être plus souple et se fonder sur le concept d'interaction qui a été introduit.

– la construction des politiques individuelles utilisera des partages de récompenses au cours des interactions pour inciter un agent à reproduire une situation intéressante pour le système même s'il n'en recevait pas directement un bénéfice. Cela consiste à répartir automatiquement et correctement la tâche entre les agents pour tirer parti de l'aspect égoïste de leurs comportements. Ces techniques sont proches de celles proposées dans (Schneider *et al.*, 1999) présentées dans le chapitre 2.3.2.3 mais ces techniques n'utilisent plus des communications constantes puisqu'elles reposent sur l'interaction directe qui a été introduite dans le formalisme.

4.3. *Apprentissage collectif des politiques*

Dans cette partie, nous cherchons à construire les politiques les plus proches possibles de l'optimal à partir de l'approche présentée dans la partie précédente. Il s'agit pour les agents d'apprendre simultanément quelles actions effectuer et quelle interaction déclencher.

4.3.1. *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 (Watkins *et al.*, 1992) pour synchroniser l'exécution des politiques. A l'exécution du système, tous les agent effectuent simultanément un apprentissage décentralisé. 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.

4.3.2. *Représentation des politiques*

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 (Watkins *et al.*, 1992), 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 $v(s) = max_a Q(s, a)$.

Chacune des politiques (politique d'action, de déclenchement ou d'interaction) sera représentée au niveau de l'agent par des Q-valeurs individuelles (respectivement Q-valeurs d'action, de déclenchement et d'interaction).

4.4. *Cycle d'apprentissage*

4.4.1. *Exécution des actions*

Un agent décide de son action en fonction d'une politique déterminée par ses Q-valeurs d'actions. 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.

4.4.2. *Exécution des interactions*

Chaque agent à son tour évalue individuellement l'interaction qu'il souhaite déclencher (échange de seau ou non dans notre cas) à partir de ses Q-valeurs de déclenchement. 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 d'action individuelles après exécution de l'interaction selon la formule [2] :

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

2. Pour effectuer cette maximisation, il suffit que les agents impliqués dans l'interaction s'échangent leurs Q-valeurs d'actions pour les états $TR(s'1)$ et $TR(s'2)$. Ce calcul peut donc être effectué localement par les agents et ne nécessite pas de vision globale de l'environnement.

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 (Thomas *et al.*, 2006) 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 sous forme de récompense sociale. Ces récompenses sociales ont pour objectif d'inciter au mieux les agents égoïstes à reproduire leurs actions lorsqu'elles sont utiles au groupe.

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

4.4.3. *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 sociales reçues au cours des interactions $r_{s,i}$

– de la Q-valeur de l'état d'arrivée s''_i après interactions

La formule de mise à jour des Q-valeurs est analogue à celle du Q-learning (Watkins *et al.*, 1992) ($\gamma \in [0, 1]$ est le discount factor et α le coefficient d'apprentissage)

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

4.4.4. *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.

4.5. *Algorithme*

L'algorithme d'apprentissage est présenté formellement dans l'algorithme 3.

Algorithme 3 Apprentissage pendant t_{fin} pas de temps

$t \leftarrow 0$
répéter
  Module d'action
  Observation : $O = (o_0, ...o_n) \leftarrow O(s)$
  pour tout agent $i \in [0..n]$ **faire**
    choix de l'action i : $a_i \leftarrow \pi_i(o_i)$
  fin pour
  action jointe : $a \leftarrow (a_0...a_n)$
  exécution de l'action jointe : $s' \leftarrow T(s, a)$
  récompense pour chaque agent : $r_i \leftarrow [R(s, a, s')]_i$
  modification de l'état du système : $s \leftarrow s'$
  Module d'interaction
  Liste aléatoire des agents : $liste \leftarrow ordre_{aleatoire}$
  pour tout agent $\in liste$ **faire**
    Observation de l'agent i : $o_i \leftarrow O(s)_i$
    Choix de l'interaction : $(I_k, agent_j) \leftarrow \pi_{decl,i}(o_i)$
    si $possible(I_k, agent_j)$ **alors**
      Observation de l'agent j : $o_j \leftarrow O(s)_j$
      Échanges locaux et choix du résultat de l'interaction : $RI_{k,l} \leftarrow \Pi_{i,j}(o_i, o_j)$
      • Calcul des récompenses sociales : $r_{soc,i,j}$
      Exécution du résultat : $s \leftarrow TRI_{k,l}(S, i, j)$
      • Mise à jour des Q-valeurs d'interaction
      • Mise à jour des Q-valeurs de déclenchement
      Fin de l'interaction
    fin si
  fin pour
  • Mise à jour des Qvaleurs d'action
  $t \leftarrow t + 1$
jusqu'à $t > t_{fin}$

4.6. *Bilan de l'algorithme*

L'approche que nous proposons réside dans plusieurs propositions jointes :

– la première consiste à synchroniser les agents à partir d'une mémoire à long terme et de techniques d'apprentissage par renforcement (Watkins *et al.*, 1992)

– la seconde réside dans des heuristiques qui résolvent les interactions en fonction des Q-valeurs individuelles et qui permettent de répartir automatiquement les récompenses et la tâche dans le système au cours des interactions.

5. 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.

5.1. *Résultats bruts*

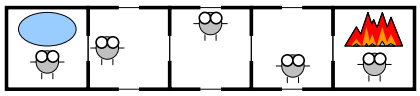

Figure 3. *Exemple 1*

Sur un exemple constitué de 5 agents dans un couloir (cf fig 3), 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 4) 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 5) et même si elles évoluent encore à cause d'une re-répartition des récompenses dans le système, la politique reste inchangée. A partir du de temps 200000 (cf fig 5), ϵ a pour valeur 0. Les agents ont alors des comportements déterministes tout en continuant d'échanger des récompenses sociales ce qui explique le fait que les Q-valeurs évoluent plus rapidement.

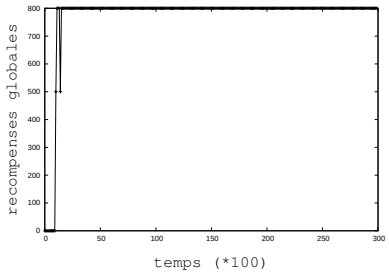

Figure 4. *Récompenses reçues par exploitation des politiques pendant 20 pas de temps ($\epsilon = 0$) au cours de l'apprentissage en fonction de t*

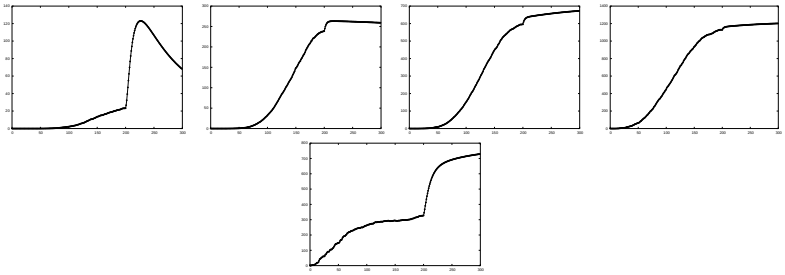

Figure 5. *Évolution de la somme des Q-valeurs pour chaque agent durant l'apprentissage*

5.2. *Résultats Comparés*

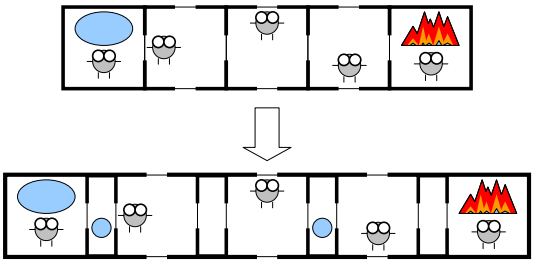

Figure 6. *DEC-POMDP modélisant le problème de l'exemple 1*

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 (cf figure 6).

Du fait du problème du *credit assignment* (cf partie 2.3.2.2), les comportements des agents oscillent et aucun comportement global n'émerge (cf fig 7 et 8). 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. Les interactions et l'algorithme que nous avons proposé constituent donc bien **une réponse au problème du credit assignement**.

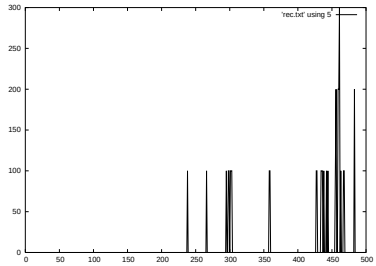

Figure 7. *Récompenses reçues sans interaction par exploitation au cours de l'apprentissage*

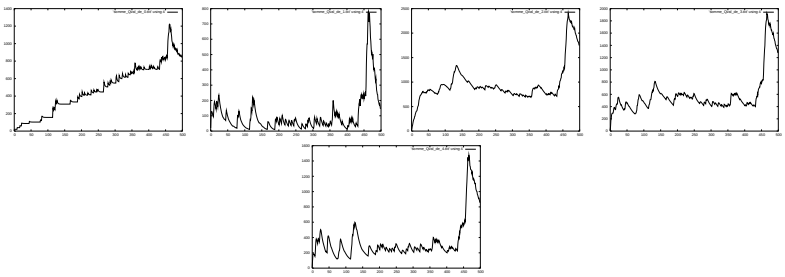

Figure 8. *Somme de Q-valeurs sans interaction*

5.3. *Augmentation du nombre d'agents*

L'approche proposée du fait d'apprentissages décentralisés permet d'appréhender des problèmes avec un nombre d'agents plus élevé. Nous avons ainsi conduit des expériences avec 25 agents (cf figure 9 a)) pour mettre en évidence des première propriétés de passage à l'échelle. Chaque politique individuelle est construite à partir de Q-valeur individuelles : 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

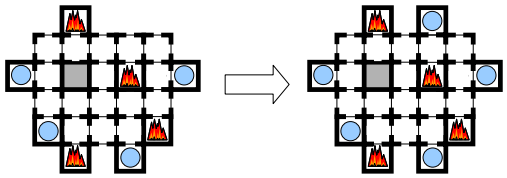

Figure 9. *Exemple 2 : problème des pompiers avec 25 agents*

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 relativement bons résultats (cf figure 10).

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.

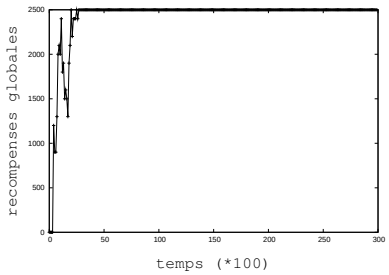

Figure 10. *Récompenses globales reçues pour l'exemple 2*

5.4. *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 11 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 9 b et 11) 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.

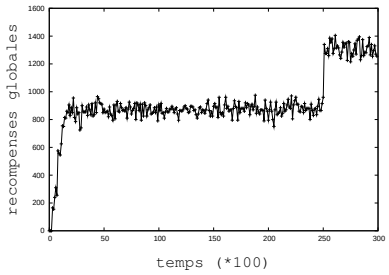

Figure 11. *Récompenses globales reçues avec exploration et re-adaptation*

5.5. *Bilan des résultats obtenus*

Les résultats des expériences semblent ainsi montrer

– que les techniques d'apprentissage proposées permettent aux agents de s'organiser de manière autonome en tirant parti des interactions pour résoudre la tâche

– que les comportements individuels et collectifs sont dotés de plasticité et peuvent se réadapter en fonction des autres organisations et des perturbation au sein de l'environnement

En cela, ces techniques nous laissent supposer qu'elles constituent un moyen à plus long terme de produire des systèmes capables de présenter des **comportements d'auto-organisation** pour la résolution automatique de tâches collectives.

6. Discussion

6.1. *Avantages de l'approche*

Cette approche présente un certain nombre d'avantages 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 développée 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

6.2. *Positionnement*

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

– Dans (Simonin, 2001), 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 (Simonin, 2001) mais les propositions de Simonin présentent en contrepartie des capacités d'adaptation plus importantes pour la tâche.

– Dans (Schneider *et al.*, 1999), 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.

– (Buffet, 2003) 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.

6.3. *Limites*

Ces travaux se heurtent néanmoins à un certain nombre de difficultés que nous souhaitons résoudre par la suite. En particulier, 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 heuristiques sont envisageables. 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.

Le dilemme exploration/exploitation habituel dans les techniques d'apprentissage s'exprime de manière particulière dans notre cas : lorsque les agents explorent leur environnement, il leur est difficile d'acquérir de l'information pertinente puisque les autres agents se comportent de manière aléatoire ; lorsque les agents exploitent, ils ne cherchent plus d'autres solutions alors que de nouvelles possibilités peuvent être offertes. Une évolution distribuée des coefficients d'exploration ϵ nous semble ainsi une piste à développer dans le futur.

6.4. *Perspectives à long terme*

L'approche originale proposée ouvre de nombreuses perspectives :

– en formalisant la notion d'interaction dans les systèmes markoviens, elle ouvre la porte à une nouvelle panoplie d'algorithmes à déployer tirant parti de la notion d'interaction pour construire des systèmes distribués et pour envisager la résolution de problèmes d'optimisation collectifs.

– en cherchant à formaliser ce qu'est un système multi-agents, ces techniques proposent des cadres opératoires permettant de manipuler des concepts éprouvés. En particulier, les approches OCMAS (Ferber *et al.*, 2003) proposent de se concentrer sur le placage d'organisations élémentaires définies a priori pour résoudre des problèmes collectifs. Des extensions de notre approche, en formalisant explicitement la notion d'organisation élémentaire (proche de notre concept d'interaction directe) pourraient alors proposer un cadre opératoire pour apprendre automatiquement l'utilisation de ces organisations et la distribution des rôles pour résoudre un problème collectif.

– en centralisant certaines prises de décision, l'interac-DECPOMDP propose de nouveaux problèmes définis formellement et caractérisés par une complexité qui reste à étudier. Cette étude nécessite de se placer dans le cadre de la planification centralisée pour pouvoir comparer la complexité des problèmes interac-DECPOMDP à ceux du DECPOMDP. Cette comparaison constitue un de nos prochains axes de recherche, nécessitera vraisemblablement de longs développements mais pourrait amener des résultats forts tirant parti de l'aspect formel de notre approche.

A plus long terme, l'objectif de nos travaux est de proposer des techniques pour obtenir une **"auto-organisation rationnelle"** : la propriété qu'ont les agents de s'organiser à l'aide d'interactions ou d'organisations élémentaires tout en raisonnant individuellement et rationnellement sur les rôles qu'ils endossent.

7. Conclusion

Dans cet article, nous avons :

– proposé un cadre formel Interac-DEC-POMDP qui permet de représenter des actions et des interactions d'un système multi-agent dans un cadre homogène

– présenté une sous-classe dans laquelle les seules influences possibles entre agents sont constituées par les interactions

– proposé un algorithme d'apprentissage par renforcement tirant parti d'apprentissages égoïstes et d'une répartition automatique des récompenses entre agents pour répondre au problème du *credit assignement* et pour construire les politiques d'actions et d'interaction des agents.

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

Nous avons ainsi montré dans cet article qu'il est possible d'utiliser des techniques d'apprentissage égoïstes pour produire des systèmes collectifs capables de s'adapter collectivement à une tâche globale à résoudre. 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 la capacité des agents à s'organiser rationnellement et de manière autonome.

8. Bibliographie

Baxter J., Bartlett P., « Infinite-Horizon Policy-Gradient Estimation », *Journal of Artificial Intelligence Research*, vol. 15, p. 319-350, 2001.

Becker R., Zilberstein S., Lesser V., « Decentralized Markov Decision Processes with Event-Driven Interactions », *AAMAS '04 : Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems*, 2004.

Bernstein D., Givan R., Immerman N., Zilberstein S., « The Complexity of Decentralized Control of Markov Decision Processes », *Mathematics of Operations Research*, vol. 27, n° 4, p. 819-840, 2002.

Bourjot C., Chevrier V., Thomas V., « A new swarm mechanism based on social spiders colonies : from web weaving to region detection », *Web Intelligence and Agent Systems : An International Journal - WIAS*, vol. 1, n° 1, p. 47-64, Mar, 2003.

Boutilier C., « Sequential Optimality and Coordination in Multiagent Systems », *IJCAI '99 : Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, p. 478-485, 1999.

Buffet O., Une double approche modulaire de l'apprentissage par renforcement pour des agents intelligents adaptatifs, PhD thesis, Université Nancy I, 2003.

Casteran J. C., Gelizes M. P., Glize P., « Des méthodologies orientées multi-agent », *8ièmes Journées Francophones Intelligence Artificielle Distribuée et Systèmes Multi-Agents*, Editions Hermès, p. 191-207, 2000.

Chades I., Planification distribuée dans les systèmes multi-agents à l'aide de processus décisionnels de Markov, PhD thesis, Université Nancy 1, 2002.

Dorigo M., Caro G. D., « The Ant Colony Optimization Meta-Heuristic », *in* D. Corne, M. Dorigo, F. Glover (eds), *New Ideas in Optimization*, McGraw-Hill, London, p. 11-32, 1999.

Ferber J., « Les systèmes multi-agents : un aperçu général », *Technique et science informatique*, vol. 16, p. 979-1012, 1997.

Ferber J., Gutknecht O., Michel F., « From agents to organizations : an organizational view of multi-agent systems », *Agent-Oriented Software Engineering IV : 4th International Workshop,*p. 185-202, 2003.

Flacher F., Sigaud O., « GACS : une approche ascendante pour la coordination spatiale », *Revue d'Intelligence Artificielle*, vol. 20, n° 1, p. 7-29, 2006.

Guestrin C., Planning Under Uncertainty in Complex Structured Environments, PhD thesis, Stanford University, August, 2003.

Guestrin C., Koller D., Parr R., « Multiagent Planning with Factored MDPs », *Advances in Neural Information Processing Systems (NIPS 2001)*, p. 1523 - 1530, 2001.

Hardin G., « the tragedy of the commons », *Science*p. 1243-1248, 1968.

Keil D., Goldin D., « Modeling Indirect Interaction in Open Computational Systems », *1st Int'l workshop on Theory and Practice of Open Computational systems (TAPOCS)*, 2003.

Lumer E. D., Faieta B., « Diversity and Adaptation in Populations of Clustering Ants », *From Animals to Animats 3*p. 501-508, 1994.

Meuleau N., Hauskrecht M., Kim K., Peshkin L., Kaelbling L., Dean T., Boutilier C., « Solving Very Large Weakly Coupled Markov Decision Processes », *AAAI/IAAI*, p. 165-172, 1998.

Pynadath D., Tambe M., « The communicative multi-agent team decision problem : analyzing teamwork : theories and models », *Journal of Artificial Intelligence Research*, vol. 16, p. 389-423, 2002.

Schneider J., Wong W., Moore A., Riedmiller M., « Distributed Value Functions », *Proceedings of the 16th International Conference on Machine Learning*, Morgan Kaufmann, San Francisco, CA, p. 371-378, 1999.

Simonin O., Le modèle satisfaction-altruisme, PhD thesis, Université Montpellier II, 2001.

Sutton R., Barto A., « Reinforcement learning : An introduction », 1998.

Szer D., « communication et apprentissage par renforcement pour une équipe d'agents », *Journées Francophones des Systèmes Multi-Agents, JFSMA2004*p. 175-186, 2004.

Thomas V., Bourjot C., Chevrier V., « Heuristique pour l'apprentissage automatique décentralisé d'interactions dans un système multi-agents réactif », *Reconnaissance de Forme et Intelligence Artcificielle, RFIA 06*, 2006.

Watkins C. J. C. H., Dayan P., « Technical Note Q-Learning. », *Machine Learning*, vol. 8, p. 279-292, 1992.

Wooldridge M., Jennings N. R., Kinny D., « The Gaia Methodology for Agent-Oriented Analysis and Design », *Autonomous Agents and Multi-Agent Systems*, vol. 3, n° 3, p. 285-312, 2000.

ANNEXE POUR LE SERVICE FABRICATION
A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER
LE FICHIER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :
   RIA.

2. AUTEURS :
   Vincent Thomas, Christine Bourjot, Vincent Chevrier *

3. TITRE DE L'ARTICLE :
   Construction de systèmes multi-agents par apprentissage collectif à base d'interactions

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :
   Apprentissage collectif par interaction

5. DATE DE CETTE VERSION :
   4 avril 2007

6. COORDONNÉES DES AUTEURS :

– adresse postale :

* Équipe MAIA - tranche C
Laboratoire LORIA
Campus scientifique - P. 239
54506 Vandoeuvre-les-Nancy, France.

vincent.thomas@loria.fr, christine.bourjot@loria.fr,
vincent.chevrier@loria.fr

– téléphone : 00 00 00 00 00

– télécopie : 03 54 95 85 08

– e-mail : vincent.thomas@loria.fr

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :
   LaTeX, avec le fichier de style `article-hermes.cls`,
   version 1.23 du 17/11/2005.

8. FORMULAIRE DE COPYRIGHT :
   Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :
   `http://www.revuesonline.com`

SERVICE ÉDITORIAL – HERMES-LAVOISIER
14 rue de Provigny, F-94236 Cachan cedex
Tél. : 01-47-40-67-67
E-mail : revues@lavoisier.fr
Serveur web : http://www.revuesonline.com