Actes des 25es Journées Francophones sur les Systèmes Multi-Agents

Catherine Garbay, Grégory Bonnet

25es Journées Francophones sur les Systèmes Multi-Agents

Caen, du 4 au 6 juillet 2017

dans le cadre de la Plate-Forme de l'Intelligence Artificielle

Table des matières

Tifaine Inguere, Valérie Renault, Florent Carlier.
Délégation de tâches sur la plateforme multi-agents embarquée MERMAID

Jean-Baptiste Louvet, Nathalie Chaignaud, Laurent Vercouter, Guillaume Dubuisson Duplessis, Jean-Philippe Kotowicz.
Modélisation d'une tâche collaborative à l'aide d'engagements sociaux

Jean-Pierre Muller, Sitraka Oliva Raharivelo.
Un méta-modèle pour représenter les normes dans un contexte multi-institutionnel territorialisé

Simon Pageaud, Veronique Deslandres, Vassilissa Lehoux, Salima Hassas.
SmartGov : architecture générique pour exploiter les données des smart cities et co-concevoir des politiques crédibles

Thomas Paris, Laurent Ciarletta, Vincent Chevrier.
Intégration d'un simulateur multi-agent dans une plateforme de co-simulation DEVS

Irène Velontrasina, Denis Payet, Rémy Courdier.
Modèle relationnel pour la coordination des comportements des agents

Jean-Paul Barthès.
BESSICA : un environnement de test pour la simulation d'agents cognitifs DEMO

Lauriane Huguet, Domitile Lourdeaux and Nicolas Sabouret.
VICTEAMS : une équipe de personnages virtuels autonomes pour la formation au sauvetage de blessés

Stratégie de découpe de tâche pour le traitement de données massives

Quentin Baert
quentin.baert@univ-lille1.fr

Anne-Cécile Caron
anne-cecile.caron@univ-lille1.fr

Maxime Morge
maxime.morge@univ-lille1.fr

Jean-Christophe Routier
jean-christophe.routier@univ-lille1.fr

Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France

Résumé

MapReduce est un patron de conception permettant de traiter un très grand volume de données distribuées sur un cluster de machines. Ses performances sont liées aux éventuelles distorsions des données. Pour contrer ces biais, nous proposons un système multi-agent adaptatif. Les agents interagissent durant l'exécution et l'allocation dynamique des tâches est le résultats de négociations afin de soulager l'agent le plus chargé et donc le temps global d'exécution. Dans cet article nous montrons comment, lorsqu'une tâche est trop coûteuse pour être négociée, un agent peut la découper afin d'en négocier les sous-tâches.

Mots-clés : *Résolution distribuée de problème, Négociation, Données massives, MapReduce*

Abstract

MapReduce is a design pattern for processing large data sets distributed on a cluster. Its performances are linked to data skews. In order to takle the latter, we propose an adaptive multi-agent. The agents interact during the job and the dynamic tasks allocation is the outcome of negotiations in order to relieve the most loaded agent and so the running time. In this paper, we show how, when a task is too expensive to be negotiated, an agent can split it in order to negotiate its sub-tasks.

Keywords: *Distributed problem solving, Negotiation, Big Data, MapReduce*

1 Introduction

La science des données vise à traiter de grands volumes de données pour y extraire de nouvelles connaissances (en anglais, *insight*). L'analyse de ces données massives nécessite de paralléliser les traitements comme cela peut l'être avec le patron de conception MapReduce [4]. Ce dernier tient son nom des fonctions sur lesquelles il s'appuie : la fonction *map* qui filtre les données en créant des couples *(clé, valeur)* et la fonction *reduce* qui les agrège. Le framework le plus populaire pour la distribution de MapReduce est Hadoop mais de nombreuses autres implémentations existent comme le framework Spark ou la base de données NoSQL distribuée Riak construite par Amazon Dynamo.

Le développeur d'une application distribuée qui s'appuie sur le patron de conception MapReduce doit connaître l'implémentation qu'il utilise (Hadoop par exemple) ainsi que les données à traiter et l'environnement d'exécution afin de paramétrer au mieux le job avant son exécution. Les choix d'implémentation et de paramétrage du développeur peuvent être remis en cause par une variation des données ou de l'environnement d'exécution. Dans [9] les auteurs identifient ainsi deux biais récurrents lors de la phase de *reduce*. Ces derniers, appelés biais de partitionnement et biais des clés coûteuses, sont ceux que nous cherchons à résoudre. Ils ont pour même conséquence un déséquilibre dans les charges de travail des reducers : le traitement global du job MapReduce est pénalisé par le reducer le plus chargé.

Nous proposons dans cet article un système multi-agents pour mettre en œuvre le patron de programmation distribuée MapReduce où des agents reducer négocient les tâches pour réduire la charge du reducer le plus chargé et donc le temps d'exécution. En particulier, l'adaptativité des SMA permet de contrer les différents biais de données liés à la phase reduce et ainsi optimiser l'exécution sans prétraitement des données. Nous apportons ici une solution au biais des clés coûteuses : certaines clés regroupent un grand nombre de valeurs. Pour y parvenir nos agents reducers appliquent de manière autonome une stratégie de découpe des tâches volumineuses, reposant sur des critères de décisions locaux, pour leur négociation afin de répartir la charge de travail de manière équitable entre les agents reducers. Ce processus adapte en continu la ré-

partition des calculs à la dynamique du traitement du job. Nous réduisons ainsi l'écart d'effort entre le reducer le plus chargé et celui le moins chargé. Cet équilibrage de la charge de travail permet d'accélérer le traitement du job MapReduce.

La section 2 présente le modèle MapReduce et différents travaux s'attaquant aux biais des reducers. Dans la section 3, après avoir introduit le protocole de négociation, nous présentons notre stratégie de découpe des tâches coûteuses. La section 4 présente des expérimentations qui mettent en évidence l'apport de notre système multi-agents adaptatif pour le traitement de job MapReduce. Finalement, la section 5 conclut et dresse quelques perspectives.

2 Travaux connexes

Les *jobs* MapReduce se composent deux ensembles de tâches, les tâches *map* et les tâches *reduce*, qui sont distribuées sur plusieurs machines. Le modèle de programmation MapReduce s'appuie sur ces deux fonctions fournies par l'utilisateur avec les signatures suivantes :

$$\begin{array}{rl} \textit{map}: & (K1, V1) \rightarrow list[(K2, V2)] \\ \textit{reduce}: & (K2, list[V2]) \rightarrow list[(K3, V3)] \end{array}$$

La figure 1 illustre le flux de données MapReduce tel qu'il est implémenté dans Hadoop :

1. le superviseur partage les données d'entrée en donnant une "tranche" des données à chaque mapper ;
2. les mappers appliquent la fonction *map* sur leur tranche et créent les couples intermédiaires clé-valeur $(key : K2, value : V2)$;
3. une fonction de partitionnement est appliquée sur la sortie des mappers afin de les diviser en sous-ensembles, un sous-ensemble par reducer de sorte que tous les couples avec la même clé soient envoyés au même reducer. La fonction de partitionnement peut être personnalisée afin de spécifier les clés qui doivent être traitées ensemble par le même reducer ;
4. les reducers agrègent les couples intermédiaires clé-valeur pour construire les couples $(K2, list[V2])$; ils exécutent la fonction *reduce* sur chaque groupe de valeurs associé à chaque clé ;
5. les couples finaux clé-valeur $(K3, V3)$ sont écrits dans un fichier du système de fichiers distribué.

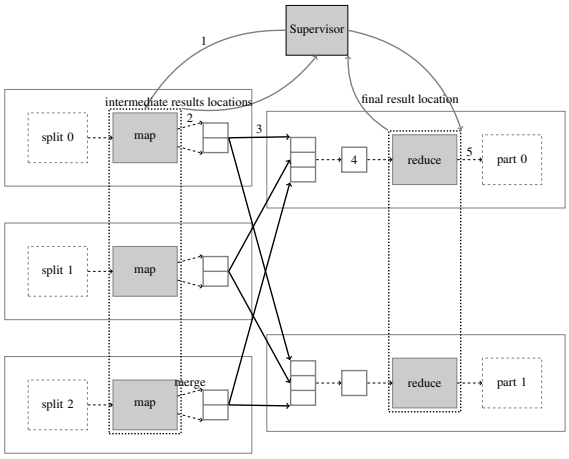


FIGURE 1 – Flux de données MapReduce.

Au niveau du système de fichiers, les tranches affectées en entrée des mappers peuvent être de grande taille, et découpées en blocs. De même, à la sortie des mappers, les données liées à une clé sont écrites dans une série de blocs, le reducer doit donc lire pour une clé fixée un groupe de valeurs formé physiquement de plusieurs blocs de données provenant de plusieurs mappers.

L'application du modèle MapReduce présente le risque de l'occurrence de biais (*skew*) liés aux données, qui viennent pénaliser l'efficacité de l'exécution des calculs. Certains travaux comme [10] et [17] prédisent la performance avec le profil des jobs en recueillant des données lors des exécutions précédentes. Nous ne voulons pas pré-traiter les données et supposons n'avoir aucune connaissance a priori des données d'entrée car le sur-coût computationel nécessaire peut être élevé, en particulier pour des grandes masses de données.

Le biais de partitionnement est lié au problème du déséquilibre qui peut être dû à l'affectation par les mappers des clés aux reducers. Sans connaissance a priori des données la fonction de partitionnement ne peut garantir une répartition équitable des clés et donc des tâches de calcul entre les reducers. Plusieurs travaux ([9, 2, 13]) visent à contrer ce biais. Dans [1] nous adressons également ce problème en adoptant une approche basée sur une allocation dynamique des tâches qui repose sur des négociations entre les agents reducers. A la différence des travaux précédents, notre solution est décentralisée et ne nécessite pas de paramétrage.

Un autre biais, appelé biais des clés coûteuses, est dû aux clés qui regroupent un nombre très important de valeurs, relativement aux autres

clés. Cette situation peut par exemple apparaître quand la répartition des données suit une distribution de type Zipf [11] où quelques éléments regroupent un grand nombre de valeurs pendant que de nombreux autres éléments ne sont concernés que par très peu de valeurs. C'est le cas pour le nombre d'offres par mot-clé émises sur la plate-forme marketing de Yahoo ! (cf section 4). Le phénomène d'engorgement des reducers lié à une telle situation a notamment été étudié dans [12]. Ce problème ne peut être résolu par une simple répartition différente des clés entre les reducers, les reducers responsables des clés coûteuses se trouvent nécessairement surchargés.

Si dans [9] ce biais de données est évoqué, la solution proposée par les auteurs ne le résout pas. Dans [7] les auteurs s'attaquent également aux problèmes des biais de données dans MapReduce. Ils proposent une méthode pour mieux répartir la charge de travail entre les reducers et identifient le problème des clés coûteuses. Mais argumentant que le modèle MapReduce exige qu'une même clé soit traitée par un seul reducer, ils estiment que les possibilités d'adaptation du système à ce biais sont très reduites. Leur proposition se limite alors à suggérer que le système alerte l'utilisateur que ses données rencontrent ce problème afin qu'il tente de trouver une solution ad hoc. Dans [13] les auteurs s'attaquent plus concrètement au problème. Ils proposent une solution dans laquelle les sorties des mappers sont découpées en blocs dont la taille doit être paramétrée a priori par le créateur du job. Les valeurs liées à une même clé peuvent ainsi se retrouver réparties dans plusieurs blocs. Les auteurs introduisent alors la notion de tâche de *reduce intermédiaire* qui travaille donc sur un sous-ensemble des valeurs d'une clé pour produire un *résultat intermédiaire*. Ces tâches étant de taille « calibrée » il est plus facile d'en répartir équitablement la charge. Les différents résultats intermédiaires d'une même clé doivent être agrégés lors d'une *phase de reduce finale*. L'approche adoptée est totalement centralisée, un nœud maître regroupe toutes les informations sur les tâches intermédiaires et organise la répartition des tâches sur les différents reducers. De plus la solution repose sur des paramètres que doit définir a priori le créateur du job, ce qui implique qu'il ait une connaissance préalable de la forme des données. Enfin la découpe des tâches en blocs est systématique.

Dans cet article, nous adoptons une approche similaire en découpant les tâches liées aux clés coûteuses en sous-tâches. Ces dernières peuvent ensuite être négociées et donc dynamiquement redistribuées entre les reducers afin d'obtenir une distribution plus équitable de la charge de travail. Cependant à la différence de [13] la découpe des tâches coûteuses n'est réalisée qu'à la demande, lorsqu'elle permet effectivement d'atteindre une répartition plus équitable des tâches. De plus notre mécanisme de découpe ne nécessite pas de définir a priori un paramètre déterminant la taille de la découpe.

Comme expliqué précédemment, le découpage des tâches nécessite de modifier légèrement le patron de conception, en décomposant la phase de reduce en une phase intermédiaire et une phase finale. Plus formellement, on définit maintenant trois fonctions :

$$\begin{aligned} \textit{map}: &\quad (K1, V1) \rightarrow list[(K2, V2)] \\ \textit{IR}: &\quad (K2, list[V2]) \rightarrow (K2, list[V2]) \\ \textit{FR}: &\quad (K2, list[V2]) \rightarrow list[(K3, V3)] \end{aligned}$$

Etant donnée une fonction reduce R, le programmeur doit donc la décomposer en deux fonctions IR (*intermediate reduce*) et FR (*final reduce*) telles que pour toute clé k et les valeurs associées S,

$$R(k, S) = FR(k, < IR(k, S_1); \cdots, IR(k, S_n) >)$$

quelle que soit la partition de l'ensemble de valeurs $S = S_1 \cup \cdots \cup S_n$.

La phase reduce sert à agréger les données. Dans [6], les auteurs identifient trois familles de fonctions d'agrégation : les fonctions distributives où le reduce, le reduce intermédiaire et final sont une seule et même fonction (par exemple `sum`, `min`, `max`, `count` sont distributives) ; les fonctions algébriques qui peuvent être décomposées en utilisant un nombre borné de fonctions distributives (par exemple `avg` va être décomposée en utilisant un reduce intermédiaire calculant une somme (`sum`) et un nombre de valeurs (`count`), la division de la somme totale par le nombre total de valeurs se faisant dans le reduce final) ; et les fonctions holistiques qui ne sont ni distributives ni algébriques (comme la médiane). Il est difficile de trouver une "bonne" décomposition d'une fonction holistique, en particulier de trouver une fonction intermédiaire efficace qui ne nécessite pas de mémoriser trop d'informations. Dans la suite de l'article, nous ne considérons que des fonctions distributives ou algébriques.

D'une manière générale, nous adoptons ici une approche comportementale pour la résolution distribuée de problème, approche qui englobe

la stigmergie [18], la résolution distribuée de contraintes [3] ou la négociation [15]. On peut remarquer que, dans une large partie de la littérature sur les mécanismes de négociation, les agents favorisent leur intérêt individuel. A l'inverse, dans notre contexte de résolution distribuée de problème, les agents ont un but commun qui prime sur leur intérêt individuel. Contrairement à [16], notre résolution ne consiste pas affecter une fois pour toute des paquets de ressources en fonction des préférences des agents mais d'itérer les négociations de tâches individuelles en fonction d'une estimation locale des tâches restant à réaliser. Contrairement à [3], le problème abordé ici est dénué de contraintes d'ordonnancement et les agents n'ont ni des capacités ni des compétences limitées. A notre connaissance, l'unique SMA qui implémente le patron MapReduce s'appuie sur des agents mobiles pour assurer la réplication du code et des données afin de garantir la tolérance aux pannes [5]. Toutefois, ce travail ne met en œuvre aucune technique d'auto-organisation [14] pour que le système multi-agents s'adapte aux données ou à l'environnement informatique. Pour ce faire, nous avons opté ici pour les techniques de négociation multi-agents.

3 Proposition

Notre modèle corrige à ce jour deux biais de données à savoir : le biais de partionnement des clés et le biais des clés coûteuses.

Pour résoudre le biais de partitionnement, nos agents reducers possèdent une architecture complexe et utilisent un protocole de négociation basé sur le Contract Net Protocol afin de se répartir les tâches dynamiquement en cours d'exécution et d'équilibrer leurs charges de travail (ou contributions). Ce protocole a été présenté en détails dans [1]. Nous en rappelons les principes généraux au travers d'un exemple dans la section 3.1. Nous adressons ici le biais des clés coûteuses. Nous présentons un mécanisme de découpe de tâches basé sur les blocs de données qui les composent. Cette division permet aux reducers de partiellement traiter une tâche (grâce notamment à l'introduction d'une fonction de reduce intermédiaire) et d'affiner encore l'équilibre de la charge de travail. La section 3.2 décrit le principe de la découpe de tâche ainsi que les conditions nécessaires pour qu'un reducer amorce cette opération. Les sections 3.3 et 3.4 présentent dans un premier temps une heuristique de découpe de tâche puis la stratégie effective qui utilise l'heuristique précédente en considérant les blocs de données. Lorsqu'une tâche est divisée, elle est traitée par plusieurs reducers. Se pose alors la question de la cohérence des résultats et de la construction du résultat définitif de la tâche. La section 3.5 introduit le rôle de *reducer final* dont l'objectif est d'assurer qu'une tâche reduce divisée et exécutée sur plusieurs nœuds aboutisse finalement à un unique résultat par agrégation des multiples résultats intermédiaires.

3.1 Protocole de négociation

Afin d'illustrer la délégation des tâches par la négociation, nous considérons ici une enchère particulière au sein d'un job MapReduce simple.

Nous supposons que la phase des mappers a été réalisée et que les tâches reduce sont initialement allouées à un ensemble de quatre reducers, $\Omega = \{1, 2, 3, 4\}$. Nous appelons *contribution* d'un reducer la somme des coûts des tâches de reduce qui lui incombent. Nous nous focalisons sur la distribution de tâches à l'instant t telle que les contributions individuelles sont : $c_1(t) = 10$, $c_2(t) = 8$, $c_3(t) = 3$ et $c_4(t) = 5$, où $c_i(t)$ est la contribution de l'agent i au temps t (voir Figure 2a).

Les agents vont engager des négociations dont le but est de diminuer la contribution de l'agent le plus chargé. Afin de diminuer sa contribution, le reducer 1 initie une enchère pour la tâche τ de coût 3 ($c_\tau = 3$) via un cfp (*call for proposal*) aux autres reducers (voir figure 2b). Un cfp inclut la contribution de l'initiateur (ici $c_1(t)$) et le coût de la tâche à négocier (ici c_τ).
Pour déterminer s'il peut gérer la tâche τ au temps $t + 1$, le reducer i ($i \in \Omega \backslash \{1\}$) doit satisfaire le critère d'acceptabilité suivant :

$$c_i(t) + c_\tau < c_1(t)$$

Les reducers qui satisfont ce critère feront baisser la contribution la plus haute s'ils accueillent la tâche τ. Ils font donc une proposition pour τ alors que les autres refusent de participer à la délégation de tâche. Ainsi, le reducer 2 ne peut pas réaliser τ, sinon sa contribution résultante $c_2(t) + c_\tau$ serait plus élevée que $c_1(t)$. Pendant ce temps, les reducers 3 et 4 font des propositions pour τ en envoyant leurs contributions au reducer 1 (voir figure 2c).
Le reducer 1 reçoit donc des propositions des agents $\Omega' = \{3, 4\}$. Il choisit d'attribuer τ à l'enchérisseur le moins chargé, en appliquant le critère de sélection suivant :

$$\underset{j\in\Omega}{\operatorname{argmin}}(c_j(t))$$

De cette manière, le reducer 1 accepte la proposition du reducer 3 et rejette celle du reducer 4 (voir figure 2d).

Après la négociation (à l'instant $t+1$), on observe que :

— la tâche τ appartient maintenant au reducer 3 ;

— les nouvelles contributions sont $c_1(t+1) = 7$, $c_2(t+1) = 8$, $c_3(t+1) = 6$ et $c_4(t+1) = 5$.

En conséquence, le reducer 2 a maintenant la contribution maximale. Toutefois, on peut observer que la contribution du reducer le plus chargé a diminué car nous avons $c_2(t+1) < c_1(t)$. La négociation mène à une allocation plus efficace où la charge est plus équitablement répartie (voir figure 2e).

3.2 Découpe de tâche : amorce et principe

Le biais des clés coûteuses ne peut être résolu ni par une fonction de partition classique ni uniquement par notre processus de négociation (cf. Figure 3). Afin de permettre aux reducers avec des tâches trop coûteuses de faire décroître leur contribution à l'aide d'enchères, nous proposons de diviser ces tâches en sous-tâches moins coûteuses et négociables. Afin de réduire le coût lié aux négociations, le découpage n'est réalisé que si nécessaire. Dans ce but, nos reducers sont construits avec deux fonctions de reduce : la fonction *IR* et la fonction *FR* (voir section 2) et sont ainsi en mesure de traiter une sous-tâche de reduce (grâce à la fonction *IR*) comme une tâche complète (grâce à la fonction *FR*).

Considérons une population de n reducers. Pour que le reducer i puisse débuter une découpe de tâche, il doit remplir des conditions de divisibilité :

1. ne pas être le reducer le moins chargé du système, c.à.d. qu'il existe m reducers ($1 \leq m \leq n-1$) moins chargés que i ;
2. être en état de pause, c.à.d. que selon les croyances de i concernant les contributions des autres reducers de son réseau d'accointances, il ne peut pas déléguer de tâche ;
3. que la plus coûteuse de ses tâches soit au moins composée de deux blocs de données ; une division de celle-ci pouvant ainsi donner au moins deux sous-tâches.

Lors d'une découpe de tâche, l'objectif de i est de diminuer sa contribution. L'idée est de découper la tâche la plus coûteuse de i en $k+1$ sous-tâches de telle sorte que $1 \leq k \leq m$. La répartition des k sous-tâches est négociée de telle sorte qu'à l'issue de ces négociations, aucun ne soit plus chargé que i avant ces négociations.

Les k sous-tâches sont créées de sorte à contenir le même nombre de blocs de données et donc à avoir approximativement le même coût, facilitant ainsi leur future délégation. Ces sous-tâches pourront être également découpées.

3.3 Heuristique de découpe de tâches

L'heuristique de découpe de tâches repose sur les croyances qu'a le reducer des contributions de ses accointances et sur la différence entre sa contribution et chacune de celles-ci.

Définition 1 (Delta de contribution)**.** *Soit $\Omega = \{1, \ldots, n\}$ une population de n reducers. Au temps t, chaque reducer i de contribution $c_i(t)$ possède un vecteur $\vec{r_i} = < r_{i_1}, \ldots, r_{i_{n-1}} > \in \Omega^{n-1}$ de ses accointances par ordre croissant de leur contribution. Soit $c_{i_k}(t)$ la contribution estimée du reducer r_{i_k} (c.à.d. la croyance qu'a i de la contribution du k^{ime} reducer de $\vec{r_i}$). Pour tout reducer $r_{i_k} \in \vec{r_i}$, nous définissons le delta de contribution comme :*

$$\Delta_i^k = c_i(t) - c_{i_k}(t)$$

D'après les conditions de divisibilité, si l'agent i est en situation de découper une tâche, alors il est en état de pause et il existe m reducers moins chargés que lui. Sa tâche la plus coûteuse τ est non négociable et on a : $c_\tau \geq \Delta_i^k, \forall k \in [1; m]$. Ainsi la découpe de la tâche τ, si on ne prend pas en compte les blocs de données des tâches, a pour objectif de déléguer k sous-tâches de même coût. Cette délégation doit permettre de diminuer le plus possible la contribution du reducer le plus chargé.

Le reducer i calcule k de la manière suivante :

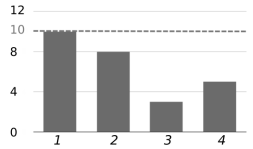
$$k = \underset{k\in[1;m]}{\operatorname{argmin}}(c_i(t) - \tfrac{k\Delta_i^k}{k+1})$$

Ce qui amène à la création de $k+1$ sous-tâches $\tau_1, \ldots, \tau_{k+1}$ telles que :

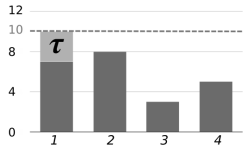
— $c_{\tau_1} = \ldots = c_{\tau_k} = \frac{\Delta_i^k}{k+1}$;

— $c_{\tau_{k+1}} = c_\tau - \frac{k\Delta_i^k}{k+1}$.

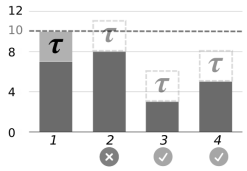
L'exemple suivant illustre comment l'indice k est choisi et l'impact qu'a ce choix sur les contributions du système après négociations.



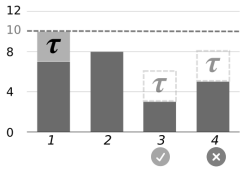
(a) Allocation initiale : le reducer 1 est le plus chargé



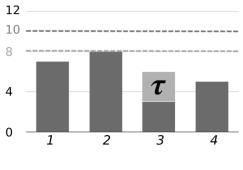
(b) Le reducer 1 propose la tâche τ



(c) Chaque enchérisseur applique le critère d'acceptabilité



(d) Le reducer 1 applique le critère de sélection



(e) Nouvelle allocation après la négociation

FIGURE 2 – Processus de négociation pas-à-pas : comment le reducer 1 délègue la tâche τ.

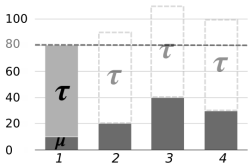


FIGURE 3 – Situation initiale, le reducer 1 ne peut pas négocier la tâche τ.

Exemple. Soit $\Omega = \{1, 2, 3, 4\}$, un ensemble de quatre reducers interconnectés avec les contributions respectives $c_1(t) = 80, c_2(t) = 20, c_3(t) = 40, c_4(t) = 30$. Le reducer 1 possède deux tâches : τ et μ. Puisqu'il est en train d'exécuter la tâche μ, le reducer 1 ne peut initier une enchère qu'avec la tâche τ qui n'est pas négociable (voir Figure 3). Le reducer 1 est donc en état de pause et il existe $m = 3$ reducers moins chargés que lui dans le système. Il peut découper la tâche τ afin de faire baisser sa contribution.

On observe :

— $\vec{r_1} =< 2, 4, 3 >$ (c.à.d. les accointances du reducer 1 dans l'ordre croissant de leur contribution)
— $\Delta_1^1 = c_1(t) - c_2(t) = 60$, $\Delta_1^2 = c_1(t) - c_4(t) = 50$, $\Delta_1^3 = c_1(t) - c_3(t) = 40$

Le nombre de sous-tâches et le nombre d'enchérisseurs pris en compte influencent les contributions résultantes à la division de tâche, et ce n'est pas toujours $k = m$ qui donne la contribution la plus faible pour le reducer le plus chargé. L'exemple que nous décrivons ici en est une illustration.

Si le reducer 1 partage τ avec un seul reducer ($k = 1$), il crée les sous-tâches afin d'équilibrer sa contribution avec celle du reducer le moins chargé. Ainsi, il va découper τ afin d'équilibrer sa contribution avec c_2 au terme d'une négociation. La meilleure découpe pour équilibrer c_1 et c_2 consiste à ne considérer que la partie de coût Δ_1^1 de τ et de la diviser en deux sous-tâches de même coût. Par conséquent, les sous-tâches τ_1 et τ_2 sont créées à partir de τ telles que :

— $c_{\tau_1} = \frac{\Delta_1^1}{2}$
— $c_{\tau_2} = c_\tau - c_{\tau_1} = c_\tau - \frac{\Delta_1^1}{2}$

De cette manière, le reducer 2 est en mesure d'accepter la tâche τ_1 amenant à une situation dans laquelle $c_1(t+1) = c_2(t+1) = 50$ (voir Figure 4).

En procédant au même raisonnement, les situations pour $k = 2$ (voir Figure 4) et pour $k = 3$ (voir Figure 4) donnent des nouvelles contributions différentes :

— pour $k = 2$, $c_1(t+1) = c_4(t+1) = 46$
— pour $k = 3$, $c_1(t+1) = c_3(t+1) = 50$

Plus généralement, après une négociation avec k enchérisseurs, le reducer 1 délègue k sous-tâches de coût $\frac{\Delta_1^k}{k+1}$ et sa nouvelle contribution

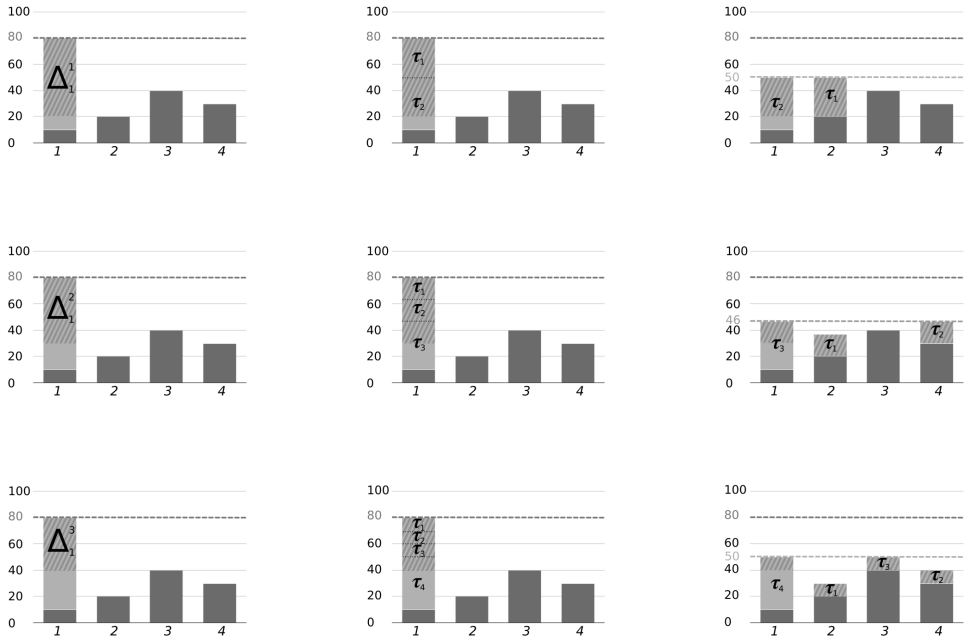


FIGURE 4 – Découpe de la tâche τ entre les reducers 1 et 2 (haut), entre les reducers 1, 2 et 4 (centre) et entre les reducers 1, 2, 3 et 4 (bas).

vaut $c_1(t+1) = c_1(t) - \frac{k\Delta_1^k}{k+1}$. Comme on peut le constater, il existe une valeur de k (ici $k = 2$) qui minimise c_1' :

$$k = \underset{k\in[1;3]}{\operatorname{argmin}}(c_1(t) - \tfrac{k\Delta_1^k}{k+1}).$$

3.4 Découpe effective de tâche

La section précédente présente une création *idéale* des sous-tâches si l'on considère des tâches complètement divisibles. Cependant, les tâches de reduce sont composées de blocs de données et ne peuvent pas être divisées avec une telle précision. Cette section présente la découpe effective de tâche qui repose sur l'heuristique ci-dessus.

Les tâches de reduce sont composées de plusieurs blocs de données dont il faut tenir compte lors de la découpe de celles-ci. Un bloc de données est dit complet s'il n'est pas possible de lui ajouter des données, il est dit partiel sinon. Soient $\mathcal{B}$ la taille d'un bloc et $c_\mathcal{B}$ la contribution des données d'un bloc complet. Les blocs d'une tâche sont initialement créés par les mappers qui produisent pour chaque clé qu'ils traitent, selon leur nombre de valeurs associées, des blocs complets et au plus un bloc partiel (de contribution inférieure à $c_\mathcal{B}$). Une tâche reduce, dont x mappers ont traité la clé, contient donc un certain nombre de blocs complets (la somme des blocs complets construits par les x mappers) et au plus x blocs partiels.

Afin de créer des sous-tâches composées d'au moins un bloc de données, le calcul de k pour un reducer i doit être revu de la façon suivante :

$$k = \underset{k\in[1;N]}{\operatorname{argmin}}(c_i(t) - \tfrac{k\Delta_i^k}{k+1})$$

avec $N = max(\{n \in [1;m] \mid \Delta_i^n \geq c_\mathcal{B}\})$.

Pour le reducer i découper une tâche τ consiste

(i) à déterminer k en utilisant la formule ci-dessus, k étant le nombre de reducers à qui l'on souhaite déléguer une sous-tâche ;

(ii) à diviser les blocs de données de τ pour construire $k+1$ sous-tâches en répétant la routine suivante : parmi les blocs de données non attribués de τ, le bloc le plus coûteux est attribué à la sous-tâche de plus faible coût.

Les $k+1$ sous-tâches ainsi construites sont proches de leur coût idéal (c.à.d. $\frac{\Delta_i^k}{k+1}$ pour k

d'entre elles et $c_\tau - \frac{k\Delta_i^k}{k+1}$ pour la dernière). En fait, pour une sous-tâche τ' de coût idéal $c^*_{\tau'}$ et de coût réel $c_{\tau'}$, on observe que $|c^*_{\tau'} - c_{\tau'}| \leq c_{\mathcal{B}}$.

3.5 Agrégation des résultats intermédiaires

Comme expliqué dans la section 2, le découpage des tâches se base sur la définition de deux fonctions pour la phase de reduce : une fonction de reduce intermédiaire appelée IR et une fonction de reduce final appelée FR. Lorsqu'une tâche de reduce n'est pas découpée, on lui applique directement la fonction FR. Lorsqu'une tâche de reduce est découpée, ses sous-tâches sont marquées comme tâches de reduce intermédiaire, et on doit leur appliquer la fonction IR. Ces sous-tâches sont considérées comme n'importe quelle autre tâche et sont donc candidates à la négociation ainsi qu'au découpage. Les résultats des différentes sous-tâches doivent cependant être regroupés pour permettre d'établir le résultat de la tâche initiale. C'est le reducer initiateur de la première découpe qui a cette responsabilité. Cette information est portée par les différentes sous-tâches.

Considérons par exemple une tâche τ affectée au reducer i. Ce reducer décide de découper τ en $\{\tau_1, \tau_2, \tau_3\}$, il traite lui-même τ_1 avec la fonction IR, et délègue τ_2 et τ_3 aux reducers j et l. Le reducer j décide de découper à son tour τ_2 en τ_{21} et τ_{22}, afin de déléguer τ_{22} à un quatrième reducer. Les résultats de l'application de IR sur toutes ces tâches $\{\tau_1, \tau_{21}, \tau_{22}, \tau_3\}$ sont envoyés à i, le reducer qui a initié la division de la tâche de reduce τ qui leur applique la fonction de reduce final FR.

Les résultats intermédiaires sont bien amenés à un même reducer final, quelque soit le nombre de découpes intermédiaires, ce qui préserve la cohérence du résultat final.

4 Expérimentations

Nos expérimentations ont pour objectifs de comparer notre proposition avec la distribution classique du MapReduce et d'évaluer la valeur ajoutée du découpage des tâches, c'est-à-dire de comparer notre SMA à celui proposé dans [1].

Nous avons implémenté notre prototype avec le langage de programmation Scala [1] et la boîte à outils Akka [2]. Cette dernière, en s'appuyant sur le modèle d'acteur [8], nous permet de réduire la distance entre les spécifications du SMA et son implémentation. De plus, grâce à Akka, le déploiement du SMA sur un cluster de PC est simple et direct.

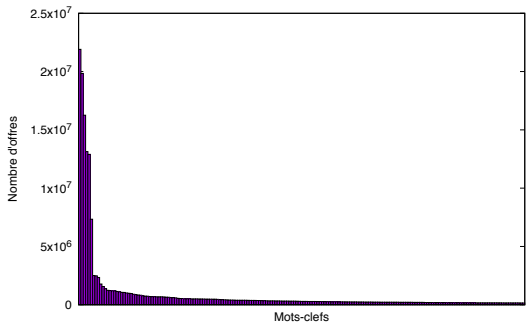


FIGURE 5 – Nombres d'offres par mot-clé.

Yahoo ! utilise un mécanisme d'enchères pour vendre des espaces publicitaires au sein de son moteur de recherche. Par exemple, une agence de voyage veut apparaître aux côtés des résultats de recherche « voyage à Caen ». L'offre de cet annonceur correspond au prix qu'il est prêt à payer chaque fois qu'un internaute clique effectivement sur son annonce. Nous avons analysé le jeu de données correspondant à la période allant du 15 juin 2002 au 14 juin 2003 qui contient $7.7\ 10^7$ offres (journée, identifiant de l'annonceur, liste de mots-clés, etc.), soit environ 8 Go [3]. Attendu que chaque offre porte sur une liste de mots-clés, nous considérons le job qui consiste à comptabiliser le nombre d'offres pour chacun des mots-clés. Comme illustré dans la figure 5, 6 mots-clés parmi 75 359 font l'objet d'un grand nombre d'offres. Ainsi, ce job doit exécuter 6 tâches particulièrement coûteuses.

Nous faisons l'hypothèse que l'équilibre des charges parmi les reducers et a fortiori le découpage des tâches permet de réduire le temps d'exécution de la phase de reduce. À cette fin, nous comparons le temps d'exécution de ces phases dans la distribution classique du MapReduce, celle du SMA proposé dans [1] et celle de notre SMA qui découpe les tâches. Nous avons exécuté le job présenté précédemment avec 10 mappers et 10 reducers. Nous présentons les durées d'exécution en fonction du nombre de machines utilisées, des PCs 3.30GHz Intel(R) Core(TM) i5 avec 4 cœurs et 8 Go de RAM. Pour chaque ensemble de paramètres, nous effectuons 3 exécutions. Comme l'écart-type dû au non-déterminisme de l'ordonnanceur est faible, nous n'exhibons que les moyennes sur les différentes exécutions.

1. http ://www.scala-lang.org/
2. http ://akka.io
3. `http://webscope.sandbox.yahoo.com/`

La figure 6 présente les temps d'exécutions des différentes phases. La figure 7 exhibe l'équité, c.à.d. le rapport entre le temps d'exécution du reducer le moins sollicité et de celui le plus sollicité. Comme on peut le constater le temps d'exécution de la phase de map décroît avec le nombre de machines car elle bénéficie du parallélisme. Quelque soit la méthode utilisée, le temps d'exécution de la phase de reduce décroît globalement. Cette décroissance n'est pas parfaitement proportionnelle au nombre de machines car la phase de reduce est pénalisée par la non-localité des données : un reducer peut avoir besoin de traiter des données issues d'un mapper qui se situe sur une autre machine. De plus, l'approche classique est pénalisée par les biais dus à la forme des données : ne s'adaptant pas à la présence des clés coûteuses, cette approche ne permet pas de réduire l'écart entre l'effort réalisé par le reducer le plus sollicité et ceux qui le sont moins. Selon cette approche, l'équité reste faible comme le montre la figure 7 : le reducer le moins sollicité travaille environ 50% moins que celui qui l'est le plus.

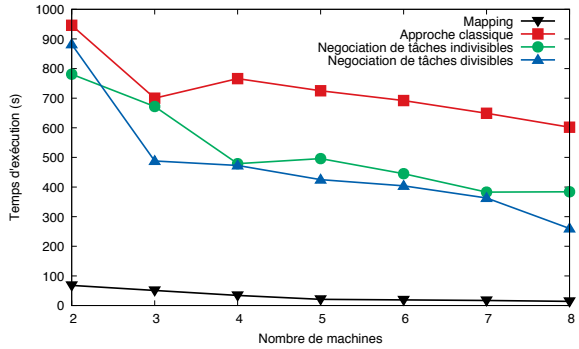


FIGURE 6 – Temps d'exécution des différentes phases.

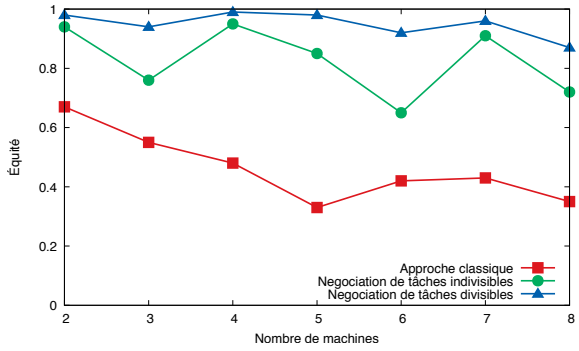


FIGURE 7 – Équité entre reducers.

À l'inverse, les approches basées sur la négociation terminent plus rapidement. Ce gain de temps s'explique par une meilleure exploitation de l'ensemble des ressources disponibles, i.e. une équité proche de 1 qui traduit le fait que le travail lié au job est réparti uniformément sur tous les reducers. En effet, la négociation permet de répartir dynamiquement et en continu les tâches vers les agents les moins chargés. Le jeu de données présentant 6 clés particulièrement coûteuses, dès que l'on atteint ce nombre de machines, l'intérêt de la découpe des tâches est réduit, d'autant que l'équité est déjà proche de 1. La négociation attribue rapidement au plus une clé coûteuse par reducer équilibrant ainsi naturellement les charges de travail même sans avoir recours à la division des tâches. Cependant notre SMA exploitant la division reste plus performant que celui basé uniquement sur la négociation, même si ce gain est parfois faible.

Ces expériences montrent que notre SMA bénéficie du parallélisme plus que les autres implémentations. En résumé, la division combinée à l'équilibrage par la négociation des tâches permet de réduire le temps consacré à la phase de reduce en améliorant l'équité de la charge.

5 Conclusion

Dans cet article nous montrons comment la mise en œuvre du patron de conception MapReduce à l'aide d'un SMA permet de contrer les biais de données pénalisant la phase de reduce, en particulier le biais des clés coûteuses. Notre système ne demande ni de pré-traitement ni de paramétrage dépendant des données. Notre implémentation de MapReduce à l'aide de SMA s'appuie sur des agents reducers qui découpent et négocient leurs tâches au cours de la phase de reduce. Leurs décisions se basent sur la quantité de données restant à traiter par chacun (i.e. la contribution). Cette prise de décision est locale, et ne nécessite pas de centralisation de l'information. Notons que si l'environnement d'exécution est hétérogène, le système s'adapte automatiquement. Nos expérimentations montrent que la division et l'équilibrage des tâches permettent de réduire le temps consacré à la phase de reduce grâce à une réduction des écarts d'effort entre les reducers.

Ces expériences nous amènent à envisager plusieurs perspectives. D'une part, nous souhaitons améliorer le comportement des enchérisseurs pour qu'ils participent simultanément à plusieurs enchères afin de réduire le taux d'échec (actuellement 50 %) et donc réduire le coût communicationnel. D'autre part, nous devons

intégrer un critère de localité des données dans la prise de décision lors des négociations et ainsi réduire le coût de transfert des données. A cette intention, nous envisageons de nous abstraite de notre problématique applicative pour considérer le problème générale de ré-allocation dynamique de tâches entre machines hétérogènes.

Remerciements

Ce travail est soutenu par le défi CNRS MASTODONS. Nous remercions le comité de programme des JFSMA qui, par ses remarques, nous a permis d'améliorer cet article.

Références

[1] Baert, Q., Caron, A.-C., Morge, M., and Routier, J.-C. (2016). Allocation équitable de tâches pour l'analyse de données massives. In *JFSMA*, pages 55–64. Cépaudès éditions.

[2] Chen, Q., Zhang, D., Guo, M., Deng, Q., and Guo, S. (2010). SAMR : A self-adaptive MapReduce scheduling algorithm in heterogeneous environment. In *ICCIT*, pages 2736–2743. IEEE.

[3] Clair, G., Gleizes, M.-P., Kaddoum, E., and Picard, G. (2008). Approches multi-agents auto-organisatrices pour un contrôle manufacturier intelligent et adaptatif. In *JFSMA*, pages 191–200. Cépaduès.

[4] Dean, J. and Ghemawat, S. (2004). MapReduce : Simplified data processing on large clusters. In *SOSDI*, pages 137–150.

[5] Essa, Y. M., Attiya, G., and El-Sayed, A. (2014). Mobile agent based new framework for improving big data analysis. *IJACSA*, 5(3) :25–32.

[6] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., and Pirahesh, H. (1997). Data cube : A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *Data Mining and Knowledge Discovery*, 1(1) :29–53.

[7] Gufler, B., Augsten, N., Reiser, A., and Kemper, A. (2011). Handling data skew in mapreduce. In *ICCCSS*, pages 574–583.

[8] Hewitt, C., Bishop, P., and Steiger, R. (1973). A universal modular actor formalism for artificial intelligence. In *IJCAI*, pages 235–245.

[9] Kwon, Y., Balazinska, M., Howe, B., and Rolia, J. (2012). Skewtune : Mitigating skew in mapreduce applications. In *ACM SIGMOD ICMD*, pages 25–36.

[10] Lama, P. and Zhou, X. (2012). Aroma : Automated resource allocation and configuration of mapreduce environment in the cloud. In *ICAC*, pages 63–72.

[11] Li, W. (1992). Random texts exhibit zipf's-law-like word frequency distribution. *IEEE Transactions on Information Theory*, 38(6) :1842–1845.

[12] Lin, J. (2009). The curse of zipf and limits to parallelization : A look at the stragglers problem in mapreduce. In *Workshop on Large-Scale Distributed Systems for Information Retrieval*, page 2009.

[13] Liroz-Gistau, M., Akbarinia, R., and Valduriez, P. (2015). FP-Hadoop : efficient execution of parallel jobs over skewed data. *VLDB Endowment*, 8(12) :1856–1859.

[14] Mathieu, P., Routier, J.-C., and Secq, Y. (2002). Principles for dynamic multi-agent organizations. In *PRIMA*, pages 109–122.

[15] Nongaillard, A. and Mathieu, P. (2011). Egalitarian negotiations in agent societies. *AAI*, 25(9) :799–821.

[16] Nongaillard, A., Mathieu, P., and Jaumard, B. (2008). La négociation du bien-être social utilitaire. In *JFSMA*, pages 55–64.

[17] Verma, A., Cherkasova, L., and Campbell, R. (2011). Aria : Automatic resource inference and allocation for mapreduce environments. In *ICAC*, pages 235–244.

[18] Wagner, I., M., L., and Bruckstein, A. M. (1999). Distributed covering by ant-robots using evaporating traces. *IEEE Trans. Robotics and Automation*, 15(5) :918–933.

Cognition, émotions et relations sociales pour la simulation multi-agent

M. Bourgais[a,b] mathieu.bourgais@insa-rouen.fr
P. Taillandier[c] patrick.taillandier@gmail.com
L. Vercouter[a] laurent.vercouter@insa-rouen.fr

[a]Normandie Univ, INSA Rouen, UNIHAVRE, UNIROUEN, LITIS
76000 Rouen, France

[b]Normandie Univ, UNICAEN, UNIHAVRE, UNIROUEN, CNRS, IDEES
76000 Rouen, France

[c]MIAT, INRA, 31000 Toulouse, France

Résumé

Les simulations sociales ont besoin d'agents au comportement réaliste pour être utilisées comme outil scientifique par les sciences sociales. La simulation d'une société humaine avec un comportement crédible implique l'utilisation de cognition, de liens sociaux entre les personnes ainsi que la prise en compte des émotions et des dynamiques entre ces composants. Cependant, développer un tel comportement est souvent trop complexe pour des chercheurs ayant peu de connaissances en programmation. Dans cet article, nous présentons un formalisme qui a pour but de représenter la cognition, les relations sociales et les émotions et qui est intégré dans une architecture d'agent pour donner un comportement émotionnel dynamique à des agents sociaux. Cette architecture est implémentée dans la plateforme multi-agent open-source GAMA. Un cas d'étude d'évacuation lors d'un incendie de brousse en Australie est utilisé pour montrer le potentiel de nos travaux.

Mots-clés : *Simulations sociales, Émotions, Cognition, Architecture agent*

Abstract

Social Simulations need agents with a realistic behavior to be used as a scientific tool by social scientists. When simulating a human society, a realistic behavior implies the use of cognition, social relations between people but also to take into account emotions and the dynamic between these features. However, developing such a behavior is often too complex for people with little knowledge in programming. In this paper, we present a formalism to represent cognition, social relations and emotions, which is integrated in an agent architecture to give a dynamic emotional behavior to social agents. This architecture is implemented in the open-source multi-agent platform GAMA. A use case about evacuation during bush fires in Australia is used to show the possibilities of our work.

Keywords: *Social Simulation, Emotions, Cognition, Agent Architecture*

1 Introduction

La simulation multi-agent est devenu un outil important, spécialement en sciences sociales où elle est utilisée pour étudier des systèmes complexes composés de centaines ou de milliers d'humains simulés. Pour augmenter la précision des résultats obtenus avec ces simulations sociales, il est important qu'elle soit aussi proche que possible des cas réels étudiés. Cette recherche du réalisme conduit à l'utilisation d'agent sociaux crédibles, c'est à dire des agents possédant de plus en plus de composantes sociales [35].

Prenons l'exemple des feux de brousses en Australie étudié grâce aux simulations multi-agents [2]. Dans cette étude, le but était de simuler l'évacuation d'une zone ouverte dans le contexte d'un incendie de brousse en Australie. Les auteurs ont utilisé une architecture BDI [8] pour la modélisation de la cognition des agents mais dans cette situation, une personne réagit aussi en fonction de ses émotions ainsi que de ses relations sociales.

Dans cet article, nous nous intéressons à ce problème en intégrant à une architecture d'agent cognitive la possibilité de définir des émotions et des relations sociales dynamiques. Le but principal est de fournir un formalisme pour la création d'émotions par la cognition ainsi que pour l'évolution des relations sociales en fonction des états mentaux d'un agent. Ce formalisme est ensuite intégré à une architecture d'agent. La facilité d'utilisation du mo-

dèle proposé est une considération majeure du travail décrit ici afin qu'il puisse être manipulé par des chercheurs venant des sciences sociales qui n'ont pas nécessairement des compétences avancées en programmation. Notre travail a été implémenté dans la plateforme opensource GAMA [14]. Le principal avantage de cette intégration est de bénéficier du langage de modélisation offert par GAMA qui a pour but de faciliter l'utilisation de la plateforme par des modélisateurs non expert en programmation.

Cet article est structuré comme suit. En section 2, nous discutons des travaux existants pour créer des agents sociaux possédant une cognition, des émotions ou des relations sociales. En Section 3, nous proposons un formalisme utilisé pour organiser les états mentaux de l'agent en termes de cognition, d'émotions et de relations sociales. L'architecture développée à partir du formalisme proposée est présentée en section 4. Dans la section 5, nous présentons un exemple pour illustrer la façon dont notre architecture peut être utilisée sur un modèle d'évacuation. Finalement, la Section 6 sert de conclusion.

2 État de l'art

Créer des agents vraisemblables est un point crucial dans le domaine des simulations sociales. Dans cette section, nous présentons différents travaux traitant de l'intégration de la cognition, des émotions ou des relations sociales dans des agents pour améliorer le réalisme de simulations sociales.

2.1 Des agents sociaux cognitifs

Définir le comportement d'agents avec des notions relatives à la cognition humaine est une première étape dans l'optique d'augmenter le réalisme des simulations sociales [3]. Pour cela, des architectures cognitives ont été proposées parmi lesquelles l'architecture BDI [8] qui est la mieux adaptée au contexte de la simulation [3]. Le paradigme BDI utilise la logique modale [9] pour définir les concepts de croyances (Beliefs en anglais), désires (Desires en anglais) et d'intentions (Intentions en anglais) qui composent les états mentaux de l'agent. Ce paradigme fourni aussi des liens logiques entre ces concepts ainsi qu'une base de plans d'actions permettant de donner un comportement cognitif aux agents.

Pour faciliter son utilisation, l'architecture BDI a été implémentée dans différents frameworks. Un framework classique est le Procedural Reasoning Systems (PRS) [21] qui est basé sur trois étapes : une perception de l'environnement pour mettre à jour la base des croyances, une délibération entre les désirs et l'état du monde et finalement la sélection d'une action à exécuter. PRS sert de base à d'autres framework comme JACK [15] ou Jadex [26].

Certains chercheurs ont tenté d'intégrer l'architecture BDI à des plateformes de modélisation et de simulation. Une extension à NetLogo [39] implémente une version simplifiée de l'architecture BDI à des fins éducatives [30]. Sing et Padgham [32] ont décidé de connecter une plateforme multi-agent avec un framework BDI existant (Jack ou Jadex par exemple) et, dans le même esprit, une application connectant la plateforme Matsim [4] avec le framework GORITE BDI [28] a été proposée.

2.2 Architectures émotionnelles

Différents travaux ont montré que l'ajout d'émotions dans des agents augmente la crédibilité de leur comportement [5] [22]. Cette amélioration de vraisemblance est utile dans le cas de simulations sociales dont le but est de produire des simulations aussi réalistes que possible.

En psychologie, il n'y a pas de consensus sur une unique théorie émotionnelle. La théorie la plus utilisée en intelligence artificielle est la théorie de l'évaluation cognitive des émotions [31] [33] et plus particulièrement la théorie OCC [25] qui a été spécialement développée pour intégrer des émotions en intelligence artificielle.

Le modèle OCC des émotions définit vingt-deux émotions distribuées en onze paires en fonction de l'appréciation cognitive d'une situation par un agent. Cette appréciation cognitive est réalisée selon trois aspects : les conséquences des évènements, les actions des autres agents et l'aspect des objets.

Différentes implémentations de systèmes émotionnels pour des simulations multi-agent ont été proposées. Par exemple, DETT (Disposition, Emotion, Trigger, Tendency) [38] considère la perception d'une situation comme la condition de déclenchement du mécanisme de création d'émotions suivant la théorie OCC. Gratch et Marsella proposent une approche différente avec leur modèle EMA [13] qui prend en compte non seulement la création des émotions

en se basant sur l'appréciation d'une situation par le biais de variables d'appréciations mais aussi le comportement induit par l'émotion sur la cognition de l'agent. Finalement, eBDI [17] propose d'intégrer directement le modèle OCC dans une architecture BDI.

2.3 Des relations sociales en simulation multi-agent

Puisque les personnes créent des relations sociales en vivant avec d'autres personnes, il parait logique de modéliser des relations sociales entre des agents simulant des humains. Dans [29], le comportement des agents est calculé à partir d'un modèle socio-physiologique comprenant une personnalité, des émotions et une attitude de l'agent envers son environnement. Dans un autre cas, Gratch [12] ajoute un niveau social à son architecture pour modifier le comportement d'un agent en fonction de l'état social du monde.

Dans ces modèles d'agents sociaux, les relations sociales sont représentées avec un nombre fini de variables, chacune définissant une dimension précise de la relation. Ces variables correspondent à celles présentes dans le modèle dimension des relations interpersonnelles de Svennevig [36] :

- Le degré d'**appréciation** envers un autre agent [16].
- Le degré de **dominance** qu'un agent a sur un autre [29]. Cela représente le niveau de puissance qu'un agent pense avoir sur un autre.
- Le degré de **solidarité** aussi connu comme la distance sociale [6]. Il indique les similarités en terme de désirs, croyances et valeurs entre deux agents.
- Le degré de **familiarité** qui caractérise le nombre et le type (privé ou public) d'informations qui peuvent être transmises à un autre agent [6].

Dans, les travaux cités précédemment les relations sociales sont étudiées comme des facteurs de changement comportemental, ils sont donc définis comme des variables statiques. Pourtant, de façon évidente, une relation sociale entre deux personnes peut évoluer dans le temps. Pour répondre à ce problème, Ochs [23] a proposé une architecture agent incluant la personnalité et utilisant les émotions pour donner une dynamique aux relations sociales de personnages non jouables dans des jeux vidéo.

2.4 Synthèse

Dans le contexte de la simulation sociale, qui pose des problème de de passage à l'échelle et de simplicité de prise en main, ces trois notions, la cognition, les émotions et les relations sociales, non jamais été combinées pour fournir un comportement réaliste à des humains simulés. De plus, chacun de ces concepts n'a jamais été implémenté de sorte qu'il soit simple à utiliser pour des scientifiques venant des sciences sociales qui n'ont pas un niveau élevé en programmation.

Cet article a pour but de proposer une architecture pour les simulations sociales incluant la cognition, les émotions et les relations sociales et pouvant être utilisée par des modélisateurs ne possédant qu'un niveau basique en programmation. Cette architecture est développée de façon générique puis implémentée dans la plateforme de modélisation et de simulation multi-agent GAMA [14] qui a prouvé sa facilité de prise en mais [20] [27] grâce à son langage de modélisation, le GAML, que nous avons étendu pour pouvoir utiliser notre architecture.

3 Création d'émotions et de relations sociales par la cognition

La principale contribution de cet article consiste à définir un formalisme pour représenter et pour articuler les états mentaux d'un agent social. Ces états mentaux sont composés d'un état cognitif, d'un état émotionnel et de relations sociales avec les autres agents.

3.1 Représenter les états mentaux des agents

Représenter la cognition avec des prédicats. La partie cognitive de notre architecture est basée sur le paradigme BDI [8], dans lequel les agents possèdent une base des croyances, une base des désirs ainsi qu'une base des intentions pour stocker les états cognitifs à propos du monde. Nous utilisons aussi une base des croyances incertaines attendues, appelée base des incertitudes, qui est utilisée lors de la création d'émotions à propos de faits attendus sans certitude.

Pour représenter cette connaissance, nous utilisons des prédicats. Un prédicat unifie la représentation des informations sur le monde ce qui signifie que cela peut représenter une situation,

un évènement ou une action. Puisque le but de ce travail est de créer des émotions à partir de la cognition sur des évènements ou sur la valeur des actions des autres agents, nous représentons une information P causée par un agent j avec une valeur d'appréciation morale pr par $\mathbf{P}_{j,pr}$. La valeur d'appréciation morale peut être positive (dans ce cas, l'information P est louable) ou négative (dans ce cas, l'information P est blâmable). Un prédicat $\mathbf{P}_j$ représente une information causée par un agent j avec n'importe quelle valeur d'appréciation morale alors qu'un prédicat **P** représente une information causée par n'importe quel agent avec n'importe quelle valeur d'appréciation morale. Nous représentons l'opposé d'un prédicat P par **non P**.

En fonction de la base dans laquelle il est stocké, un prédicat peut être considéré comme une croyance, une croyance incertaine ou un désir et cela est représenté de la façon suivante :

- **Croyance$_i$(P)** : indique que le prédicat P appartient à la base des croyances de l'agent i.
- **Attendu$_i$(P)** : indique que le prédicat P appartient à la base des incertitudes de l'agent i.
- **Desir$_i$(P)** : indique que le prédicat P appartient à la base des désirs de l'agent i.

Représentation formelle des émotions. Pour la définition des émotions, nous basons notre travail sur la théorie des émotions OCC [25]. Selon cette théorie, une émotion est une réponse valuée à l'appréciation d"une situation. Comme nous utilisons les émotions pour mettre à jour dynamiquement les relations sociales entre agent, notre définition des émotions a aussi besoin de contenir l'agent qui a causé l'émotion. Avec cette définition, nous représentons une émotion par **E$_i$(P,A,I,D)** avec les éléments suivant :

- $\mathbf{E}_i$: le nom de l'émotion ressentie par l'agent i.
- **P** : le prédicat représentant le fait à propos duquel l'émotion est ressentie.
- **A** : l'agent causant l'émotion.
- **I** : l'intensité de l'émotion.
- **D** : la valeur de décroissance de l'intensité de l'émotion dans le temps.

Par exemple, si un agent Alice ressent de la peur à propos d'une action P causée par l'agent Bob avec une intensité de 4.5 et une décroissance de 0.6, cela sera représenté par l'émotion Peur$_{Alice}$(P$_{Bob}$,Bob,4.5,0.6). Une émotion sans intensité spécifique ou sans décroissance est représentée par **E$_i$(P,A)** et une émotion ne possédant qu'un fait comme cause est représentée par **E$_i$(P)**.

Formalisation des relations sociales. En se basant sur le travail de Svennevig [36] exposé en section 2, nous définissons un lien social avec un autre agent par le tuple <agent, appréciation, dominance, solidarité, familiarité> avec les éléments suivants :

- **Agent** : l'agent concerné par le lien, identifié par son nom.
- **Appréciation** : une valeur réelle comprise entre -1 et 1 représentant le degré d'appréciation avec l'agent concerné par le lien. Une valeur de -1 indique que l'agent concerné est détesté, une valeur de 1 indique que l'agent est adoré.
- **Dominance** : une valeur réelle comprise entre -1 et 1 représentant le degré de pouvoir exercé sur l'agent concerné par le lien. Une valeur de -1 indique que l'agent concerné est dominant, une valeur de 1 indique l'agent concerné est dominé.
- **Solidarité** : une valeur réelle comprise entre 0 et 1 représentant le degré de solidarité avec l'agent concerné par le lien. Une valeur de 0 indique qu'il n'y a aucune solidarité avec l'agent concerné, une valeur de 1 indique une totale solidarité avec l'agent concerné.
- **Familiarité** : une valeur réelle comprise entre 0 et 1 représentant le degré de familiarité avec l'agent concerné par le lien. Une valeur de 0 indique qu'il n'y a aucune familiarité, une valeur de 1 indique une totale familiarité avec l'agent concerné.

3.2 Créer du dynamisme dans les émotions et les relations sociales

Création dynamique d'émotions. Nous basons la création automatique des émotions par rapport aux états mentaux de l'agent sur le modèle OCC [25] et son formalisme logique [1] qui a été proposé pour intégrer le modèle OCC dans une architecture BDI.

Selon la théorie OCC, les émotions peuvent être divisées en trois groupes : les émotions liées aux évènements, les émotions liées aux personnes et aux actions commises par ces personnes et enfin les émotions liées aux objets. Dans ce travail,

comme nous nous focalisons sur les relations entre des agents sociaux, nous ne travaillons que sur les deux premiers groupes d'émotions (les émotions liées aux situations et les émotions liées aux personnes et à leurs actions) ce qui signifie que nous laissons de coté les émotions liées aux objets.

Les vingt émotions définies dans cet article peuvent être divisées en trois parties : huit émotions liées aux évènements, quatre émotions en lien avec les autres agents et huit émotions liées aux actions des agents.

Les huit émotions en lien avec les évènements ont la définition suivante :

- **Joie**$_i$**(P**$_j$**,j)** = Croyance$_i$(P$_j$) & Desir$_i$(P)
- **Tristesse**$_i$**(P**$_j$**,j)** = Croyance$_i$(P$_j$) & Desir$_i$(non P)
- **Espoir**$_i$**(P**$_j$**,j)** = Attendu$_i$(P$_j$) & Desir$_i$(P)
- **Peur**$_i$**(P**$_j$**,j)** = Attendu$_i$(P$_j$) & Desir$_i$(non P)
- **Satisfaction**$_i$**(P**$_j$**,j)** = Espoir$_i$(P$_j$,j) & Croyance$_i$(P$_j$)
- **Déception**$_i$**(P**$_j$**,j)** = Espoir$_i$(P$_j$,j) & Croyance$_i$(non P$_j$)
- **Soulagement**$_i$**(P**$_j$**,j)** = Peur$_i$(P$_j$,j) & Croyance$_i$(non P$_j$)
- **Peur Confirmée**$_i$**(P**$_j$**,j)** = Peur$_i$(P$_j$,j) & Croyance$_i$(P$_j$)

En plus de ces définitions, selon le formalisme logique [1], quatre règles peuvent être définies :

- La création de **peur confirmée** ou la création de **soulagement** remplace l'émotion de **peur**.
- La création de **satisfaction** ou la création de **déception** remplace l'émotion d'**espoir**.
- La création de **satisfaction** ou de **soulagement** mène à la création de **joie**.
- La création de **déception** ou de **peur confirmée** mène à la création de **tristesse**.

Les quatre émotions liées aux autres agents ont la définition suivante :

- **Content pour**$_i$**(P,j)** = i apprécie j & Joie$_j$(P)
- **Désolé pour**$_i$**(P,j)** = i apprécie j & Tristesse$_j$(P)
- **Resentiment**$_i$**(P,j)** = i n'apprécie pas j & Joie$_j$(P)
- **Jubilation**$_i$**(P,j)** = i n'apprécie pas j & Tristesse$_j$(P)

Les termes "i apprécie j" et "i n'apprécie pas j" ont la définition suivante :

- **i apprécie j** : l'agent i possède une relation sociale avec l'agent j avec une valeur d'appréciation positive.
- **i n'apprécie pas j** : l'agent i possède une relation sociale avec l'agent j avec une valeur d'appréciation négative.

Finalement, les huit émotions liées aux actions réalisées par des agents ont la définition suivante :

- **Fierté**$_i$**(P**$_i$**,i)** = Croyance$_i$(P$_i$) & P$_i$ louable
- **Honte**$_i$**(P**$_i$**,i)** = Croyance$_i$(P$_i$) & P$_i$ blâmable
- **Admiration**$_i$**(P**$_j$**,j)** = Croyance$_i$(P$_j$) & P$_j$ louable
- **Reproche**$_i$**(P**$_j$**,j)** = Croyance$_i$(P$_j$) & P$_j$ blâmable
- **Gratification**$_i$**(P**$_i$**,i)** = Fierté$_i$(P$_i$,i) & Joie$_i$(P$_i$)
- **Remords**$_i$**(P**$_i$**,i)** = Honte$_i$(P$_i$,i) & Tristesse$_i$(P$_i$)
- **Gratitude**$_i$**(P**$_j$**,j)** = Admiration$_i$(P$_j$,j) & Joie$_i$(P$_j$)
- **Colère**$_i$**(P**$_j$**,j)** = Reproche$_i$(P$_j$,j) & Tristesse$_i$(P$_i$)

Les termes "louable" et "blâmable" ont la définition suivante :

- **louable** : indique que le fait P a une valeur d'appréciation morale positive.
- **blâmable** : indique que le fait P a une valeur d'appréciation morale négative.

Mettre à jour automatiquement les relations sociales. Comme expliqué dans la section 2, certains travaux ont montré que les relations sociales sont faites pour être dynamiques. En se basant sur les précédents travaux de Ochs [23], nous intégrons dans notre architecture un moteur social qui met à jour les liens sociaux d'un agent en fonction de sa cognition et de son état émotionnel.

Par la suite, nous étudions la mise à jour du lien social <j, Appréciation, Dominance, Solidarité, Familiarité> possédé par l'agent i. Chaque variable de ce lien social évolue selon sa propre règle.

- **Appréciation** : selon Ortony [24], le degré d'appréciation entre deux agents dépend de la valence (positive ou négative) de l'émotion causée par l'agent concerné

par le lien. Dans notre modèle, *joie* et *espoir* sont considérées comme des émotions positives (*satisfaction* et *soulagement* créent automatiquement de la *joie* dans notre moteur) tandis que *tristesse* et *peur* sont considérées comme des émotions négatives (*peur confirmée* et *déception* créent automatiquement de la *tristesse* dans notre moteur). Cette évolution est réalisée par un niveau fixe pour chaque émotion positive ou négative impliquée. Ainsi, cette évolution ne dépend pas de l'intensité de l'émotion concernée puisque notre moteur crée des émotions sans intensité.
De plus, certains travaux ont montré que le degré d'appréciation est influencé par la valeur de solidarité [34]. La règle de calcul de mise à jour de l'appréciation peut être formalisée comme suit avec $nbPE(t)$ le nombre d'émotions positives causées par l'agent j à l'agent i au temps t, $nbNE(t)$ le nombre d'émotions négatives causées par l'agent j à l'agent i au temps t et α le coefficient d'évolution compris entre 0 et 1 :

$$appreciation(t+1) = appreciation(t) * \\ * (1 + solidarite(t)) + \\ + \alpha * (nbPE(t+1) - nbNE(t+1))$$

— **Dominance** : Keltner et Haid [18] expliquent qu'une émotion de peur ou de tristesse causée par un autre agent représente un statut inférieur. Mais Knutson [19] explique que percevoir de la peur ou de la tristesse chez les autres augmente le sentiment de pouvoir sur ces personnes. La règle de calcul peut être formalisée comme suit avec $nbONE(t)$ le nombre d'émotions négatives causées par l'agent i sur l'agent j au temps t et $nbSNE(t)$ le nombre d'émotions négatives causées par l'agent j à l'agent i au temps t :

$$dominance(t+1) = dominance(t) + \\ + \alpha * (nbONE(t+1) - nbSNE(t+1))$$

— **Solidarité** : comme expliqué en section 2, la solidarité représente le degré de similarité des désirs, croyances et attentes entre deux agents. Dans notre travail, l'évolution de la valeur de solidarité dépend du ratio de similarité en terme de désirs, croyances et attentes entre l'agent i et l'agent j. Nous comparons les bases de désirs, de croyances et d'incertitudes des deux agents et nous cherchons les similarités (lorsque les prédicats sont égaux) et les différences (lorsque les prédicats sont de valeur de vérité opposée mais égaux surle reste). De plus selon de Riviera et Grinkis [11], les émotions négatives ont tendance à réduire la valeur de solidarité entre deux personnes. La règle de calcul peut être formalisée comme suit avec $nbS(t)$ le nombre de similarités au temps t et $nbD(t)$ le nombre de différences au temps t :

$$solidarite(t+1) = solidarite(t) + \\ + \alpha(nbS(t+1) - nbD(t+1) - nbNE(t+1))$$

— **Familiarité** : En psychologie, les émotions et la cognition ne semblent pas avoir d'impact sur la familiarité. Cependant, Collins et Miller [10] expliquent que les personnes ont tendance à être plus familier avec les autres personnes qu'elles apprécient. Nous modélisons cette notion en basant l'évolution de la valeur de familiarité sur la valeur d'appréciation entre deux agents. La règle de calcul peut être formalisée comme suit :

$$familiarite(t+1) = familiarite(t) * \\ * (1 + appreciation(t+1))$$

4 Une architecture agent mêlant cognition, émotions et relations sociales

Le formalisme proposé dans cet article a été utilisé pour améliorer une architecture agent existante [7] dans le but de faciliter la définition d'agents sociaux avec des comportements prenant en compte la cognition, les émotions et les relations sociales. La nouvelle architecture obtenue est représentée par le schéma en figure 1.

Dans cette architecture, les bases cognitives de l'agent (croyances, incertitudes, désirs, intentions) contiennent des prédicats décrits en section 3, la base des émotions et la base des relations sociales contiennent respectivement des émotions et des relations sociales selon le formalisme décrit en section 3.

La première étape du cycle de raisonnement de l'agent avec notre architecture est la perception de l'environnement (étape 1 sur la figure 1). Les perceptions permettent de mettre à jour les

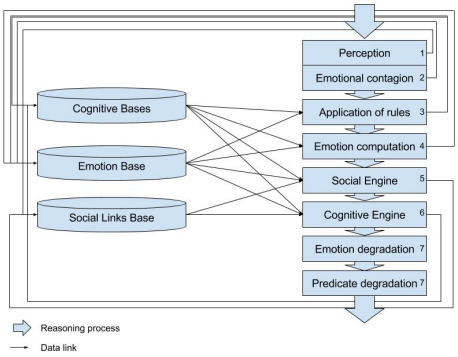


FIGURE 1 – Schéma de l'architecture cognitive, émotionnelle et sociale

croyances de l'agent et de créer des liens sociaux avec les autres agents rencontrés. Cette perception peut être paramétrée, avec une distance de perception par exemple. Une contagion émotionnelle est réalisée (étape 2) mais nous ne détaillons pas le processus car ce n'est pas l'objet de cet article.

Ensuite, l'agent applique des règles d'inférence (étape 3), définies par le modélisateur, pour modifier ses bases de croyances et de désirs en fonction de ses perceptions. Cette étape donne de la dynamique au comportement global car l'agent peut agir en fonction d'une modification de l'environnement. Ces règles d'inférence peuvent être influencées par des émotions ou des relations sociales.

Le module de calcul émotionnel (étape 4) est un moteur qui crée automatiquement, sans intervention du modélisateur, des émotions en fonctions des règles définies dans la section 3.2.

Le moteur social (étape 5) est utilisé pour mettre à jour dynamiquement les relations sociales de l'agent en fonction des règles énoncées en section 3.2.

Le moteur cognitif (étape 6) est basé sur le paradigme BDI et sélectionne un désir pour en faire une intention. Il sélectionne ensuite un plan décrit par le modélisateur pour répondre à l'intention courante. Ce processus est influencé par les bases cognitives mais aussi par les émotions et les relations sociales et peut, par l'exécution de plan, modifier ces mêmes bases. Le moteur cognitif est décrit en détail dans [37].

L'étape finale du cycle de raisonnement est la dégradation de la connaissance de l'agent (étape 7). Les prédicats stockés voient leur durée de vie baisser et l'intensité des émotions est réduite en fonction de la valeur de décroissance. Ce mécanisme offre une dynamique temporelle au comportement de l'agent.

Cette architecture (avec sa nouvelle extension) a été implémentée dans la plateforme GAMA [14]. Les modélisateurs peuvent déjà l'utiliser - avec simplement quelques lignes de code - à travers le langage de modélisation de GAMA.

5 Cas d'exemple

L'architecture exposée en section 4 a été utilisée sur le cas d'exemple de l'évacuation d'une grande zone ouverte subissant un feu de brousse en Australie.

5.1 Feux de brousse en Australie

Les feux de brousse sont une préoccupation importante en Australie car ils tuent des gens et détruisent des propriétés chaque année. Une étude a été menée pour simuler l'évacuation d'une zone pendant un incendie de brousse en utilisant une architecture BDI pour gérer le comportement des agents [2]. Si le modèle BDI proposé montre des résultats intéressants sur la réplication de situation réelle, il montre quelques limites puisqu'il ne prend pas en compte le comportement induit par les émotions ou les relations sociales.

Le but de cet exemple, dans notre travail, n'est pas de proposer un modèle réaliste pour simuler l'évacuation d'une zone pendant un incendie mais de montrer comment utiliser les émotions et les relations sociales fournies par notre architecture. Nous prenons comme base le modèle développé par [2] et nous expliquons comment ajouter des émotions et des relations sociales et comment prendre en compte ces composants dans le comportement des agents.

5.2 Description du modèle de base

Le modèle de base est fait d'un environnement composé de bâtiments, d'abris et d'incendies et définit des agents civils qui essayent de survivre. Les abris sont des endroits sûrs qui ne peuvent pas être endommagées par le feu tandis que les bâtiments peuvent brûler. Les feux sont placés aléatoirement dans l'environnement et, au cours de la simulation, ils peuvent grossir, se propager, brûler des bâtiments et des gens et, enfin, disparaître.

Les agents civils ont deux comportements principaux : soit ils restent dans leur habitation et ils combattent l'incendie ou alors ils fuient vers l'abri connu le plus proche. En fonction de leurs motivations ils choisissent l'intention de rester ou de fuir. Si la motivation de fuir devient plus grande que la motivation de rester, l'agent peut décider d'abandonner le combat contre l'incendie et essayer de fuir vers un abri.

5.3 Implémentation de l'exemple utilisant l'architecture développée

Ajout de relations sociales entre agents. Nous proposons d'améliorer le comportement des agents en terme de réalisme en leur donnant des relations sociales. Les liens sociaux entre les agents peuvent être inclus au démarrage de la simulation ou peuvent apparaître dynamiquement au cours de la simulation.

Un exemple de relation sociale qui peut exister au début de la simulation est la relation familiale. Dans le contexte des incendies de brousse, on peut aisément imaginer que deux membres d'une même famille vont essayer de s'aider entre eux pour survivre à la catastrophe.

Du point de vue de l'implémentation, le modélisateur a simplement besoin d'ajouter un lien familial lors de la phase d'initialisation des agents. Cette relation peut être représentée par un lien social envers le membre de la famille avec une valeur de familiarité à 1.0 comme montré en figure 2.

```
do add_social_link(
    new_social_link(familyMember) set_familiarity 1.0
);
```

FIGURE 2 – Définition d'un lien famillial

Les relations sociales peuvent aussi être utilisées dynamiquement lorsque les agents s'échappent vers des abris sans en connaître la localisation précise. En rencontrant un autre agent qui s'échappe, un lien social est créé. Si la valeur de solidarité du lien est assez élevée, les deux agents vont s'entraider pour aller vers un abri.

Après avoir créé dynamiquement une relation sociale comme montrée dans la figure 3, le modélisateur a simplement besoin de modifier le plan pour s'échapper afin de suivre un agent avec lequel un lien social avec une valeur de solidarité de 1.0 existe si l'emplacement de l'abri n'est pas connu.

```
perceive target: pedestrianBDI{
        socialize;
}
```

FIGURE 3 – Définition d'une création dynamique de lien social

Créer des émotions pour changer le comportement des agents. Pour ajouter des émotions aux agents civils, nous utilisons le module émotionnel de notre architecture qui crée automatiquement des émotions en fonction des états mentaux de l'agent.

La simulation se lance une première fois sans émotion en tant que session d'entraînement pour les agents civils. Un agent qui a décidé de fuir après avoir combattu l'incendie est fier d'avoir fui s'il est vivant et si sa maison est détruite à la fin de la session d'entraînement. Cette émotion de fierté augmente sa motivation à fuir dans le cas d'un futur incendie car il saura que sa maison ne peut pas être protégée des flammes. S'il est vivant mais que sa maison n'est pas détruite, il aura honte de s'être échappé et cette émotion va réduire sa motivation à fuir pour un futur cas. D'autre règles de ce genre peuvent être définies par le modélisateur, permettant de créer des émotions à la suite d'une session d'entraînement.

L'implémentation consiste à ajouter la croyance que l'agent est mort ou vivant, a essayé de fuir ou non ou encore si sa maison est détruite ou non à la fin de l'entraînement comme montré par la figure 4

Ensuite, le modélisateur peut définir des règles pour modifier les motivations de l'agent en fonction de ses émotions comme expliqué en figure 5. Les émotions générées lors de l'entraînement pourront alors servir dans une deuxième phase de la simulation qui reproduirait véritablement un incendie de brousse. Les agents auront alors un comportement différent de celui qu'ils ont eu en entraînement.

```
do add_belief(flee with_praiseworthiness -1.0);
```

FIGURE 4 – Ajout de la croyance correspondant à l'état de l'agent à la fin de la session d'entraînement

Le modèle complet se trouve à l'adresse suivante :

https://github.com/mathieuBourgais/ExampleModels

```
if(has_emotion(shameFlee)){
    if(escape_motivation>0.0){
        escape_motivation <-
         escape_motivation*(1.0-0.2);
    }
}
```

FIGURE 5 – Utilisation d'une émotion de honte pour mettre à jour la motivation de fuite

5.4 Discussion

L'implémentation de l'exemple nous montre qu'un modélisateur peut simplement améliorer le comportement de ses agents en écrivant quelques ligne de code en GAML, le langage de programmation de la plateforme GAMA. Comme montré par [20] ou [27], ce langage de programmation est plutôt simple à apprendre et à utiliser par des scientifiques qui ne sont pas expert en programmation.

De plus, des modélisateurs avec plus de connaissances en programmation et en intelligence artificielle peuvent utiliser plus en profondeur notre architecture pour créer des comportements complexes. Par exemple, un expert en théories émotionnelles peut redéfinir manuellement certaines émotions grâce à la définition de règles d'inférences ou alors utiliser plusieurs émotions avec différentes intensités et différents liens sociaux pour créer des agents crédibles pour leurs simulations sociales.

6 Conclusion

Dans cet article, nous avons présenté un formalisme pour utiliser des émotions et des relations sociales dans la modélisation d'agents cognitifs sociaux. Ce formalisme a été intégré dans une architecture agent et implémenté dans une plateforme de simulation multi-agent pour montrer sa facilité d'utilisation du pour des modélisateurs n'ayant que des connaissances basiques en programmation. L'exemple des incendies de brousse en Australie indique une façon d'utiliser notre travail sur la simulation de l'évacuation d'un grand espace ouvert.

Dans l'avenir, nous projetons de mener des expérimentations avec des modélisateurs pour tester la facilité d'utilisation de notre architecture. Nous voulons aussi améliorer notre travail en ajoutant une personnalité ainsi qu'une gestion des normes sociales pour créer des agents sociaux de plus en plus réalistes tout en conservant à l'esprit la contrainte de facilité de prise en main pour des personnes non expert en programmation.

Remerciements

Ce travail est réalisé dans le cadre de l'ANR ACTEUR (Agents Cognitifs Territorialisés pour l'Étude des dynamiques Urbaines et des Risques).

Références

[1] Carole Adam. Emotions : from psychological theories to logical formalization and implementation in a bdi agent, 2007.

[2] Carole Adam and Julie Dugdale. Comparing agent architectures in social simulation : Bdi agents versus finite-state machines. In *HICSS*, 2016.

[3] Carole Adam and Benoit Gaudou. Bdi agents in social simulations : a survey. *The Knowledge Engineering Review*, 2016.

[4] Michael Balmer, Marcel Rieser, Konrad Meister, David Charypar, Nicolas Lefebvre, Kai Nagel, and K Axhausen. Matsim-t : Architecture and simulation times. *Multi-agent systems for traffic and transportation engineering*, 2009.

[5] Joseph Bates. The role of emotion in believable agents. *Communications of the ACM*, 1994.

[6] Timothy Bickmore and Justine Cassell. Relational agents : a model and implementation of building user trust. In *Proceedings of SIGCHI*. ACM, 2001.

[7] M. Bourgais, P. Taillandier, and L. Vercouter. An agent architecture coupling cognition and emotions for simulation of complex systems. In *SSC*, 2016.

[8] M Bratman. Intentions, plans, and practical reason. 1987.

[9] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. 1990.

[10] Nancy L Collins and Lynn Carol Miller. Self-disclosure and liking : a meta-analytic review. *Psychological bulletin*, 1994.

[11] Joseph de Rivera and Carmen Grinkis. Emotions as social relationships. *Motivation and emotion*, 1986.

[12] Jonathan Gratch. Socially situated planning. In *Socially Intelligent Agents*. Springer, 2002.

[13] Jonathan Gratch and Stacy Marsella. A domain-independent framework for modeling emotion. *Cognitive Systems Research*, 2004.

[14] Arnaud Grignard, Patrick Taillandier, Benoit Gaudou, Duc An Vo, Nghi Quang Huynh, and Alexis Drogoul. *GAMA 1.6 : Advancing the Art of Complex Agent-Based Modeling and Simulation*. Springer Berlin Heidelberg, 2013.

[15] Nick Howden, Ralph Rönnquist, Andrew Hodgson, and Andrew Lucas. Jack intelligent agents-summary of an agent infrastructure. In *5th International conference on autonomous agents*, 2001.

[16] Katherine Isbister. *Better game characters by design : A psychological approach*. Elsevier San Francisco, 2006.

[17] Hong Jiang, Jose M Vidal, and Michael N Huhns. Ebdi : an architecture for emotional agents. In *AAMAS*. ACM, 2007.

[18] Dacher Keltner and Jonathan Haidt. Social functions of emotions. 2001.

[19] Brian Knutson. Facial expressions of emotion influence interpersonal trait inferences. *Journal of Nonverbal Behavior*, 1996.

[20] EG Macatulad and AC Blanco. 3dgis-based multi-agent geosimulation and visualization of building evacuation using gama platform. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2014.

[21] Karen L Myers. User guide for the procedural reasoning system. *SRI International AI Center Technical Report.*, 1997.

[22] Ranjit Nair, Milind Tambe, and Stacy Marsella. The role of emotions in multiagent teamwork. *Who Needs Emotions*, 2005.

[23] Magalie Ochs, Nicolas Sabouret, and Vincent Corruble. Simulation of the dynamics of nonplayer characters' emotions and social relations in games.

[24] Andrew Ortony. Memories, thoughts, and emotions : Essays in honor of george mandler, chapter value and emotion, 1991.

[25] Andrew Ortony, Gerald L Clore, and Allan Collins. *The cognitive structure of emotions*. Cambridge university press, 1990.

[26] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. Jadex : A bdi reasoning engine. In *Multi-agent programming*. Springer, 2005.

[27] Nur Raihan Ramli, Sazalinsyah Razali, and Mashanum Osman. An overview of simulation software for non-experts to perform multi-robot experiments. In *ISAMSR*. IEEE, 2015.

[28] Ralph Rönnquist. The goal oriented teams (gorite) framework. In *International Workshop on Programming Multi-Agent Systems*. Springer, 2007.

[29] Daniel Rousseau and Barbara Hayes-Roth. A social-psychological model for synthetic actors. In *Proceedings of the second international conference on Autonomous agents*. ACM, 1998.

[30] Ilias Sakellariou, Petros Kefalas, and Ioanna Stamatopoulou. Enhancing netlogo to simulate bdi communicating agents. In *Hellenic Conference on Artificial Intelligence*. Springer, 2008.

[31] Klaus R Scherer and Paul Ekman. On the nature and function of emotion : A component process approach. *Approaches to emotion*, 2293 :317, 1984.

[32] Dhirendra Singh and Lin Padgham. Opensim : A framework for integrating agent-based models and simulation components. In *ECAI*. IOS Press, 2014.

[33] Craig A. Smith and Richard S. Lazarus. Emotion and adaptation. 1990.

[34] Eliot R Smith, Diane M Mackie, and Heather M Claypool. *Social psychology*. Psychology Press, 2014.

[35] Ron Sun. *Cognition and multi-agent interaction : From cognitive modeling to social simulation*. Cambridge University Press, 2006.

[36] Jan Svennevig. *Getting acquainted in conversation : a study of initial interactions*. John Benjamins Publishing, 2000.

[37] Patrick Taillandier, Mathieu Bourgais, Philippe Caillou, Carole Adam, and Benoit Gaudou. A bdi agent architecture for the gama modeling and simulation platform. In *MABS 2016*.

[38] H. Van Dyke Parunak, Robert Bisson, Sven Brueckner, Robert Matthews, and John Sauter. A model of emotions for situated agents. In *Proceedings of AAMAS*. ACM, 2006.

[39] Uri Wilensky and I Evanston. Netlogo : Center for connected learning and computer-based modeling. *Northwestern Univ., Evanston, IL*, 1999.

Coopération entre agents autonomes fondée sur l'éthique

N. Cointe[a,b] nicolas.cointe@emse.fr
G. Bonnet[b] gregory.bonnet@unicaen.fr
O. Boissier[a] olivier.boissier@emse.fr

[a]Université de Lyon, MINES Saint-Etienne, CNRS, Laboratoire Hubert Curien, UMR 5516

[b]Normandie Université, GREYC CNRS UMR 6072, F-14032 Caen, France

Résumé

Dans le domaine de la décision autonome, la prise en compte de la dimension éthique des décisions est généralement centrée sur l'agent, en laissant de côté sa dimension sociale. Or, l'éthique semble être une notion centrale influençant les interactions sociales entre individus. Dans cet article, nous proposons un modèle permettant à des agents de se construire une image du comportement éthique et moral des autres afin de le prendre en compte dans leurs interactions. Fondé sur une approche rationaliste et explicite, ce modèle distingue l'éthique de la moralité et permet d'aboutir à l'établissement ou non d'une relation de confiance. Nous illustrons ces fonctionnalités dans une preuve de concept dans le domaine de la gestion d'actifs financiers implémentée à l'aide de la plateforme JaCaMo.

Mots-clés : *Architecture d'agent, Modèles de comportement agent, Éthique computationelle*

Abstract

In decision theory, dealing with ethics is mainly considered in an agent-centered perspective, letting aside the social dimension of multi-agent systems. However, ethics seems to be a central notion when considering interactions among agents. In this paper, we propose a model for ethics-based cooperation. Each agent uses an ethical judgment process based on a rationalist and explicit approach to compute images of the other agents' ethical behavior. From these images of the other agents' ethics, the judging agent computes trust used to interact with the judged agents. We illustrate these functionalities in an asset management scenario with a proof-of-concept implemented in the JaCaMo Multi-Agent Platform.

Keywords: *Agent architecture, Agent's model of behavior, Computer ethics*

1 Introduction

L'usage croissant des agents autonomes dans un grand nombre de domaines tels que la santé, les transports ou la finance ajoute aux habituels problèmes de l'optimalité de leurs décisions, la problématique de la prise en compte des dimensions morales et éthiques de leur choix dans leur raisonnement. Par exemple, dans le cas de la gestion éthique d'actifs financiers, un grand nombre de modèles ont été suggérés pour prendre des décisions profitables [2], mais bien peu de solutions permettent à un agent de juger de la conformité de ses investissements avec les valeurs morales et principes éthiques des investisseurs qu'il représente. De plus, l'hétérogénéité de ces éléments soulève de nombreux problèmes lorsque les agents ont besoin de collaborer avec d'autres agents en respectant leur propre éthique.

Par exemple, comment un agent peut-il évaluer la conformité du comportement des autres agents à une éthique qui lui est propre ? Comment un tel agent peut-il décider d'accorder sa confiance à un autre en se fondant sur cette évaluation ? L'objectif de cet article est de répondre à de telles questions en proposant un cadre permettant de construire des mécanismes de coopération entre agents reposant sur la construction d'images du comportement d'autrui et de confiance par agrégation de jugements éthiques.

Pour ce faire, nous incorporons le processus de jugement éthique proposé dans [14] au sein d'une architecture BDI pour permettre à un agent de juger du comportement des autres. La conception de tels agents autonomes suppose qu'un concepteur ou un utilisateur décrive l'éthique et la morale employés. En utilisant ce processus, nous proposons ensuite de construire une image de l'autre en évaluant et en agrégeant ces jugements. Ensuite, nous proposons de permettre à un agent de prendre en compte cette image pour décider de faire confiance à un autre afin d'envisager des actions de coopération. En-

fin, nous instancions ce modèle dans une application de gestion d'actifs financiers et montrons son usage dans une preuve de concept implémentée à l'aide de la plate-forme JaCaMo [7].

Cet article est organisé de la manière suivante. La Sec. 2 introduit et décrit les modèles d'éthique computationnelle et de confiance employés dans ce travail. Ensuite, la Sec. 3 montre comment le jugement éthique peut être utilisé pour se représenter l'éthique du comportement de l'autre. Puis, la Sec. 4 présente la construction et l'utilisation de la confiance. Enfin, nous illustrons l'usage de ce travail dans une preuve de concept en Sec. 5 avant de conclure.

2 Concepts principaux

Nous introduisons dans cette section les concepts nécessaires à la coopération fondée sur l'éthique dans les systèmes multi-agents. La Sec. 2.1 montre comment le concept de confiance peut guider les interactions et la coopération entre agents. La Sec. 2.2 introduit l'éthique et montre comment elle peut mener à la confiance. Enfin, la Sec. 2.3 propose une synthèse des éléments nécessaires à la définition de la coopération fondée sur l'éthique. Cette synthèse constitue l'ossature de la proposition décrite dans ce papier.

2.1 Confiance dans les SMA

Dans les systèmes décentralisés et ouverts, la confiance est un moyen de coexister et d'interagir avec des agents inconnus et à la fiabilité incertaine [11, 16, 31]. La confiance permet aux agents d'évaluer les interactions observées ou effectuées pour décider si collaborer avec un agent est a priori acceptable. Cette notion d'acceptation signifie que le comportement de l'agent observé est considéré comme bon et fiable du point de vue des critères de l'agent observateur.

De nombreuses définitions de la confiance existent mais, comme le fait [11], nous considérons la *confiance* comme *une disposition à coopérer avec un individu de confiance*. Ainsi, elle peut être utilisée comme une condition pour effectuer certaines actions de délégation, de partage de ressources ou d'information, ou toute forme de coopération. Pour construire cette confiance, les agents commencent par se construire une image de l'agent observé [16].

Si [16] définissent une *image* comme *une croyance qualifiant le sujet de bon ou mauvais selon son comportement* dans le cadre des systèmes de confiance, nous préférons la définir comme *une croyance qualifiant le sujet de conforme ou non selon l'adéquation de son comportement à un ensemble de règles* afin de lever toute ambiguïté sur les termes *bien* et *mal* au sens moral. Dans la littérature, les images sont agrégées à partir de l'expérience, c'est-à-dire l'observation des comportements et de leurs conséquences. Nous pouvons distinguer deux types d'approches : (1) les images statistiques [1, 9, 17, 25, 35] où l'image est une agrégation quantitative d'appréciations d'interactions passées. Cette agrégation estime la tendance d'un agent à agir conformément à des critères. Cela peut être représenté par des réseaux bayésiens, des lois de probabilité bêta, des ensembles flous, des fonctions de Dempster-Shafer et d'autres formalismes quantitatifs ; (2) les images logiques [10, 11, 27, 34] où l'image est un état mental lié à toute action de coopération produite par interaction. Une image persistante permet d'inférer des croyances sur la confiance pouvant être utilisées comme préconditions pour des actions de coopération.

Un agent peut manquer d'observations pour construire une image correcte de l'agent jugé. Une manière de traiter ce problème est d'utiliser la réputation [23, 30]. Cela consiste en l'utilisation de l'image qu'un tiers a de l'agent jugé afin d'obtenir un point de vue collectif sur le jugé. Le choix de cet agent tiers peut lui-même dépendre de l'image que l'agent jugé a des autres Ainsi, les images individuelles et la réputation peuvent être utilisées conjointement pour décider d'établir une confiance [31]. De manière générale, la confiance est dynamique car elle change en fonction de l'évolution des images et des réputations.

2.2 Comportements éthiques

Dans cet article, nous nous intéressons à la construction d'images de l'éthique du comportement des autres agents. En raison de l'absence de définitions formelles dans la littérature, nous admettons la définition suivante [33] : les connaissances de l'agent sont réparties en deux composantes, la *théorie du bien* (ou morale) et la *théorie du juste* (ou éthique). Bien que cette distinction soit discutable en raison de la grande diversité de théories contradictoires en sciences humaines, nous estimons que ces définitions fournissent un cadre intéressant pour représenter la morale et l'éthique.

Une *théorie du bien* est un ensemble de règles et valeurs morales qui permettent d'évaluer la moralité (le caractère bon ou mauvais) d'un com-

portement. Les règles morales attribuent des valuations morales à des comportements (par exemple "Mentir est mal" ou "Être honnête est bien"), et les valeurs permettent de qualifier des actions de manière plus abstraites (par exemple "Il est honnête de dire ce que l'on pense").

Une *théorie du juste* utilise un ensemble de principes éthiques pour reconnaître un choix juste, ou au moins acceptable. Les philosophes ont proposé un ensemble varié de principes éthiques, tels que l'impératif catégorique de Kant[22] ou la doctrine du double effet de Saint Thomas D'Aquin[26]. Par exemple même s'il est immoral de voler, (au regard des commandements divins), plusieurs philosophes admettent qu'il est acceptable pour des gens affamés de voler de la nourriture (au regard de la doctrine du double effet).

Comme la moralité d'un comportement repose sur une théorie du bien, son caractère éthique repose dans la conciliation des désirs, de la morale et des capacités de l'agent au regard d'une théorie du juste [28]. Ainsi, être moral ou éthique caractérise un comportement dans un contexte donné, tout comme être fiable caractérise un comportement dans un système de confiance. Par conséquent, il peut être intéressant de définir une notion de confiance dans la moralité ou le caractère éthique d'un comportement qui pourrait venir renforcer une coopération.

2.3 Une coopération fondée sur l'éthique

Plusieurs travaux prenant en compte la dimension éthique du comportement d'agents autonomes se focalisent sur la modélisation d'un raisonnement moral [6, 19, 20, 32] comme une traduction directe de théories bien connues, ou la modélisation du comportement moral en général [5, 24]. Toutefois, ces travaux ne permettent ni de représenter des valeurs morales, ni d'employer plusieurs principes éthiques dans un raisonnement. D'autres travaux traitent de l'architecture des agents éthiques. Parmi celles-ci, les *architectures éthiques implicites* [3, 4] proposent soit de concevoir des agents en implémentant dans tout état possible des moyens d'empêcher des actions non éthiques, soit un apprentissage supervisé de l'éthique. D'un autre côté, les *architectures éthiques cognitives* [12, 13, 14, 15] consistent en une représentation totalement explicite de chaque composant de l'agent, des croyances classiques (informations sur l'environnement et les autres agents), désirs (objectifs de l'agent) et intentions (actions choisies) à des concepts tels que des heuristiques ou des simulations émotionnelles. Toutefois, ces approches ne tiennent pas compte de la dimension collective de ces systèmes, excepté [29] qui considère la morale comme un élément des sociétés d'agents.

Plus précisément, l'architecture donnée en [14] propose une séparation claire entre théorie du bien et théorie du juste, et propose des croyances portant sur des composants (principes éthiques, valeurs et règles morales, etc.). De plus, l'architecture proposée par [29] permet – sans en proposer une version opérationnelle – de voir des faits moraux (des jugements sur les autres ou des blâmes par exemple) comme des croyances pouvant être employées dans les décisions des agents.

Afin de construire une coopération fondée sur l'éthique, nous avons besoin d'un modèle opérationnel de jugement éthique tel que celui proposé en [14]. Inspiré de [29], nous réutilisons et étendons ce modèle avec des croyances sur les images morales et éthiques des autres agents. Ces images sont ensuite utilisées pour construire des relations de confiance permettant d'influencer la coopération entre agents.

3 Processus de jugement

Nous présentons en Sec. 3.1 le processus de jugement décrit en [14]. Nous montrons ensuite comment un agent peut l'employer pour construire sa propre représentation qualitative de l'éthique (voir Sec. 3.2) et de la morale (voir Sec. 3.3) des autres agents, au regard des croyances de l'agent juge sur la *connaissance du bien* et la *connaissance du juste*.

3.1 Jugement des autres agents

Nous considérons le processus de jugement introduit dans [14] répondant à nos besoins énoncés en Sec. 2. Ce processus fournit une évaluation des actions connues de l'agent en matière de conformité à un ensemble de connaissances données.

Comme le montre la Figure 1, le processus de jugement est organisé en trois parties : (*i*) le processus de reconnaissance de situation et d'évaluation, (*ii*) le processus moral et (*iii*) le processus éthique. Comme ce processus de jugement peut raisonner de manière interchangeable sur les données d'autres agents, nous indiçons dans la suite la totalité des connaissances par l'identifiant de l'agent dont elles proviennent $a_i \in \mathbb{A}$ (par exemple $\mathcal{A}_{r_{a_i}}$) avec $\mathbb{A}$ l'ensemble des

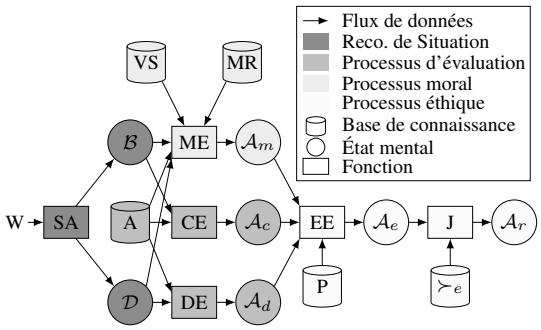


FIGURE 1 – Processus de jugement éthique [14]

agents. Il est ainsi possible pour un agent d'utiliser ses propres connaissances pour juger de son propre comportement ou de celui d'un autre, ou bien d'utiliser tout ou partie des connaissances d'un autre pour le juger.

Processus de reconnaissance et d'évaluation. Dans ce modèle, les actions sont décrites sous forme d'actions et de conséquences exprimées en termes de changements de désirs et des croyances. Le *processus d'évaluation* évalue alors l'ensemble des actions $\mathcal{A}_{a_i}$ qu'il considère comme désirable ($\mathcal{A}_{d_{a_i}}$) et réalisable ($\mathcal{A}_{c_{a_i}}$) du point de vue des connaissances de a_i conformément aux désirs $\mathcal{D}_{a_i}$ et croyances $\mathcal{B}_{a_i}$. $\mathcal{B}_{a_i}$ et $\mathcal{D}_{a_i}$ sont produits par le processus de *reconnaissance de situation* SA à partir de la perception de l'état courant. Ici, DE et CE sont respectivement les fonctions d'*évaluation de la désirabilité* et d'*évaluation de la faisabilité*. Par la suite, nous appelons *connaissance contextuelle* de a_i l'union de $\mathcal{B}_{a_i}$ et $\mathcal{D}_{a_i}$ que nous notons CK_{a_i}.

Processus moral. Le *processus moral* produit l'ensemble des actions moralement évaluées $\mathcal{A}_{m_{a_i}}$ au regard des données contextuelles CK_{a_i} de l'agent a_i, des actions A_{a_i}, des supports de valeurs VS_{a_i} et des règles morales MR_{a_i}. Ces actions sont celles qui, dans la situation décrite par CK_{a_i}, promeuvent ou trahissent les valeurs morales de VS_{a_i} et se trouvent évaluées par les règles morales de MR_{a_i}. Un *support de valeur* est un couple $\langle s, v\rangle \in VS_{a_i}$ où $v \in \mathcal{O}_v$ est une valeur morale et $s = \langle \alpha, w\rangle$ est le support de cette valeur morale où $\alpha \in A_{a_i}$, $w \subset \mathcal{B}_{ai} \cup \mathcal{D}_{a_i}$. $\mathcal{O}_v$ est l'ensemble des valeurs morales utilisées dans le système [1]. Une *règle morale* est un tuple $\langle w, o, m\rangle \in MR_{a_i}$. La situation $w \in 2^{CK_{a_i}}$ est une conjonction de croyances et désirs. o est l'objet de la règle avec $o = \langle \alpha, v\rangle$ où α est une action ($\alpha \in A_{a_i}$) et v est une valeur morale ($v \in \mathcal{O}_v$). Enfin, m est la valuation morale ($m \in \mathcal{O}_m$). Par exemple, $\mathcal{O}_m = \{\texttt{moral}, \texttt{amoral}, \texttt{immoral}\}$ permet d'associer une parmi trois valuations morales à o lorsque w est vrai. Notons qu'un ordre total doit être défini sur $\mathcal{O}_m$ (par exemple `moral` est une valuation supérieure à `amoral`, qui est supérieure à `immoral`). Par la suite, les connaissances sur les règles morales MR_{a_i}, supports de valeurs VS_{a_i} et valeurs $\mathcal{O}_v$, utilisées dans le processus moral de l'agent sont appelés *connaissances morales* et sont notées GK_{a_i}.

Processus éthique. Enfin, le processus éthique évalue l'action juste $\mathcal{A}_{r_{a_i}}$ à partir de l'ensemble des action possibles $\mathcal{A}_{c_{a_i}}$, désirables $\mathcal{A}_{d_{a_i}}$ et morales $\mathcal{A}_{m_{a_i}}$ par rapport à un ensemble de *principes éthiques* P_{a_i} pour concilier ces ensembles d'actions conformement à un ensemble de relations de préférences éthiques $\succ_{e_{a_i}} \subseteq P_{a_i} \times P_{a_i}$. Un *principe éthique* $p \in P_{a_i}$ est une fonction qui évalue s'il est juste ou non d'exécuter une action dans une situation donnée au regard d'une théorie philosophique. Cette évaluation est exprimée au travers d'évaluations des actions de $\mathcal{A}_{c_{a_i}}$, $\mathcal{A}_{d_{a_i}}$ et $\mathcal{A}_{m_{a_i}}$ dans une situation décrite par CK_{a_i}. Un principe est défini par $p : 2^{\mathcal{A}_{a_i}} \times 2^{\mathcal{B}_{a_i}} \times 2^{\mathcal{D}_{a_i}} \times 2^{MR_{a_i}} \times 2^{V_{a_i}} \rightarrow \{\top, \bot\}$. Étant donné un ensemble d'actions évaluées issues de la fonction d'évaluation éthique EE utilisant les principes éthiques, le jugement J est la dernière étape pour choisir l'action juste à effectuer, en considérant l'ensemble des préférences éthiques $\succ_{e_{a_i}}$ définissant un ordre total sur les principes éthiques. Dans ce processus de jugement, les actions justes sont celles qui satisfont les principes éthiques préférés selon un critère lexicographique. Par la suite, les principes éthiques P_{a_i} et préférences $\succ_{e_{a_i}}$ sont appelés *connaissance éthique* et sont notés RK_{a_i}.

3.2 Juger de la conformité éthique

Nous étendons maintenant le processus de jugement précédemment présenté pour juger l'éthique et la moralité d'un comportement observé attribué à un agent a_j sur une période de temps de t_0 à t. Inspiré de [29] qui considère les croyances sur des faits moraux, le processus de jugement produit ici des croyances (`ethical_conf`, `moral_conf`) informant de la conformité d'une action à des principes éthiques ou des règles et valeurs morales. Avant de définir ces croyances, nous définissons le comportement d'un agent de la façon suivante :

1. Notons que dans [14] les valeurs morales et valuations morales sont partagées entre les agents du système : les agents se distinguent par les règles morales et les éléments de leur processus éthique.

Définition 1 (Comportement) *Le* comportement $b_{a_j,[t_0,t]}$ *d'un agent* a_j *sur l'intervalle* $[t_0, t]$ *est l'ensemble des actions* α_k *que* a_j *a exécuté entre* t_0 *et* t *tel que* $0 \leqslant t_0 \leqslant t$.

$$b_{a_j,[t_0,t]} = \{\alpha_k \in A : \exists t' \in [t_0, t] \text{ s.t. } \texttt{done}(a_j, \alpha_k, t')\}$$

où $A = \bigcup_{a_i=a_1}^{a_n} \mathcal{A}_{a_i}$ *est l'ensemble des actions disponibles dans un système à* n *agents et* $\texttt{done}(a_j, \alpha_i, t')$ *un prédicat signifiant que l'action* α_i *a été exécutée*[2] *par* a_j *à l'instant* t'.

Un agent a_i peut juger la conformité d'une action α_k exécutée par un autre agent a_j conformément à sa connaissance morale et éthique.

Définition 2 (Conformité éthique) *Une action* α_k *est* éthiquement conforme *par rapport aux connaissances contextuelles* (CK_{a_i})*, connaissances morales* GK_{a_i} *et connaissances éthiques* RK_{a_i} *d'un agent* a_i *au temps* t' *si et seulement si* α_k *est dans l'ensemble des actions justes* $\alpha_k \in \mathcal{A}_{r_{a_i}}$ *évaluées par le jugement éthique* J_{a_i} *de l'agent juge* a_i *en se fondant sur* $[CK_{a_i}, GK_{a_i}, RK_{a_i}]$ *à l'instant* t'*. Une telle action produit une croyance notée :*

$$\texttt{ethical\_conf}(\alpha_k, [CK_{a_i}, GK_{a_i}, RK_{a_i}], t')$$

Notons que la conformité éthique d'une action peut être appliquée aux actions de l'agent juge ou celles exécutées par un autre agent et observées par l'agent juge. Cette conformité éthique peut être évaluée par rapport aux connaissances contextuelles, morales et éthiques de l'agent juge, ou bien celles d'un autre agent si l'agent juge peut disposer d'une représentation de celles-ci. Finalement, la conformité éthique est employée pour produire l'ensemble EC^+ des actions éthiquement conformes (resp. l'ensemble EC^- de celles qui ne sont pas éthiquement conformes) du comportement observé $b_{a_j,[t_0,t]}$ de l'agent jugé a_j entre t_0 et t (voir Fig. 2).

Ces deux ensembles fournissent une information sur le comportement de l'agent jugé et sa conformité avec les connaissances employées pour le juger. Néanmoins, il est n'est pas possible en l'état de savoir pour quelle raison un comportement ne serait pas éthique. De fait, les raisons peuvent être une différence de connaissance contextuelle, morale ou éthique. Par la suite, nous notons $EC_{a_j,[t_0,t]} = EC^+_{a_j,[t_0,t]} \cup EC^-_{a_j,[t_0,t]}$.

2. Un comportement peut tenir compte de la concurrence : plusieurs actions peuvent être exécutées à un même instant.

3.3 Juger de la conformité morale

Dans le modèle de jugement, si le jugement éthique indique une conformité ou non avec un principe, les règles morales indiquent une conformité par rapport à un ensemble de valuations morales. Ainsi, l'évaluation de la conformité morale d'une action à une règle morale donnée se fait en comparant la valuation morale associée à l'action par la règle à une valuation seuil $mt \in MV$.

Définition 3 (Conformité morale) *Une action* α_k *est* moralement conforme *par rapport aux connaissances contextuelles* (CK_{a_i})*, connaissances morales* GK_{a_i} *et connaissances éthiques* RK_{a_i} *d'un agent* a_i *au temps* t' *au regard de la règle* $mr \in MR_{a_i}$ *et un seuil* $mt \in MV_{a_i}$ *si et seulement si* α_k *appartient à l'ensemble des actions morales* $A_{m_{a_i}}$ *et se trouve affectée d'une valuation morale supérieure ou égale à* mt*, au regard de l'action morale* mr *et des connaissances* CK_{a_i}*,* GK_{a_i} *et* RK_{a_i} *à l'instant* t'*. Une telle action produit une croyance notée :*

$$\texttt{moral\_conf}(\alpha_k, [CK_{a_i}, GK_{a_i}, RK_{a_i}], mr, mt, t')$$

De même que pour la conformité éthique, nous utilisons la conformité morale pour générer l'ensemble MC^+ (resp. MC^-) des actions moralement conformes (resp. moralement non conformes) du comportement observé $b_{a_j,[t_0,t]}$ de l'agent jugé a_j entre t_0 et t au regard de la règle mr et d'un seuil moral mt (voir Fig. 2).

Nous généralisons cette équation de la conformité morale au regard d'une règle en la transposant à un ensemble de règles, en supposant la définition préalable d'ensembles de règles morales $R \subseteq MR_{a_i}$. Un ensemble R peut, par exemple représenter l'ensemble des règles en rapport avec l'expression d'une valeur, une situation spécifique, etc. Par la suite, nous notons $MC_{a_j,mr,mt,[t_0,t]} = MC^+_{a_j,mr,mt,[t_0,t]} \cup MC^-_{a_j,mr,mt,[t_0,t]}$.

4 Construction de la confiance

Dans cette section nous utilisons les résultats de jugements définis dans la section précédente pour construire l'image des autres agents (cf. Sec. 4.1). Nous montrons ensuite comment ces images sont employées pour établir une relation de confiance (cf. Sec. 4.2). La Sec. 4.3 montre enfin comment cette relation influence le comportement de l'agent.

$$EC^{+}_{a_j,[t_0,t]}=\{\alpha_k{:}\exists t'\in[t_0,t] \text{ s.t. } \texttt{done}(a_j,\alpha_i,t')\wedge \texttt{ethical\_conf}(\alpha_k,[CK_{a_i},GK_{a_i},RK_{a_i}],t')\}$$

$$EC^{-}_{a_j,[t_0,t]}=\{\alpha_k{:}\exists t'\in[t_0,t] \text{ s.t. } \texttt{done}(a_j,\alpha_i,t')\wedge \neg\texttt{ethical\_conf}(\alpha_k,[CK_{a_i},GK_{a_i},RK_{a_i}],t')\}$$

$$MC^{+}_{a_j,mr,mt,[t_0,t]}=\{\alpha_k{:}\exists t'\in[t_0,t] \text{ s.t. } \texttt{done}(a_j,\alpha_i,t')\wedge \texttt{moral\_conf}(\alpha_k,[CK_{a_i},GK_{a_i},RK_{a_i}],mr,mt,t')\}$$

$$MC^{-}_{a_j,mr,mt,[t_0,t]}=\{\alpha_k{:}\exists t'\in[t_0,t] \text{ s.t. } \texttt{done}(a_j,\alpha_i,t')\wedge \neg\texttt{moral\_conf}(\alpha_k,[CK_{a_i},GK_{a_i},RK_{a_i}],mr,mt,t')\}$$

FIGURE 2 – Ensembles des actions moralement ou éthiquement évaluées

4.1 Images éthique et morale des agents

Comme mentionné en Sec. 2.1, les images *éthiques* et *morales* d'un agent sont des croyances décrivant la conformité du comportement d'un agent au regard d'une connaissance du juste (RK) et du bien (GK).

Définition 4 (Image éthique (resp. morale)) *Une* image éthique *(resp.* image morale*) d'un agent a_j est le jugement du comportement $b_{a_j,[t_0,t]}$ de cet agent conformément à une éthique (resp. à un ensemble de règles morales R et d'un seuil moral mt) au regard des connaissances du contexte CK, de la moralité GK et de l'éthique RK d'un agent a_i. Cette image associe au comportement un élément $cv \in CV$ où CV est un ensemble ordonné de* niveaux de conformité [3]*. Ces images sont notées* $\texttt{ethical\_image}(a_j,a_i,cv,t_0,t)$ *et* $\texttt{morality\_image}(a_j,a_i,cv,R,mt,t_0,t)$

Remarquons tout d'abord que le premier paramètre fait référence à l'agent dont le comportement est évalué tandis que le second paramètre fait référence à l'agent dont la connaissance est employée pour construire l'image.

De plus, comme un jugement éthique indique une conformité ou non avec des principes éthiques, l'image éthique indique si un agent est conforme ou non dans son comportement. Ainsi, un agent ne peut avoir qu'une seule image éthique du comportement d'un autre. Dans le cas de l'image morale, comme les règles morales sont associées à divers niveaux de moralité, une image morale est associée à un seuil d'exigence pour indiquer une conformité morale ou non. Ainsi, un agent peut avoir plusieurs images morales en s'appuyant sur divers ensembles de règles R et leur seuil mt.

Afin de construire ces images, un agent a_i utilise deux fonctions d'agrégation $ethicAggregation$ et $moralAggregation$ appliquées respectivement aux actions éthiquement évaluées $EC_{a_j,[t_0,t]}$ et moralement évaluées $MC_{a_j,[t_0,t]}$. Ces deux fonctions d'agrégation calculent respectivement la proportion pondérée d'actions positivement évaluées au regard de l'éthique et de la morale. Le poids de chaque action dépend d'un critère (tel que le temps passé depuis son évaluation, les conséquences de l'action, etc.) que nous laissons volontairement abstrait dans cet article.

Définition 5 (Fonction d'agrégation éthique) $ethicAggregation : 2^{\mathcal{A}} \rightarrow \mathbb{R}$ *est une fonction d'agrégation éthique où :*

$$ethicAggregation(EC_{a_j,[t_0,t]}) = \sum_{\alpha_i\in EC^{+}_{a_j,[t_0,t]}} weight(\alpha_i) / \sum_{\alpha_i\in EC_{a_j,[t_0,t]}} weight(\alpha_i)$$

Définition 6 (Fonction d'agrégation morale) $moralAggregation : 2^{\mathcal{A}} \rightarrow \mathbb{R}$ *est une fonction d'agrégation morale où :*

$$moralAggregation(MC_{a_j,[t_0,t]}) = \sum_{\alpha_i\in MC^{+}_{a_j,[t_0,t]}} weight(\alpha_i) / \sum_{\alpha_i\in MC_{a_j,[t_0,t]}} weight(\alpha_i)$$

Afin de transformer l'évaluation quantitative en une évaluation qualitative, chaque niveau de conformité est associé à un intervalle dans l'ensemble des valeurs que peuvent prendre les fonctions d'agrégation éthiques et morales. Une fois le niveau de conformité obtenu, les états mentaux associés $\texttt{moral\_image}(a_j,a_i,cv,R,mt,t_0,t)$ ou $\texttt{ethical\_image}(a_j,a_i,cv,t_0,t)$ sont produits. Par exemple, si le niveau de conformité `congruent` correspond à l'intervalle $[0.75;1]$, le comportement de l'agent est considéré comme éthique si $ethicAggregation \geq 0.75$.

Une fois construites, ces images peuvent être employées pour influencer les interactions en construisant des relations de confiance, ou pour décrire la moralité d'interaction dépendantes du comportement des autres.

3. De manière analogue aux valuations morales, les niveaux de conformité peuvent être { `improper`, `neutral`, `congruent` }.

4.2 Construction de la confiance

Grâce aux images morales et éthiques, un agent peut décider d'accorder sa confiance à un autre ou non. La confiance peut être absolue (une confiance dans la conformité à une éthique du comportement de l'autre) ou relative à un ensemble de règles morales (confiance dans la prudence de l'autre, sa responsabilité, son obéissance à un ensemble de règles de conduite, etc.). Nous définissons deux actions épistémiques internes permettant d'évaluer la possibilité d'établir ces deux types de confiance.

Définition 7 (Fonction de confiance) *La* fonction de confiance éthique $TB^e_{a_i}$ *(resp.* fonction de confiance morale $TB^m_{a_i}$*) est définie comme :* $TB^e_{a_i} : \mathbb{A} \rightarrow \{\top, \bot\}$ *(resp.* $TB^m_{a_i} : \mathbb{A} \times 2^{\mathcal{MR}_{a_i}} \times MV_{a_i} \rightarrow \{\top, \bot\}$*)*

Ici, ces fonctions de confiance sont abstraites et doivent être instanciées. Par exemple, lorsqu'un agent a_i évalue la conformité du comportement d'un autre agent a_j au regard de CK_{a_i}, GK_{a_i} et RK_{a_i} (i.e. l'image éthique), la fonction de confiance éthique produit une croyance $\texttt{ethical\_trust}(a_j, a_i)$. De même, lorsque l'agent a_i évalue la conformité du comportement de a_j au regard de R (i.e. vérifie que la conformité morale de l'image de son comportement par rapport à R est au moins égale à mt), la fonction de confiance morale produit une croyance $\texttt{moral\_trust}(a_j, a_i, R, mt)$.

4.3 Éthique de la confiance

Les croyances sur l'image et la confiance peuvent devenir des éléments de contexte permettant d'exprimer la moralité ou l'éthique d'une action. Autrement dit, la moralité d'une action à l'égard d'un agent peut être conditionnée à la confiance ou l'image que l'agent juge a de l'autre.

Premièrement, la confiance éthique et morale peuvent enrichir la description des règles et valeurs morales. Par exemple, la valeur de *responsabilité* pourrait être supportée lorsque les actions de délégation ne sont confiées qu'à des agents de confiance. Ici, la responsabilité est définie comme la capacité à déléguer des actions sensibles uniquement à des agents appropriés.

Deuxièmement, des croyances spécifiques de confiance morale peuvent être employées comme des éléments de règle morale. Par exemple, étant donnée une valeur d'honnêteté et ses supports de valeur, un agent peut être doté d'une règle exprimant "Il est immoral de ne pas agir honnêtement à l'encontre de tout agent honnête.". Ici, "tout agent honnête" peut être modélisé par l'existence d'une croyance `moral_trust` associant à un agent une confiance morale dans la conformité de son comportement à l'ensemble R des règles définissant la moralité d'un comportement honnête.

Enfin, puisque évaluer et juger les autre constituent des actions, il est également possible d'exprimer et évaluer leur caractère moral ou éthique. Ainsi, la valeur morale de *tolérance* peut être supportée par la construction d'une image des autres avec un seuil peu élevé tant que les ensembles $EC_{a_j,[t_0,t]}$ ou $MC_{a_j,[t_0,t]}$ ne sont pas assez significatifs. Le choix du seuil, des pondérations et la conversion de l'agrégation en niveau de conformité peuvent également permettre de représenter diverses formes de confiance. Une valeur telle que l'*indulgence* peut être supportée par le fait d'accorder toujours une pondération plus faibles aux actions les moins récentes. Il est ainsi possible de décrire une morale de la confiance par l'emploi de règles comme "Il est immoral de construire la confiance sans tolérance ni indulgence" [21].

5 Preuve de concept

Cette section illustre la manière dont les éléments présentés dans la section précédente ont été implémentés dans un système multi-agent. Cette preuve de concept est implémentée à l'aide de la plate-forme JaCaMo [7] avec des agents BDI décrits en langage Jason partageant un environnement comportant des artefacts conformes aux standards de Cartago. Le code source complet est téléchargeable sur notre site [4]. L'environnement simule un marché financier sur lequel des actifs sont cotés et échangés par des agents. La Sec. 5.1 introduit le domaine de la gestion éthique d'actifs et les caractéristiques de notre application. Les morales et éthiques employés sont définies en Sec. 5.2. La construction des images et de la confiance est présentée en Sec. 5.3.

5.1 Modèle de marché financier

La gestion d'actifs financiers soulève bon nombre de problématiques éthiques et pratiques [5]. Ces décisions sont déléguées à des

4. https://cointe.users.greyc.fr/projects/ethical_market_simulator

5. http://sevenpillarsinstitute.org/

agents autonomes auxquels des utilisateurs humains délèguent les décisions d'achat et de vente, pouvant avoir des conséquences sur l'économie réelle [18]. Comme montré par [8], certains fonds d'investissement proposent une offre de gestion éthique et responsable de placements, et leur nombre ainsi que leur proportion sur les marchés tend à croître de manière significative ces dernières années. Toutefois, si les performances de ces fonds en matière de gain est objectivement mesurable, l'appréciation de la dimension éthique de leur comportement semble plus difficile et subjective car dépendante des convictions de l'observateur.

Dans cette preuve de concept, nous considérons un marché sur lequel les agents autonomes gestionnaires d'actifs peuvent échanger de la monnaie et des parts de capitaux. Chaque agent peut déposer des ordres d'achat et de vente qui seront exécutés lorsque le marché aura trouvé un autre ordre correspondant en terme de prix et de quantité. L'agent peut également annuler un ordre qui n'aurait pas encore été exécuté. Dans notre implémentation, le marché emploie une structure de *Central Limit Order Book* (CLOB) [2] pour conserver les ordres et faire correspondre l'offre et la demande.

En observant le marché, les agents peuvent percevoir les ordres en attente d'exécution et un ensemble d'indicateurs (prix et volume des derniers échanges effectués, moyenne et écart-type des prix au cours du temps, etc.).

Les agents disposent également d'informations sur le contenu de leur propre portefeuille d'actions. En raisonnant sur ces croyances comme une connaissance contextuelle CK, l'agent peut déduire ce qu'il peut vendre ou acheter à l'instant présent pour produire $\mathcal{A}_p$. En y ajoutant les informations dont il dispose sur l'évolution du marché, il est également capable de produire $\mathcal{A}_d$. Pour cela nous avons implémenté une stratégie simple basée sur des comparaisons de moyennes mobiles.

Trois types d'agents sont présents dans l'expérience : (1) des *agents aléatoires* assignés à des actifs cotés, passant des ordres au hasard en termes de prix et de volume afin de générer de l'activité et simuler le "bruit" d'un marché réel ; (2) des *agents sans éthique*, uniquement dotés d'une fonction d'évaluation de la désirabilité en guise de stratégie leur permettant de spéculer. La même fonction sera employée chez les agents éthiques pour générer $\mathcal{A}_d$; (3) des *agents éthiques* implémentant le jugement éthique comme processus décisionnel afin de se comporter conformément à leur éthique.

5.2 Paramétrage éthique

Afin d'informer leurs jugements et permettre de définir des contextes de règles morales, les agents éthiques disposent de connaissances sur les actifs. Par exemple, les certifications d'entreprise attestant des engagements pris envers l'environnement, ou encore leur secteur d'activité telle que la production d'énergie d'origine nucléaire.

De plus, nous avons ajouté à ces informations sur la situation une description de convictions morales directement inspirées de la littérature [6]. Les agents éthiques sont ainsi dotés d'un ensemble de valeurs et sous-valeurs hiérarchisées : par exemple *environmental reporting* est considéré comme une sous-valeur (au sens d'une spécificité plus forte) de la valeur *environment*.

Ces valeurs sont concrètement décrites par un ensemble de supports tes que "échanger des actifs de producteur d'énergie nucléaire n'est pas conforme à la sous-valeur *promotion of renewable energy*", "échanger des actifs d'une société labelisée FSC est conforme à la sous-valeur *environmental reporting*" ou "échanger des actifs de producteur d'énergie nucléaire est conforme à la sous-valeur *fight climate change*".

Les agents sont également dotés de règles morales pouvant employer des valeurs morales pour définir la moralité d'un comportement. Par exemple "Il est moral d'agir conformément à la valeur *environment*". Ces règles morales sont regroupées au sein d'ensembles sur lesquels pourront être agrégées les images.

À ce stade, un agent éthique est capable d'inférer par exemple que, au regard de ses connaissances sur le contexte CK et sur la morale GK, échanger des actifs d'une entreprise labellisée FSC est moral tandis qu'échanger des actifs d'une compagnie produisant de l'énergie nucléaire est à la fois moral et immoral. Pour déterminer s'il est juste d'échanger le second actif, l'agent aura besoin de sa théorie du juste. Nous avons ainsi doté ces agents de principes éthiques élaborés tels que l'éthique d'Aristote (inspirée de [20]) et de plus simples tels que "Un acte est juste s'il est possible, moral et désirable". Chaque agent peut avoir ainsi de nombreux principes éthiques et l'action juste sera celle qui satisfait le mieux les principes dans un ordre lexicographique.

6. http://www.ethicalconsumer.org/

5.3 Construction d'images et de confiance

À chaque action exécutée sur le marché, les agents reçoivent un message et réévaluent leurs images des agents impliqués dans la transaction. Comme mentionné dans la précédente section, évaluer la conformité de comportements, construire l'image et la confiance sont des actions. Elles sont donc décrites comme des plans Jason [7]. Dans la suite de cette section nous détaillons la construction de la confiance morale. La confiance éthique se construit de manière analogue.

Premièrement, un plan évalue la conformité d'une action avec chaque règle morale de l'ensemble R sur lequel porte l'image. Le nombre d'action morales ou immorales se trouve incrémenté à l'issue de chaque évaluation. Dans l'implémentation actuelle, nous utilisons une agrégation linéaire (c'est-à-dire associant la même pondération à chaque action). Ensuite le niveau de conformité est attribué en fonction de la proportion d'actions conformes afin de construire l'image. Dans cette expérimentation nous n'utilisons que trois niveaux de conformité (arbitrairement `neutral` pour un résultat compris dans $[0.4, 0.6[$, `improper` pour les résultats inférieurs et `congruent` pour les résultats supérieurs). Enfin, lorsque le niveau de conformité franchi le seuil de confiance, un plan met à jour la confiance dans l'agent jugé au regard des règles concernées.

5.4 Resultats

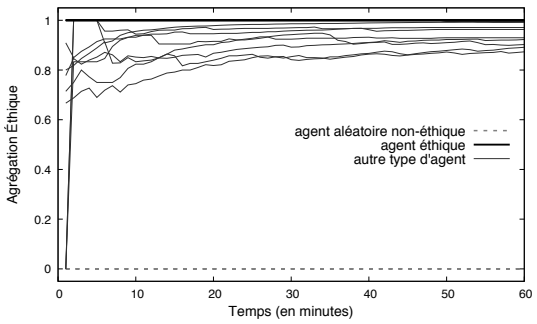


FIGURE 3 – Évolution des images des agents en sortie de la fonction d'agrégation éthique

La figure 3 montre l'évolution de l'agrégation des images éthiques en sortie de la fonction d'agrégation d'un agent éthique observant les autres agents du système (ici 20 agents aléatoires, 10 agents sans éthique et 3 agents éthiques). Remarquons premièrement qu'un agent peut construire une image d'un agent aléatoire ou d'un agent non-éthique. C'est l'une des propriétés de ce modèle : nous n'évaluons que la conformité d'un comportement observé au regard d'une éthique, sans nécessairement chercher à connaître l'intention des autres agents. Comme attendu, les agents ayant la même éthique restent durant toute l'observation à la valeur maximale de 1.0 (traits épais). À l'inverse, les agents aléatoires affectés à l'animation du cours d'un actif immoral aux yeux de l'agent juge restent à 0.0 (traits pointillés). Tous les autres agents convergent lentement vers une valeur intermédiaire dépendante de leurs investissements. En observant les croyances des agents au cours de l'expérience, il est possible de voir les agents éthiques construire les images des autres et établir une confiance lorsque cette image franchit les seuils de conformité.

6 Conclusion

Dans cet article nous avons montré comment un processus de jugement éthique peut être employé dans une architecture d'agent BDI et nous avons défini des mécanismes permettant de construire des images caractérisant la conformité d'un comportement du point de vue d'une éthique ou une morale. Nous avons ensuite montré la manière dont ces images peuvent être employées pour décider d'établir ou non une relation de confiance afin de coopérer. Une preuve de concept montre comment un tel modèle peut être implémenté dans une plate-forme BDI et utilisé dans le cadre de la gestion d'actifs financiers. D'un point de vue de la modélisation, ce travail répond au problème de l'évaluation de la proximité entre une éthique et un comportement, problème d'autant plus difficile dans les cas où l'éthique repose sur des convictions personnelles d'agents hétérogènes. Avec ce modèle, les agents peuvent savoir quel ensemble de règles en particulier ou quelle éthique en général sont concernées par cette proximité. Grâce à l'expressivité de ce modèle en matière d'éthique et de morale, nous envisageons de représenter des agents dotés d'un plus vaste ensemble de principes éthiques et de valeurs et règles morales prenant en compte dans leur description de l'image des autres, et décrire des notions telles que l'indulgence ou l'intransigeance.

Remerciements

Ce travail a été réalisé dans le cadre du projet EthicAa [7] (référence ANR-13-CORD-0006).

7. `http://ethicaa.org/`

Références

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *33th IEEE International Conference on Systems Sciences*, pages 1–9, 2000.

[2] I. Aldridge. *High-frequency trading : a practical guide to algorithmic strategies and trading systems*, volume 459. John Wiley and Sons, 2009.

[3] M. Anderson and S.L. Anderson. Toward ensuring ethical behavior from autonomous systems : a case-supported principle-based paradigm. *Industrial Robot*, 42(4) :324–331, 2014.

[4] R. Arkin. *Governing lethal behavior in autonomous robots*. CRC Press, 2009.

[5] K. Arkoudas, S. Bringsjord, and P. Bello. Toward ethical robots via mechanized deontic logic. In *AAAI Fall Symposium on Machine Ethics*, pages 17–23, 2005.

[6] F. Berreby, G. Bourgne, and J.-G. Ganascia. Modelling moral reasoning and ethical responsibility with logic programming. In *20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, pages 532–548, 2015.

[7] Olivier Boissier, Rafael H Bordini, Jomi F Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with jacamo. *Science of Computer Programming*, 78(6) :747–761, 2013.

[8] S. Bono, G. Bresin, F. Pezzolato, S. Ramelli, and F. Benseddik. Green, social and ethical funds in europe. Technical report, Vigeo, 2013.

[9] J. Carbo, J. Molina, and J. Davila. Comparing predictions of SPORAS vs. a fuzzy reputation agent system. In *3rd International Joint Conference on Fuzzy Sets and Fuzzy Systems*, pages 147–153, 2002.

[10] J. Carter, E. Bitting, and A. Ghorbani. Reputation formalization for an information-sharing multi-agent system. *Computational Intelligence*, 18(2) :515–534, 2002.

[11] C. Castelfranchi and R. Falcone. *Trust theory : A socio-cognitive and computational model*, volume 18. John Wiley & Sons, 2010.

[12] H. Coelho and A.C. da Rocha Costa. On the intelligence of moral agency. *Encontro Português de Inteligência Artificial*, pages 12–15, October 2009.

[13] H. Coelho, P. Trigo, and A.C. da Rocha Costa. On the operationality of moral-sense decision making. In *2nd Brazilian Workshop on Social Simulation*, pages 15–20, 2010.

[14] N. Cointe, G. Bonnet, and O. Boissier. Ethical judgment of agents' behaviors in multi-agent systems. In *15th International Conference on Autonomous Agents & Multiagent Systems*, pages 1106–1114, 2016.

[15] N. Cointe, G. Bonnet, and O. Boissier. Multi-agent based ethical asset management. In *1st Workshop on Ethics in the Design of Intelligent Agents*, pages 52–57, 2016.

[16] R. Conte and M. Paolucci. *Reputation in artificial societies : Social beliefs for social order*, volume 6. Springer Science & Business Media, 2002.

[17] B. Esfandiari and S. Chandrasekharan. On how agents make friends : Mechanisms for trust acquisition. In *4th Workshop on Deception, Fraud, and Trust in Agent Societies*, pages 27–34, 2001.

[18] Directorate-General for Economic and Financial Affairs. Impact of the current economic and financial crisis on potential output. Occasional Papers 49, European Commission, June 2009.

[19] J.-G. Ganascia. Ethical system formalization using non-monotonic logics. In *29th Annual Conference of the Cognitive Science Society*, pages 1013–1018, 2007.

[20] J.-G. Ganascia. Modelling ethical rules of lying with Answer Set Programming. *Ethics and Information Technology*, 9(1) :39–47, 2007.

[21] H.J.N Horsburgh. The ethics of trust. *The Philosophical Quartely*, 10(41) :343–354, 1960.

[22] R. Johnson. Kant's moral philosophy. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Summer edition, 2014.

[23] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service proposition. *Decision Support Systems*, 43(2) :618–644, 2007.

[24] E. Lorini. On the logical foundations of moral agency. In *11th International Conference on Deontic Logic in Computer Science*, pages 108–122, 2012.

[25] S. Marsh. *Formalising Trust as a Computational Concept*. PhD thesis, University of Stirling, Stirling, 1994.

[26] A. McIntyre. Doctrine of double effect. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Winter edition, 2014.

[27] G. Muller, L. Vercouter, and O. Boissier. Towards a general definition of trust and its application to openness in MAS. In *6th Workshop on Deception, Fraud and Trust in Agent Societies*, pages 49–56, 2003.

[28] P. Ricoeur. *Oneself as another*. University of Chicago Press, 1995.

[29] A. Rocha-Costa. Moral systems of agent societies : Some elements for their analysis and design. In *1st Workshop on Ethics in the Design of Intelligent Agents*, pages 32–37, 2016.

[30] J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence*, 24(1) :33–60, 2005.

[31] Jordi Sabater-Mir and Laurent Vercouter. Trust and reputation in multiagent systems. *Multiagent Systems*, page 381, 2013.

[32] A. Saptawijaya and L. Moniz Pereira. Towards modeling morality computationally with logic programming. In *Practical Aspects of Declarative Languages*, pages 104–119. 2014.

[33] M. Timmons. *Moral theory : an introduction*. Rowman & Littlefield Publishers, 2012.

[34] L. Vercouter and G. Muller. L.I.A.R. : Achieving social control in open and decentralized multiagent systems. *Applied Artificial Intelligence*, 24(8) :723–768, 2010.

[35] B. Yu and M.P. Singh. Distributed reputation management for electronic commerce. *Computational Intelligence*, 18(4) :535–549, 2002.

Que valent les stratégies probabilistes au dilemme itéré des prisonniers ?

J.P. Delahaye[a] jean-paul.delahaye@univ-lille.fr
P. Mathieu[a] philippe.mathieu@univ-lille.fr

[a]CRIStAL Lab, UMR 9189 CNRS, Université des Sciences et Technologies de Lille, France

Résumé

Nous menons une étude expérimentale minutieuse sur les stratégies probabilistes au dilemme des prisonniers. Nous utilisons pour cela la méthode des classes complètes associée à une approche évolutionniste. Les résultats que nous obtenons ont donc un caractère objectif et dépendent le moins possible des ensembles de stratégies mis en compétition. Les ensembles étudiés sont grands (plusieurs milliers de stratégies), homogènes, et systématiques. Nous testons la robustesse de nos résultats par diverses méthodes. Les stratégies les meilleures repérées sont pour certaines d'entre elles nouvelles en ce sens qu'elles n'ont jamais été identifiées clairement par des études antérieures, et cela malgré leur simplicité. Nous identifions un critère jusque là inconnu qui conduit à une bonne anticipation de leur comportement dans des univers variés. Nous confrontons les résultats de cette étude avec ceux obtenus par les approches mathématiques de Press et Dyson. Nous confrontons aussi les nouvelles stratégies avec les meilleures stratégies connues.

Mots-clés : *Theorie des jeux, Dilemme du prisonnier, stratégies d'agents, comportement*

Abstract

We conduct a thorough experimental study of probabilistic strategies to the prisoner's dilemma. To do this, we use the complete class method associated with an evolutionary approach. The results we obtain are therefore objective in nature and depend as little as possible on the sets of strategies put in competition. The studied sets are large (several thousand strategies), homogeneous, and systematic. We test the robustness of our results by various methods. The best strategies identified are for some of them new in the sense that they have never been clearly identified by previous studies, despite their simplicity. We propose a criterion that leads to a good anticipation of their behavior in various contexts. We compare the results of this study with those obtained by the mathematical approaches of Press and Dyson. We also confront the new strategies with the best known strategies.

Keywords: *Game Theory, Iterated prisoner's Dilemma, Agent's Strategy, Behaviour*

1 Introduction

Suite à la parution de l'article de Press et Dyson [21] sur ce qu'ils ont appelé les stratégies ZD et l'extorsion, et suite aux réactions parfois critiques [1, 2, 10, 12, 11, 16, 19, 25] concernant leurs conclusions, l'attention a été portée sur les stratégies probabilistes au dilemme itéré des prisonniers [24, 8, 9, 11]. Pourtant aucune méthode de tri systématique et aussi exhaustive que possible n'a été utilisée pour savoir si des stratégies probabilistes simples égalaient ou surpassaient les meilleures stratégies connues [4, 18]. Nous menons ici une telle étude en combinant deux des méthodes que nous considérons comme les plus susceptibles de produire des résultats robustes et dépourvus de subjectivité : la méthode des compétitions évolutionnaires parfois appelée écologiques [3, 5, 4, 8, 23], et la méthode des classes complètes [6, 8, 17]. Nous utilisons en particulier des classes complètes homogènes composées de stratégies aléatoires.

Nos résultats montrent que des stratégies simples et pourtant inconnues jusqu'à présent émergent parmi les milliers de stratégies mises en compétition. Une catégorie assez large de stratégies est identifiée comme robuste et performante pour les compétitions évolutives. Un paramètre noté p' est identifié et interprété ; il est corrélé à la réussite des stratégies et semble donc fournir un critère efficace pour prévoir ce que donnera une stratégie probabiliste dans une compétition évolutive. Des variantes plus complexes, mais plus fines de ce paramètre sont recherchées par une méthode d'exploration statistique exhaustive.

Des séries systématiques de tests sont menées pour s'assurer de la robustesse des résultats ob-

tenus. En particulier nous plongeons les nouvelles stratégies probabilistes identifiées dans des environnements de stratégies déterministes pour nous assurer qu'elles restent performantes en dehors du contexte qui a permis de les découvrir. Les nouvelles venues sont aussi confrontées aux stratégies repérées par Press et Dyson et aux stratégies les meilleures connues pour le dilemme itéré des prisonniers. Cela nous conduit à une nouvelle formulation des conclusions expérimentales générales sur les stratégies optimales connues au dilemme des prisonniers.

2 Définitions et rappels

Le dilemme des prisonniers [3, 22, 23, 13] est celui auquel sont soumis deux entités ayant le choix entre coopérer (`c`) ou trahir (`d` pour "Defect") et qui sont rétribuées par `R` points si chacune joue `c`, par `P` points si chacune joue `d`, et reçoivent respectivement `T` et `S` points si l'une joue `d` et l'autre `c`. On décrit ces règles avec les notations suivantes : `[c,c]`→`R+R`, `[d,d]`→`P+P`, `[d,c]`→`T+S`.

Pour que la situation soit celle d'un dilemme, on impose [3] : `T>R>P>S` et `T+S<2R`. On choisit le plus souvent les valeurs `T=5`, `R=3`, `P=1`, `S=0`.

Le dilemme est itéré quand on imagine que la situation de choix entre `c` et `d`, se présente périodiquement aux deux mêmes entités. Jouer consiste alors à choisir une stratégie qui, informée du passé (donc du comportement antérieur de l'adversaire et de son propre comportement) indique comment jouer le coup suivant.

Une multitude d'études [6, 4, 13, 14, 20, 26], conduisent aux conclusions suivantes sur lesquelles un accord général semblait établi.

(a) Il n'y a pas de stratégie meilleure que toutes les autres, mais certaines sont mauvaises dans pratiquement tous des environnements possibles, alors que d'autres sont efficaces et réussissent bien (gagnent beaucoup de points) dans des tournois variés. (b) Les bonnes stratégies sont les stratégies réactives (qui répondent quand on les trahit), prennent le risque de coopérer (elles commencent par coopérer et face à un adversaire qui coopère, elles ne tentent pas de trahir), et savent être indulgentes (après une trahison de l'adversaire elles finissent par pardonner pour renouer la coopération. C'est le cas de `gradual` (voir définition section 6).

2.1 Simulation de l'évolution

À côté de l'évaluation des stratégies obtenues en organisant des tournois variés, il existe des méthodes de tests simulant des évolutions auxquelles ne réussissent que les stratégies robustes.

Les résultats obtenus par ces calculs modélisent la sélection naturelle. Ils confirment souvent (mais pas toujours comme on va le voir) ceux des tournois et en accroissent les contrastes. Ils conduisent de plus à une conclusion surprenante : sauf dans de très rares cas, l'arène finit par n'être occupée que par des stratégies qui ne prennent jamais l'initiative de trahir (c'est le cas de `tit_for_tat` de `gradual` ou de `pavlov`). Au bout de quelques générations, l'arène est donc occupée par des stratégies qui ne jouent entre elles que des coups `[c,c]`. L'arène se trouve donc dans un état de coopération généralisée.

2.2 Les classes complètes

Pour mener des tests objectifs qui ne dépendent pas des stratégies repérées bonnes ou robustes, et pour se donner des chances de découvrir de nouvelles stratégies performantes, nous utilisons la *méthode des classes complètes* [5, 8, 18] qui consiste à regrouper systématiquement toutes les stratégies ayant des capacités équivalentes ou fonctionnant selon un principe abstrait fixé. Nous considérons en particuliers les classes `Mem(X,Y)` qui regroupent toutes les stratégies dont le coup n dépend de manière déterministe des `X` derniers coups que la stratégie a joués et des `Y` derniers coups que l'adversaire a joués. Tant qu'il n'y a pas `max(X,Y)` coups joués, la stratégie utilise une amorce fixée une fois pour toutes. Une stratégie de `Mem(1,2)` se définit donc par les deux premiers coups qu'elle joue, puis par ce qu'elle fait quand le passé est par exemple `[d, dc]` (elle a joué `d` au coup $n-1$, et l'adversaire a joué `d` au coup $n-2$, et `c` au coup $n-1$. Dans ce cas il y a 8 passés possibles). Nous notons une telle stratégie par un nom du type `mem12_cdCDCCDDCC` avec la convention que la suite désigne les 2 premiers coups puis les répliques pour les 8 passés possibles pris dans l'ordre lexicographique `[c cc] [c cd] [c dc] [c dd] [d cc] [d cd] [d dc] [d dd]`. Le nombre de stratégies de `Mem(1,2)` est $1024 = 2^{10}$ (il y a dix paramètres binaires à fixer pour chacune) et plus généralement le nombre de stratégies de `Mem(X,Y)` est $2^{max(X,Y)}.2^{2^{(X+Y)}}$.

2.3 Une sélection de 21 stratégies

Dans la suite de l'article nous utiliserons entre autres l'ensemble `Select` de 21 stratégies issues de [18] qui peut être considéré à la fois comme contenant les stratégies les plus simples et les stratégies les meilleures identifiées aujourd'hui (voir section 6). On trouvera Figure 1 le résultat de leur confrontation évolutionnaire. Trouver des stratégies qui se classent bien quand on les ajoute à E est un défi souvent difficile.

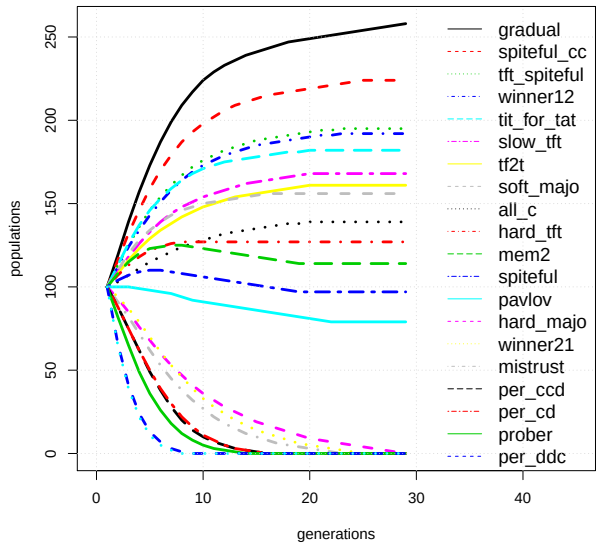


FIGURE 1 – Compétition évolutionnaire des 21 stratégies contenues dans `Select`

3 Les résultats de Press et Dyson

3.1 Une percée théorique

L'article de William Press et Freeman Dyson [21] possède un titre provocateur : *"Iterated Prisoner's Dilemma contains strategies that dominate any evolutionary opponent"* ("Dans le dilemme itéré des prisonniers, il existe des stratégies qui dominent tout adversaire évolutif"). L'observation quasi universelle d'une convergence vers la coopération généralisée semblait interdire qu'il existe des stratégies dominantes donc exploitant les autres. Les affirmations proposées par [21] proviennent de raisonnements et de calculs mathématiques alors que dans ce domaine, il est difficile d'obtenir des résultats démontrés puisque l'espace associé au problème est infini et discret et n'est muni d'aucune topologie naturelle.

Une question simple se pose naturellement à laquelle les arguments mathématiques ne répondent pas : parmi un ensemble aussi neutre que possible de stratégies probabilistes soumis à un processus évolutif, quelles stratégies probabilistes s'imposent ? C'est la question que nous traitons ici.

Nos conclusions rejoignent d'autres conclusions déjà obtenues depuis la parution de l'article de Press et Dyson [1, 2, 10, 12, 11, 16, 19, 25] mais grâce à notre méthode systématique, nous parvenons ici à mettre en avant une série de stratégies robustes et efficaces qui n'avait pas été extraites des résultats expérimentaux antérieurs et nous montrons qu'un critère simple permet de les identifier rapidement.

3.2 Stratégies probabilistes à mémoire d'un coup

Le travail de Press et Dyson propose deux théorèmes. Le premier théorème concerne le dilemme itéré dans une version limitée aux stratégies probabilistes à mémoire d'un coup : les stratégies `lunatique`, `tit_for_tat` et `pavlov` appartiennent à cette catégorie, mais pas la stratégie `gradual` qui pour décider ce qu'elle fait consulte le passé de tous les coups déjà joués.

Une stratégie à mémoire d'un coup est définie par 4 paramètres `p1`, `p2`, `p3`, `p4` qui indiquent la probabilité de jouer `c` lorsque le dernier coup a été `[c c]` ou `[c d]` ou `[d c]` ou `[d d]`. Notons `proba(p1,p2,p3,p4)` cette stratégie générale. On ne précise pas comment est jouée le premier coup, mais c'est sans importance pour le résultat mathématique qui n'en dépend pas. En pratique pour les simulations, on considérera aussi bien les stratégies dont le premiers coup est `c` que celles dont le premier coup est `d`.

Press et Dyson considèrent une classe particulière des stratégies `proba(p1,p2,p3,p4)` dépendant de trois paramètres `a`, `b` et `c` et dénommées ZD. Nous les noterons `ZD(a,b,c)`.

3.3 Les stratégies ZD.

Les équations générales reliant les paramètres `p1`, `p2`, `p3`, `p4` pour les ZD dans le cas `R=3`, `S=0`, `T=5`, `P=1` sont :

```
p1=1+3a+3b+c        p3=5a+c
p2=1+5b+c           p4=a+b+c
```

Press et Dyson démontrent que lorsqu'on oppose une stratégie ZD(a,b,c) à une stratégie probabiliste à mémoire d'un coup proba(p1,p2,p3,p4), et que l'on note G1 le gain moyen par coup de la première et G2 le gain moyen par coup de la seconde, alors ces gains moyens vérifient : aG1+bG2+c=0. Les deux stratégies ont des gains liés linéairement l'un à l'autre.

Lorsqu'elles se rencontrent, proba(p1,p2,p3,p4) est contrôlée par ZD(a,b,c).

3.4 Les stratégies égaliseurs

Lorsque a=0 et que b≠0 alors G2=-c/b, autrement dit une stratégie probabiliste quelconque a un gain moyen indépendant des probabilités qui la définissent, gain qui ne dépend que de la stratégie ZD(a,b,c) qui lui fait face. Une telle stratégie ZD est nommée égaliseur. Les relations deviennent :

```
p1=3b+c+1          p3=c
p2=5b+c+1          p4=b+c
```

et donc G2=-c/b.

Contre une telle stratégie, toutes les stratégies probabilistes à mémoire d'un coup obtiennent le même gain moyen qui est connu d'avance : -c/b. Inutile de se débattre face à une stratégie égalisateur vous gagnerez -c/b et pas plus ! Les valeurs possibles pour -c/b sont toutes les valeurs situées entre P et R. Voici les résultats des rencontres entre quelques stratégies connues et un égaliseur equa qui est ZD(0,-1/3,2/3) qui est équivalent à un proba(2/3,0,2/3,1/3), donc G2=-c/b=2. Il force donc son adversaire à avoir un gain moyen de 2.

```
equa = 2,5 vs tit_for_tat = 2
equa = 3 vs gradual = 2
equa = 3 vs all_c = 2
equa = 1 vs all_d = 2
```

Comme on le voit, equa force le gain moyen de l'adversaire, mais cela se fait parfois à ses dépens, elle en sera victime lorsqu'elle jouera contre elle-même !

3.5 Les stratégies extorqueurs

Parmi les stratégies ZD découvertes par Press et Dyson certaines opèrent une forme d'extorsion. En effet, si c=-(a+b)P (donc a+b+c=0 avec P=1) on démontre que le gain moyen G1 de la stratégie ZD contre une autre (obtenant le gain moyen de G2) vérifie G1-P=X.(G2-P) avec X=-b/a.

En clair, si la seconde veut gagner plus, donc augmenter (G2-P), cela entraîne mécaniquement que la stratégie ZD augmente son gain moyen, dont l'écart à P est toujours X fois l'écart à P du gain moyen de la seconde.

Les quatre paramètres définissant ces stratégies extorqueurs sont donnés par les équations :

```
p1=2a+2b+1          p3=4a-b
p2=4b-a+1           p4=0
```

Un des gros défauts des stratégies de type extorqueur est que si X>1 alors elles jouent mal contre elles-mêmes. Vouloir par exemple gagner deux fois plus contre son adversaire (par rapport à P) entraîne que face à elle-même elles ne gagneront que P, ce qui est moins bien que C.

Quand X>1, les extorqueurs ne vous permettent d'avoir un bon résultat que si vous leur en accordez un proportionnellement meilleur : G1-P=X.(G2-P). Les extorqueurs sont donc des variantes de la stratégie all_d : personne ne peut les battre, mais cela a pour conséquence qu'elles prennent le risque de gagner peu.

On notera que les résultats concernant les stratégies égaliseurs avaient déjà été présentées dans [7] avant [21].

3.6 Utilité d'une longue mémoire

Le second théorème important de l'article de Press et Dyson indique que si, dans une partie supposée infinie, la stratégie A est face à une stratégie B ayant une mémoire de k coups, il existe alors une stratégie A' qui obtient le même score moyen face à B et qui n'a qu'une mémoire de k coups. La combinaison des deux résultats mathématiques de Press et Dyson conduit à l'affirmation que face à une stratégie égaliseur ou extorqueur ce ne sont pas seulement toutes les stratégies à mémoire un coup qui se trouvent contraintes mais toutes les stratégies à mémoire finie. De là on est tenté de conclure que : *"(a) les stratégies mémorisant plus que le dernier coup sont inutiles. (b) On dispose avec les stratégies ZD de stratégies dominantes dans l'absolu pour le dilemme itéré des prisonniers"*.

Certains ont d'ailleurs interprété ainsi les théorèmes démontrés, et le titre retenu pour leur article suggère que c'est le cas de Press et

Dyson. Pourtant la double affirmation sur l'inutilité des stratégies à mémoire étendue et sur la dominance absolue des stratégies ZD est fausse. Considérons d'abord l'affirmation de dominance des stratégies ZD. On sait depuis longtemps qu'au dilemme des prisonniers itéré battre son adversaire (obtenir plus de points que lui) peut se faire aux dépens de celui qui gagne et que celui-ci aurait pu obtenir plus de points en moyenne en acceptant d'être battu. La stratégie `all_d` bat tout autre stratégie (c'est une évidence) et par exemple sur une partie de 100 coups contre `tit_for_tat` obtient 104 points alors que `tit_for_tat` en gagne 99. La stratégie `all_c` ne bat pas `tit_for_tat` et gagne 300 points en 100 coups contre `tit_for_tat` qui gagne aussi 300 points. Contre `tit_for_tat`, `all_d` gagne peut-être, mais elle a tort de gagner car en faisant comme `all_c`, elle obtiendrait un bien meilleur score.

La plupart des stratégies ZD sont dans la même situation : elles ne gagnent contre leur adversaire qu'en renonçant elles-mêmes à avoir de bons scores. En un sens, ce que propose [21] est un ensemble de stratégies généralisant `all_d`. Outre l'erreur de croire que battre son adversaire c'est gagner des points, un autre oubli a conduit à croire que les stratégies ZD étaient dominantes : pour s'imposer il faut jouer correctement contre soi-même. C'est important dans les tournois, mais plus encore dans les compétitions évolutives. En effet, si vous l'emportez lors des premières générations, l'arène se peuple de stratégies identiques à vous, et vous allez donc très fréquemment les rencontrer. Si vous jouez mal face à vous-même, cela se retourne à terme contre vous. Rien n'est faux dans les résultats mathématiques de Press et Dyson, mais en n'abordant que le problème *"qui gagne dans un combat un contre un ?"* et en oubliant le problème *"combien de points sont gagnés au total ?"* et le problème *"face à toi-même te fais-tu du tort ?"* les théorèmes démontrés ne permettent pas de conclure que les stratégies ZD sont de bonnes stratégies au sens évolutionnaire. Nos expériences de simulations montrent que les stratégies ZD sont mauvaises !

Le résultat de Press et Dyson sur l'inutilité pour une stratégie d'avoir de la mémoire est juste : Si A est face à une stratégie B ayant une mémoire de k coups, il existe une stratégie A' qui obtient le même score moyen face à B et qui n'a qu'une mémoire de k coups. Cependant cela ne signifie pas que face à deux stratégies B et C ayant une mémoire de k coups, il existe une stratégie A' à mémoire de k coups qui fasse le même score face à B et face à C. En effet, celle A' qui peut remplacer A face à B, n'est pas nécessairement la même que celle, A", qui face à C peut remplacer A.

À nouveau les simulations confirment que les bonnes stratégies pour des environnements comportant plusieurs stratégies tirent un avantage de l'utilisation d'une large mémoire du passé. Sur les questions de la mémoire utile ou non on pourra consulter [15, 20].

4 Stratégies probabilistes à mémoire de un coup

Dans une première série d'expériences nous considérons des ensembles aussi grands que possible et aussi homogènes que possible de stratégies probabilistes à mémoire d'un coup, et nous les soumettons à un processus évolutionnaire.

4.1 Expériences évolutionnistes massives

Pour obtenir des ensembles de stratégies probabilistes de la forme `proba(p1,p2,p3,p4)`, nous fixons un pas de variation des paramètres probabilistes. Pour `K=5` par exemple nous faisons varier les `pi` dans l'ensemble fini $0, \frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}, 1$ ce qui donne $2 * 6^4 = 2592$ stratégies (le 2 provient des deux choix possibles pour le coup initial). Nous notons cette classe complète `ProbaCD_K=5` dont on voit l'évolution Figure 2.

Les résultats obtenus pour le tournoi sont indiqués ci-dessous. Nous avons utilisé des parties de 1000 coups et les paramètres usuels. Nous avons calculé les résultats de chaque partie un contre un en la faisant jouer 5 fois, cela de manière à limiter les effets des variations probabilistes.

```
1   probaD_0.0_0.0_0.0_0.2  39345809
2   probaC_0.0_0.0_0.0_0.2  39264669
3   probaD_0.2_0.0_0.0_0.2  39244163
4   probaD_0.0_0.2_0.0_0.2  39193409
5   probaC_0.2_0.0_0.0_0.2  39145486
6   probaD_0.4_0.0_0.0_0.2  39136783
7   probaC_0.0_0.2_0.0_0.2  39104825
8   probaD_0.2_0.2_0.0_0.2  39069164
9   probaC_0.4_0.0_0.0_0.2  39043576
10  probaD_0.6_0.0_0.0_0.2  38995942
```

La stratégie classée première est aussi notée `probaD(0,0,0,1/5)` ; le `D` indique que son premier coup est `D` (defect) ; l'entier en bout de

ligne indique le gain en points lors du tournoi répété 5 fois.

La première stratégie ZD est 34ème, il s'agit de la stratégie équivalente à `spiteful`. Dans les 100 premières, il y a que six stratégies ZD. Sans surprise, les ZD, les extorqueurs et les égaliseurs ne réussissent pas particulièrement bien en tournoi.

Pour la compétition évolutionnaire, les dix premières stratégies avec leurs effectifs finaux sont données ici :

```
1   probaC_1.0_0.8_0.0_0.0   34262
2   probaC_1.0_0.6_0.0_0.0   31579
3   probaC_1.0_0.4_0.0_0.0   30550
4   probaC_1.0_0.2_0.0_0.0   28640
5   probaC_1.0_0.0_0.0_0.0   27746
6   probaC_1.0_0.0_0.0_0.2    9540
7   probaC_1.0_0.2_0.0_0.2    8893
8   probaC_1.0_0.0_0.2_0.0    8451
9   probaC_1.0_0.2_0.2_0.0    7701
10  probaC_1.0_0.4_0.2_0.0    5984
```

On notera que la stratégie classée cinquième correspond à `spiteful`

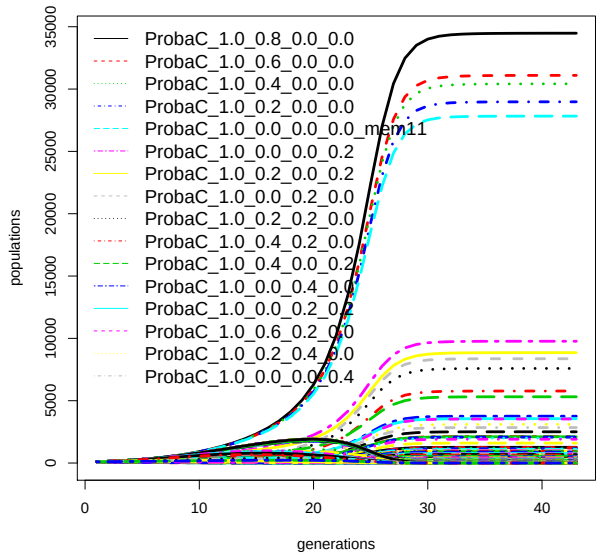


FIGURE 2 – Evolution évolutionnaire des `ProbaCD_K=5`

Remarque.

Le mécanisme évolutif programmé consiste à remplacer chaque génération par une nouvelle génération dont les effectifs pour un type de stratégies donné sont proportionnels au nombre de points gagnés par les stratégies de ce type lors d'un tournoi général impliquant toutes les stratégies présentes à la génération précédente : la descendance d'un type de stratégies à la génération n est proportionnelle aux points gagnés par les stratégies de ce type dans ne tournoi entre stratégies de la génération $n-1$. Au départ, on considère que chaque effectif est 100 et on impose que l'effectif total d'une génération à l'autre reste le même (aux problèmes d'arrondis près). Nous avons aussi mené des tests quand l'effectif d'une stratégie à la génération n est obtenu en prenant a fois l'effectif de la stratégie à la génération $n-1$, et $(1-a)$ fois l'effectif donné par le calcul précédent, avec le paramètre a compris entre 0 et 1. Cela revient à considérer que le remplacement d'une génération par la suivante n'est que partiel : les parents ne meurent pas tout de suite ! Les résultats (classement et effectifs finaux) obtenus sont toujours très proches (si $a < 1$) de ceux qu'on obtient quand le remplacement d'une génération par une autre est total, seule la durée de convergence vers l'état de coopération généralisée s'allonge.

Comme c'est assez souvent le cas au dilemme, les résultats d'un tournoi dans une classe complète qui comporte beaucoup de stratégies médiocres ou mauvaises ne reflètent pas du tout ce qu'on trouve comme résultat de compétitions évolutionnaires. La raison en est simple : les stratégies qui profitent de la présence de mauvaises stratégies dans la soupe initiale, sont rapidement rétrogradées, voire éliminées lorsque les mauvaises disparaissent. Pour réussir dans un processus évolutif, il faut obtenir de bons résultats avec ceux qui obtiennent de bons résultats et qui sont les seuls sur le long terme à survivre. Le résultat d'un tournoi dans une soupe comportant beaucoup de stratégies médiocres n'a guère de sens, seul celui de compétitions évolutives est pertinent.

La gagnante du tournoi est la variante de `all_d` consistant à remplacer le 4è paramètre `0.0` en `0.2` (`all_d` elle-même est classée 42ème de ce tournoi). Pour être bien classé au tournoi de la classe complète `probaCD_K=5`, il suffit d'exploiter les stratégies médiocres qui sont très nombreuses ; c'est très facile : il suffit de ne presque jamais coopérer. Cela ne nous apprend rien. Seule le résultat d'un processus évolutif qui commence par faire disparaître les stratégies médiocres présente de l'intérêt.

Dans le cas de la compétition évolutionnaire, ce qu'on découvre est remarquable et n'avait semble-t-il jamais été noté. La meilleure stratégie de notre large ensemble

de plus de deux mille stratégies est : probaC_1.0_0.8_0.0_0.0. L'effectif initial de 100 est passé à 34262 quand l'état de coopération généralisé s'est installé.

Son comportement est d'une grande simplicité : c'est une rancunière (une fois qu'elle a commencé à trahir, elle trahit toujours puisque p3 et p4 valent 0), mais c'est une rancunière qui réagit sans se précipiter quand on la trahit : en cas de coup [c, d], elle ne se met à trahir qu'avec une probabilité de 20%. Autrement dit, dans la phase initiale du jeu, elle coopère et continue de le faire aussi longtemps que l'autre coopère, et lorsqu'elle est trahie dans cette phase initiale, elle pardonne dans 80% des cas. En revanche lorsqu'elle se décide à trahir, il n'y a plus de retour possible à la coopération. Nous nommerons ce type des stratégies des *rancunières patientes*.

Les quatre stratégies suivantes sont elles aussi des rancunières patientes, mais dont la réactivité en cas de trahison augmente : d'abord 40% pour la seconde, puis 60% pour la troisième, puis 80% pour la quatrième, puis 100% pour la cinquième, ce qui donne la rancunière habituelle (spiteful).

La sixième stratégie possède encore un comportement qui s'interprète assez facilement. C'est une rancunière (sans aucune patience puisque p2=0), mais qui une fois dans sa phase de punition mène des tentatives de réconciliation : lorsque le coup qui vient d'être joué est [d, d], elle coopère dans 20% des cas, comme si elle disait à son adversaire : *"nous sommes mal partis, je tente un premier pas (dans 20% des cas) pour renouer une meilleure entente"*.

Les suivantes sont, elles encore, susceptibles d'interprétations du même type, bien que de plus en plus compliquées. Nous proposons de nommer *rancunières conciliantes* toutes les stratégies de ce type. Pour être précis et mener des décomptes nous nommerons rancunières conciliantes toutes les stratégies probaC(p1,p2,p3,p4) avec p1=1 et p2+p3+p4 $\leq$ 1. On notera que tit-for-tat et Pavlov sont de ce type. Les 37 premières stratégies dans la composition finale de la soupe après stabilisation de la compétition évolutionnaire appartiennent à cette catégorie.

Une autre remarque s'impose : dans la compétition évolutionnaire, seules 133 stratégies gardent des effectifs non nuls, et ce sont toutes des stratégies qui commencent par coopérer et qui vérifie p1=1. La soupe a donc incontestablement convergé vers un état de coopération généralisé.

Les seules ZD qui survivent sont les suivantes. Elles sont indiquées avec leur classement et leur catégorie :

```
29   probaC_1.0_0.0_1.0_0.0_ZD_Extorq
34   probaC_1.0_0.6_0.4_0.0_ZD_Extorq
35   probaC_1.0_0.4_0.6_0.0_ZD_Extorq
72   probaC_1.0_0.2_1.0_0.4_ZD
125  probaC_1.0_0.6_0.6_0.4_ZD_Equa
```

La première ZD qui est donc 29è est en fait tit_for_tat qui est effectivement une ZD de coefficient X=1. Ce X=1 signifie qu'en réalité elle n'extorque rien, mais oblige ses adversaires à gagner autant qu'elle, ni plus, ni moins. Ce type de stratégies a parfois été appelée "generous extorquer" [24] et leur capacité à survivre dans une compétition évolutive a été identifiée bien supérieure à celles ayant un X>1. Ce que nous trouvons confirme bien que ce type d'extorqueurs a une certaine capacité à survivre, mais ce que nous observons est que ce ne sont ni les seules, ni les meilleures !

La seconde ZD qui est 34è est une ZD avec a=2/25, b=-2/25, c=0, X=1. C'est une ZD de coefficient X=1 (qui comme tit_for_tat n'extorque donc rien)

La troisième ZD qui est 35è est une ZD avec a=3/25, b=-3/25, c=0, X=1. C'est encore une ZD de coefficient X=1.

La quatrième ZD, 72è du classement, n'est ni extorqueur, ni égaliseur. C'est une ZD avec a=2/25, b=-7/25, c=3/5.

La relation liant le gain moyen qu'elle obtient G1 et le gain moyen de son adversaire G2 est : aG1+bG2+c=0. Ce qui donne : 2G1+15=7G2. La stratégie dans la situation de coopération généralisée gagne 3 en moyenne par coup comme son adversaire.

La cinquième ZD, 125è du classement, est de type égaliseur avec a=0, b=-1/5, c=3/5, -c/b=3. C'est une stratégie égaliseur qui oblige son adversaire à gagner 3 points en moyenne par coup, ce qui est le cas pour elle en cas de coopération généralisée.

4.2 Reconnaître les stratégies efficaces ?

On le voit les seules extorqueurs ou égaliseurs qui survivent sont en réalité des stratégies qui n'extorquent rien au sens strict puisqu'elles se contentent de gagner comme leur adversaire. Il

est remarquable qu'elles soient largement battues par les rancunières patientes et des rancunières conciliantes.

Comme le montre la figure 3, le classement des 133 stratégies dont l'effectif ne s'annule pas est très directement corrélé au paramètre p'=p2+p3+p4.

Pour anticiper la réussite d'une stratégie probabiliste à un coup de mémoire, le double critère p1=1 et p'=p2+p3+p4 aussi petit que possible est très efficace.

Nous proposons maintenant une interprétation et une explication de ce double critère. Pour gagner une compétition évolutive ou seulement y réussir correctement, il faut survivre lorsque la coopération généralisée s'installe, il faut donc coopérer avec les stratégies qui coopèrent, d'où le p1=1. Il faut aussi être réactif, c'est-à-dire ne pas se laisser faire et adopter un comportement suffisamment sévère pour inciter l'autre à coopérer. La forme de réactivité la plus dure est celle de rancunière p2=p3=p4=0. Tempérer cette dureté est acceptable, si c'est modérément, et s'interprète de la manière suivante : (a) choisir une valeur non nulle pas trop grande de p2 revient à ne pas passer systématiquement dans l'état de rétorsion après un coup [c, d], mais n'y passer qu'avec une certain probabilité ; (b) choisir une valeur non nulle pas trop grande de p3 revient à accepter avec une certain probabilité après un coup [d c] de réessayer de coopérer avec un adversaire qui semble le désirer ; (c) choisir une valeur non nulle pas trop grande de p3 revient à accepter avec une certain probabilité après un coup [d d] de faire le premier pas dans le but de faire renaître un état de coopération mutuelle.

Combiner ces trois formes de tempérance dans une stratégie en adoptant des petites valeurs non nulles de p2 , p3 et p4 n'est pas absurde à condition que le total de la tempérance introduite dans son comportement ne soit pas trop grand, d'où le critère sur p'=p2+p3+p4.

Dans la figure 4 les stratégies de ProbaCD_K=5 ont été groupés en 7 sous-ensembles dont on a calculé le classement moyen génération par génération. Il y a les stratégies pour lesquelles p1≠1, puis pour celles avec p1=1, celles avec p' dans l'intervalle [0;0,5[puis [0,5;1[, [1;1,5[, [1,5;2[, [2;2,5[, [2,5;3].

Amélioration et variante de p'. Si nous calculons le coefficient de corrélation de Spearman entre le rang de classement des stratégies qui terminent avec des effectifs non nuls dans la compétition évolutionnaire ProbaCD_K=5 (il y en a 133) et le paramètre p', nous trouvons que Cor(rang,p')=0,8805

ce qui est une très bonne corrélation. Nous avons cherché systématiquement à améliorer ce paramètre. Le paramètre suivant, encore très simple donne un remarquable résultat. p"=p2+p3/2+p4 Cor(rang,p")=0,9411231. L'optimal, obtenu par analyse canonique des corrélations (CCA) permet encore une légère amélioration : p⋆ = 0,266 p2+0,138 p3+0,277 p4 Cor(rang,p⋆)= 0,9413768. Les figures 3 et 4 illustrent cette corrélation.

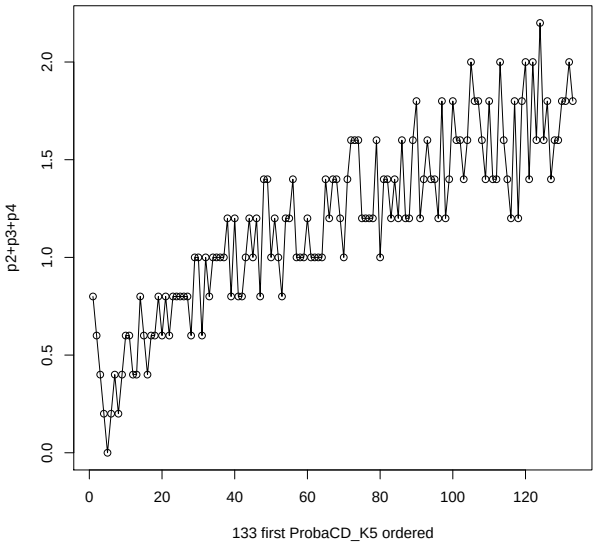


FIGURE 3 – Courbe avec en abscisse le rang et en ordonné p'

4.3 Robustesse des expériences

Les résultats que nous venons de commenter et analyser sont-ils robustes ? Persistent-ils quand on change les paramètres précis de notre expérience avec 2592 stratégies. C'est ce que nous étudions maintenant.

Dans une autre expérience, nous n'avons mis au départ que des stratégies qui commencent par c (il y en a donc deux fois moins exactement), ce qui correspond à la classe complète que nous notons ProbaC_K=5) le résultat est très proche : les mêmes 5 premières se retrouvent avec quelques permutations du classement.

Avec `ProbaCD_K=4` nous obtenons des résultats équivalents : en tête les rancunières patientes classées par ordre de patience décroissante (`p2=75%`, `p2=50%`, `p2=25%`, `p2=0`). Avec `ProbaCD_K=3` c'est encore la même chose.

Une objection pourrait être faite à notre méthode : les probabilistes composant la soupe initiale sont uniformément réparties (en faisant varier les `pi` par pas constant) : cette régularité pourrait entraîner des résultats spécifiques qui n'auraient donc pas de valeur générale.

Nous avons donc mené des expériences où nous avons choisi 2000 (puis 4000) stratégies au hasard parmi la classe `ProbaCD_K=10` (qui comporte $2 * 11^4 = 29282$ stratégies).

Les résultats obtenus confirment ceux de l'expérience principale avec `ProbaCD_K=5` : les gagnantes sont à chaque fois des stratégies de type rancunière patientes ou rancunière conciliante. On y trouve la confirmation du double critère.

Voici un exemple :

```
1   probaC_1.0_0.0_0.0_0.1
2   probaC_1.0_0.4_0.0_0.1
3   probaC_1.0_1.1_0.1_0.0_ZD_Extorq
4   probaC_1.0_0.3_0.7_0.0_ZD_Extorq
5   probaC_1.0_0.1_1.1_0.0_ZD_Extorq
6   probaC_1.0_0.3_0.8_0.0
7   probaC_1.0_0.3_0.4_0.1
8   probaC_1.0_0.1_1.1_0.1
9   probaC_1.0_0.4_0.3_0.1
10  probaC_1.0_0.0_0.3_0.2
```

Les trois extorqueurs qui apparaissent en position 3, 4 et 5 ont toutes un coefficient X qui vaut 1 (ce ne sont donc pas au sens strict des extorqueurs), et surtout elles sont toutes du type rancunières patientes ou conciliantes.

Une autre objection pourrait être formulée à notre méthode : nous ne considérons pas un assez grand nombre de ZD dans notre soupe initiale. Nous avons donc pris `ProbaCD_K=5` et ajouté une famille de 880 ZD (toutes celles dont les `pi` sont des multiples de $\frac{1}{32}$). Rien ne change pour l'essentiel : les 5 premières sont exactement les mêmes dans le même ordre que pour la soupe `ProbaCD_K=5` ; la première ZD est 25ème (à l'exception de rancunière qui est 5ème) et c'est `tit_for_tat` : `probaC_1.0_0.0_1.0_0.0_ZD_Extorq`.

La ZD suivante dans le classement est `ZD a=0.03125 b=-0.03125 c=0.0_ZD_Extorq` dont le X vaut 1.

Robustesse des stratégies nouvelles. Nous avons voulu savoir si les meilleures stratégies déterministes connues telles que les conclusions de [18] les ont déterminées obtenaient de bons résultats dans ces classes complètes probabilistes.

Nous avons donc mené le calcul pour `ProbaCD_K=5` + `Select` .

Peu de choses changent concernant les positions relatives des ProbaCD mais les meilleures de `Select` s'intercalent et prennent de très bonnes places.

```
1   spiteful_cc              19286
2   tft_spiteful             19078
3   gradual                  18235
4   probaC_1.0_0.8_0.0_0.0   17944
5   probaC_1.0_0.6_0.0_0.0   16922
6   probaC_1.0_0.4_0.0_0.0   15759
7   probaC_1.0_0.2_0.0_0.0   14921
8   probaC_1.0_0.0_0.0_0.0   14147
9   mem2                     14109
10  spiteful                 14073
```

On note que la huitième correspond à `spiteful` version probabiliste. Les deux `spiteful` ne sont pas côte à côte à cause fluctuations statistiques.

Toujours dans le but de tester la robustesse des stratégies identifiées, et en particulier pour voir ce qu'elles donnent quand elles se retrouvent parmi peu de probabilistes, nous avons composés une soupe avec les 20 premières de `ProbaCD_K=5`, les 1024 des `Mem(1,2)` et les stratégies de `Select`. On obtient :

```
1   probaC_1.0_0.2_0.0_0.2   5255
2   tft_spiteful             4945
3   probaC_1.0_0.4_0.0_0.2   4877
4   probaC_1.0_0.2_0.0_0.4   4415
5   winner12                 4331
6   mem12_CCCDCDDCDD         4331
7   probaC_1.0_0.6_0.0_0.2   4081
8   mem12_CCCDCDDDDD         3557
9   spiteful_cc              3557
10  probaC_1.0_0.8_0.0_0.0   3551
```

La sixième du classement est `winner12`, la huitième du classement est `spitefull_CC`. La onzième commence par CC et se fâche définitivement quand l'autre trahit deux fois de suite.

Les résultats (et de nombreux autres qui les confirment) sont très clairs : les stratégies les meilleures connues sont toujours bien classées (bien qu'elles proviennent d'expériences et de processus de sélection où seules des stratégies déterministes sont impliquées). Réciproquement, les nouvelles stratégies probabilistes repérées réussissent très bien dans des environnements composés presqu'exclusivement des

stratégies déterminismes (comme pour la dernière expérience).

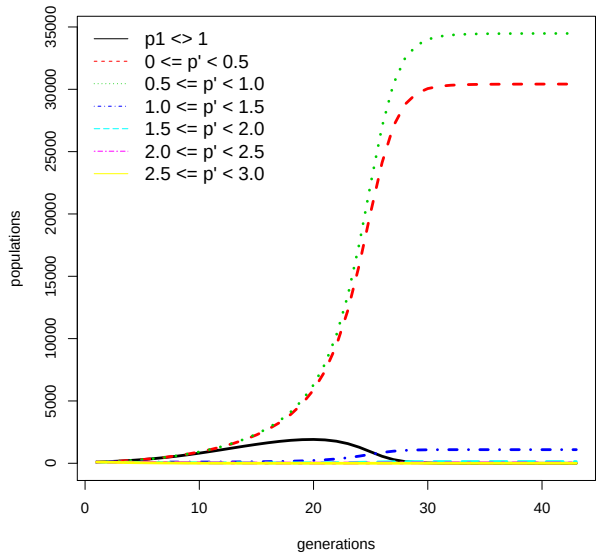


FIGURE 4 – Synthese par catégorie

5 Conclusion

L'expérimentation approfondie et systématique dans un modèle évolutionniste général et paramétré avec des stratégies de type "probabiliste à mémoire un coup" conduit à des résultats stables et que l'analyse permet d'anticiper une fois les bons paramètres identifiés. La double condition que nous avons extraite et qui semble-t-il n'avait jamais été notée conduit à définir une classe particulière de stratégies probabilistes à mémoire d'un coup (les rancunières conciliantes) dont les membres se retrouvent systématiquement en tête de tous les classements des compétitions évolutives que l'on peut imaginer, y compris quand on varie les soupes initiales pour y mettre de nombreuses stratégies extorqueurs, égaliseur ou ZD. De plus les stratégies repérées se montrent robustes et efficaces dans des soupes évolutives composées de manière variées, par exemple ne contenant que des stratégies déterministes. De ce fait, elles rejoignent la famille des meilleures stratégies connues répertoriées dans [18].

6 Annexe

Liste des 21 stratégies constituant `Select`.

1. `all_c` (gentille) : je coopère toujours.
2. `all_d` (méchante) : je trahis toujours.
3. `tit_for_tat` (donnant-donnant) : je coopère au premier coup, puis je joue au coup n ce que l'autre a joué au coup $n-1$.
4. `spiteful` (rancunière) : je coopère au premier coup et tant que l'adversaire coopère, mais dès qu'il trahit, je trahis indéfiniment.
5. `soft_majo` (majorité-mou) : je coopère au premier coup et tant que mon adversaire a coopéré plus ou autant qu'il a trahi dans le passé ; sinon je trahis.
6. `hard_majo` (majorité-dur) : je trahis au premier coup et tant que mon adversaire a trahi plus ou autant qu'il a coopéré dans le passé ; sinon je coopère.
7. `per_ddc` (périodique-ddc) : je joue d, d, c, d, d, c, ...
8. `per_ccd` (périodique-ccd) : je joue c, c, d, c, c, d, ...
9. `mistrust` (méfiante) : je trahis au premier coup, puis je joue au coup n ce que l'autre a joué au coup $n-1$.
10. `per_cd` (périodique-cd) : je joue c, d, c, d, c, d, ...
11. `pavlov` : Je coopère au premier coup, puis je joue toujours coopérer sauf quand les deux joueurs n'ont pas joué la même chose au coup précédent.
12. `tf2t` (donnant-donnant-mou) : je coopère aux deux premiers coups, puis je coopère toujours au coup n, sauf si mon adversaire a trahi au coup $n-1$ et au coup $n-2$.
13. `hard_tft` (donnant-donnant-dur) : je coopère aux deux premiers coups, puis je coopère toujours au coup n, sauf si mon adversaire a trahi au coup $n-1$ ou au coup $n-2$.
14. `slow_tft` (donnant-donnant-lent) : je coopère aux deux premiers coups, ensuite je me mettrais à trahir lorsque l'autre aura trahi deux fois de suite, et je ne me remettrai à coopérer dès que l'autre aura coopéré deux fois de suite.
15. `gradual` (graduelle) : je coopère au premier coup et quand la règle suivante n'est pas en train de s'appliquer : à chaque fois que l'autre me trahit, je compte le nombre, n, de ses trahisons passées et je trahis n fois consécutivement suivi de deux coopérations.
16. `prober` (sondeur) : aux trois premiers coups, je joue trahir-coopérer-coopérer (d-c-d) ; ensuite, si aux coups 2 et 3 l'adversaire n'a pas trahi, je trahis toujours ; sinon je joue `tit_for_tat`.
17. `mem2` : je commence par jouer deux coups comme `tit_for_tat` ; ensuite je change de comportement pour deux coups en fonction des résultats des deux derniers coups, selon les règles : (A) si les deux coups précédents ont été `[c, c]` `[c, c]`, je passe à `tit_for_tat` ; (B) si le dernier coup a été `[c, d]` ou `[d, c]` alors je passe à `hard_tft` ; (C) dans les autres cas, je passe à `all_d`. De plus, si à un moment l'autre trahit deux fois de suite, je passe définitivement à `all_d` [15].

18. `winner12` (gagnante de `Mem(1,2)`) ; je coopère aux deux premiers coups puis je joue la table : `[C CC]->C [C CD]->D [C DC]->C [C DD]->D [D CC]->D [D CD]->C [D DC]->D [D DD]->D`
19. `winner21` (gagnante de `Mem(2,1)`) ; aux deux premiers coups je joue D C puis je joue la table : `[CC C]-> C [CC D]->D [CD C]->C [CD D]->D [DC C]->C [DC D->D [DD C]->D [DD D]->D`
20. `tft_spiteful` (donnant-donnant-rancunier) : je joue `tit_for_tat`, sauf que si on me trahit deux fois de suite, je me mets à trahir indéfiniment.
21. `spiteful_cc` (rancunièreCC) : je coopère aux deux premiers coups puis je joue `spiteful`.

Références

[1] Christoph Adami and Arend Hintze. Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything. *Nature communications*, 4, 2013.

[2] Christoph Adami and Arend Hintze. Corrigendum : Evolutionary instability of zero-determinant strategies demonstrates that winning is not everything. *Nature communications*, 5, 2014.

[3] Robert M Axelrod. *The evolution of cooperation*. Basic books, 2006.

[4] Bruno Beaufils, Jean-Paul Delahaye, and Philippe Mathieu. Our meeting with gradual, a good strategy for the iterated prisoner's dilemma. In *Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems, ALIFE V*, pages 202–209. The MIT Press/Bradford Books, 1997.

[5] Bruno Beaufils, Jean-Paul Delahaye, and Philippe Mathieu. Complete classes of strategies for the classical iterated prisoner's dilemma. In *International Conference on Evolutionary Programming, EP7, Vol1447*, pages 33–41. Springer, 1998.

[6] Bruno Beaufils and Philippe Mathieu. Cheating is not playing : Methodological issues of computational game theory. *In Proceedings of the 17th European Conference on Artificial Intelligence (ECAI'06)*, 141 :185–189, 2006.

[7] Maarten C Boerlijst, Martin A Nowak, and Karl Sigmund. Equal pay for all prisoners. *The American mathematical monthly*, 104(4) :303–305, 1997.

[8] Jean-Paul Delahaye and Philippe Mathieu. Méta-stratégies pour le dilemme itéré du prisonnier. In *24e Journées Francophones sur les Systèmes Multi-Agents (JFSMA'16)*, pages 13–22. Cépaduès, 2016.

[9] Hao Dong, Rong Zhi-Hai, and Zhou Tao. Zero-determinant strategy : An underway revolution in game theory. *Chinese Physics B*, 23(7) :078905, 2014.

[10] Christian Hilbe, Martin A Nowak, and Karl Sigmund. Evolution of extortion in iterated prisoner's dilemma games. *Proceedings of the National Academy of Sciences*, 110(17) :6913–6918, 2013.

[11] Christian Hilbe, Martin A Nowak, and Arne Traulsen. Adaptive dynamics of extortion and compliance. *PloS one*, 8(11) :e77886, 2013.

[12] Christian Hilbe, Torsten Röhl, and Manfred Milinski. Extortion subdues human players but is finally punished in the prisoner's dilemma. *Nature communications*, 5, 2014.

[13] Graham Kendall, Xin Yao, and Siang Yew Chong. *The iterated prisoners' dilemma : 20 years on*. World Scientific Publishing Co., Inc., 2007.

[14] Jiawei Li, Philip Hingston, and Graham Kendall. Engineering design of strategies for winning iterated prisoner's dilemma competitions. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(4) :348–360, 2011.

[15] Jiawei Li and Graham Kendall. The effect of memory size on the evolutionary stability of strategies in iterated prisoner's dilemma. *IEEE Transactions on Evolutionary Computation*, 18(6) :819–826, 2014.

[16] Jie Liu, Y Li, C Xu, and PM Hui. Evolutionary behavior of generalized zero-determinant strategies in iterated prisoner's dilemma. *Physica A : Statistical Mechanics and its Applications*, 430 :81–92, 2015.

[17] Philippe Mathieu, Bruno Beaufils, and Jean-Paul Delahaye. Studies on dynamics in the classical iterated prisoner's dilemma with few strategies. In *European Conference on Artificial Evolution*, pages 177–190. Springer, 1999.

[18] Philippe Mathieu and Jean-Paul Delahaye. New winning strategies for the iterated prisoner's dilemma. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 1665–1666. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

[19] Manfred Milinski, Christian Hilbe, Dirk Semmann, Ralf Sommerfeld, and Jochem Marotzke. Humans choose representatives who enforce cooperation in social dilemmas through extortion. *Nature communications*, 7, 2016.

[20] Colm O'Riordan et al. A forgiving strategy for the iterated prisoner's dilemma. *Journal of Artificial Societies and Social Simulation*, 3(4) :56–58, 2000.

[21] William H Press and Freeman J Dyson. Iterated prisoner's dilemma contains strategies that dominate any evolutionary opponent. *Proceedings of the National Academy of Sciences*, 109(26) :10409–10413, 2012.

[22] Anatol Rapoport and Albert M Chammah. *Prisoner's dilemma : A study in conflict and cooperation*, volume 165. University of Michigan press, 1965.

[23] Karl Sigmund. *The calculus of selfishness*. Princeton University Press, 2010.

[24] Alexander J Stewart and Joshua B Plotkin. From extortion to generosity, evolution in the iterated prisoner's dilemma. *Proceedings of the National Academy of Sciences*, 110(38) :15348–15353, 2013.

[25] Attila Szolnoki and Matjaž Perc. Defection and extortion as unexpected catalysts of unconditional cooperation in structured populations. *Scientific reports*, 4, 2014.

[26] Elpida Tzafestas. Toward adaptive cooperative behavior. In *Proceedings of the Simulation of Adaptive Behavior Conference, Paris*. Citeseer, 2000.

Approche formelle pour la modélisation et la simulation de systèmes multi-agents

R. Franceschini[a] franceschini@univ-corse.fr.fr
P.-A. Bisgambiglia[a] pa.bisgambiglia@univ-corse.fr
P. Bisgambiglia[a] bisgambi@univ-corse.fr

[a]UMR CNRS SPE, équipe informatique SiSU,
Université de Corse Pasquale Paoli Corte, France

Résumé

Dans cet article, nous proposons d'intégrer une approche formelle pour la spécification de modèles de simulation orientés agent, dans le but de rendre les modèles interopérables et de rendre les résultats de simulation plus facilement reproductibles.

Mots-clés : *Simulation, Modèle, IRM4S, PDEVS*

Abstract

In this paper, we propose a formal approach allowing to specify agent-based models in order to make simulation results more easily reproducible.

Keywords: *Simulation, Model, IRM4S, PDEVS*

1 Introduction

Nous proposons dans cet article les bases théoriques d'une nouvelle association entre les systèmes multi-agents (SMA) et la théorie de la modélisation et de la simulation (TMS) à partir du formalisme PDEVS [1]. Elle s'appuie sur des travaux déjà éprouvés (DSDEMAS [2] et IRM4S [3]) et ajoute des mécanismes de modélisation générique des environnements. L'un de nos objectifs est de montrer que l'utilisation d'une méthode formelle pour la modélisation et la simulation de systèmes multi-agents est gage de respect de la méthode scientifique, et donc une avancée en terme de vérification et de validation (V&V) des modèles et de reproductibilité des expériences numériques de simulations [4].
La reproductibilité au niveau modélisation (échange, test et vérification de modèles) est déjà une problématique bien identifiée par la communauté [5, 3]. L'étude menée par [6] qui consiste à implémenter un modèle épidémiologique simple sur trois plateformes SMA différentes montre qu'après simulation, chaque plateforme produit des résultats différents. Cette étude nous alerte donc sur la nécessité de spécifier un comportement non-ambigu du modèle, qui est primordial pour la simulation d'un modèle. Parmi les facteurs qui peuvent influencer les résultats, l'ordonnancement des agents a été souligné [7]. Le fait qu'une multitude d'agents effectuent leurs actions potentiellement de manière simultanée dans un environnement commun est problématique. Leur ordre d'activation peut influer sur les résultats et donc entraîner des trajectoires d'états différentes.
Les contributions de cet article pour résoudre ces problèmes sont : (1) nous soulevons le problème de la reproductibilité des simulations multi-agents ; (2) nous listons, sous forme de cahier des charges, les fonctionnalités à mettre en oeuvre pour faciliter cette reproductibilité ; (3) nous présentons l'approche générique *Dynamic Parallel Discrete Event Multi Agent System* (DPDEMAS), une spécification SMA basée sur un formalisme muni d'une sémantique et indépendant d'une plateforme.

2 Contexte

2.1 Positionnement

Le but de notre spécification est de définir la structure d'un modèle SMA accompagné de sa sémantique pour la simulation à évènements discrets. Dans cette optique, nous veillons à respecter un ensemble de propriétés exposées dans la littérature [8, 3, 9].

En ce qui concerne les agents : respect des propriétés qui définissent un agent [10] : *proactivité*, *réactivité*, *sociabilité* et *autonomie*, laquelle passe notamment par le respect de la *contrainte d'intégrité interne* [3] (l'agent dispose du contrôle total de son état).
En ce qui concerne les environnements : (1) permettre la définition d'environnements multiples [9] (physiques et/ou sociaux), (2) permettre une dynamique endogène, (3) respect de la contrainte de localité (les agents perçoivent leur voisinage en fonction d'une position) et enfin, (4) respect de la contrainte d'intégrité de

l'environnement, qui consiste à s'assurer que les agents n'effectuent pas de modification directe de l'état de l'environnement.
La spécification doit permettre une dynamique structurelle afin de matérialiser les aspects dynamiques d'un SMA (création/destruction d'entités, ...). Enfin, elle doit permettre la gestion d'actions simultanées afin d'éviter les problématiques liées à l'ordonnancement des agents [7], qui peut influer sur les résultats de simulation.

2.2 Le formalisme PDEVS

La théorie de la modélisation et de la simulation (TMS) développée par [1] permet de spécifier des systèmes (continus, discrets ou hybrides) à l'aide du formalisme Parallel Discrete Event System Specification (PDEVS). Ce formalisme permet de décrire des systèmes de manière modulaire et hiérarchique. Un modèle couplé décrit le système en tant que réseau de composants ; un modèle atomique décrit le comportement autonome d'un système. Formellement, un modèle atomique est définit par $M = (X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, ta)$, où X et Y correspondent aux ensembles port-valeur d'entrée et de sortie et S à l'état du modèle. Le comportement est défini par un ensemble de transitions d'états déclenchées lors de l'occurence d'évènements. Il y a deux types d'évènements : le premier est interne au modèle et correspond à la transition interne (δ_{int}) ; le second est externe au modèle (réception d'entrées) et correspond à la transition externe (δ_{ext}). Une troisième transition permet de gérer l'arrivée simultanée de ces deux types d'évènements (δ_{con}). Les évènements internes sont planifiés par le modèle à travers la fonction d'avancement du temps ta. Au moment où ils se produisent, le modèle peut effectuer une sortie à travers la fonction λ.

L'un des atouts de PDEVS réside dans la séparation explicite de la modélisation et de la simulation. Il bénéficie également d'une sémantique opérationnelle, ce qui permet d'exécuter les modèles sur une implémentation du simulateur PDEVS de manière non-ambigüe. Il a été montré que PDEVS est suffisamment abstrait pour représenter d'autres types de formalismes [11] et qu'il permet donc de faire interagir des modèles exprimés dans des formalismes différents au sein d'une même simulation. Grâce aux différentes extensions du formalisme, il est tout à fait envisageable de représenter le paradigme agent à l'aide de cet outil. Enfin, PDEVS permet de gérer la simultanéité des évènements et de rendre explicite le comportement du système face à la simultanéité, au même titre que le modèle d'action IRM4S [3] dont la sémantique d'exécution est compatible avec PDEVS. L'intérêt de formaliser les SMAs à l'aide du formalisme PDEVS a déjà été mis en évidence et a déjà fait l'objet d'un certain nombre de travaux. Nous en présentons une partie dans la section suivante.

2.3 Travaux connexes

Une première analogie. Les premiers travaux de formalisation des SMAs basés sur le formalisme DEVS sont proposés par [12]. Ces travaux mettent en évidence les similitudes entre les propriétés d'un agent et la structure d'un modèle DEVS. Ainsi, une première analogie est faite entre la perception d'un agent et la transition externe. La fonction de sortie est considérée comme analogue à l'action et enfin, comme l'agent, le modèle atomique est capable de proactivité à travers sa transition interne.

Travaillant sur des problématiques de modélisation d'écosystèmes, leur besoin de représenter des structures dynamiques les poussent à proposer l'extension ρ-DEVS [13]. En revanche, bien que la simulation orientée agent (ABS) soit à la base de ces travaux, rien n'est mentionné à propos de la modélisation de l'environnement. Cette lacune a été récemment comblée à travers la définition de ML-DEVS [14], une extension de ρ-DEVS qui permet de définir un comportement à un modèle couplé. Le modèle couplé joue alors le rôle d'environnement pour ses différents composants, notons cependant que cette approche ne permet pas de représenter des SMAs multi-environnements.

Dynamic Structure Discrete Event Multiagent Systems. La spécification DSDEMAS, pour *Dynamic Structure Discrete event Multiagent Systems* [2], est une formalisation du paradigme multi-agent basée sur le formalisme PDEVS et l'extension DSDE [15], une alternative à ρ-DEVS pour la dynamique structurelle. Dans cette formalisation, un SMA est représenté par un modèle couplé DSDE, dont les composants forment l'ensemble des agents et des environnements. Le SMA a la possibilité de changer sa propre structure. Le SMA peut lui-même être considéré comme un agent.

Un agent est également représenté par un modèle DSDE et est formé par l'ensemble des modèles qui le compose. Une analogie est faite

entre la *perception* des agents et l'arrivée d'évènements externes sur les ports d'entrées des modèles, impliquant un changement d'état via les fonctions de transition externe des modèles. En ce qui concerne l'*action*, une analogie est faite avec les fonctions de sorties. L'ensemble des modèles dénués de ports d'entrée et initiant des sorties forment la *proactivité* de l'agent. Le comportement *réflexe* est formé par l'ensemble des modèles initiant une sortie juste après la perception, à travers un état transitoire. L'*autonomie* de l'agent est formée par l'ensemble des modèles non couplés aux ports d'entrée ou de sortie du modèle agent. L'union de la perception, proaction, action et autonomie forme le comportement de l'agent. En ce qui concerne l'environnement, les auteurs laissent au modélisateur la possibilité d'utiliser un environnement *centralisé*, *distribué* ou vu comme un *agent*, conformément aux travaux de [9].

Cette spécification, contrairement à l'approche présentée précédemment, supporte une approche multi-environnement. En revanche, seuls les modèles génériques sont formalisés et doivent être adaptés à chaque application. Ainsi, la modélisation n'est pas facilitée et reste dépendante de chaque application. L'agent est systématiquement considéré comme un réseau dynamique de modèles, ce qui permet de définir des agents de manière modulaire et d'envisager des architectures d'agents complexes mais qui n'est pas forcément adapté pour représenter des agents dont le processus de délibération est plus simple (*eg.* une architecture de subsombption).

M-DEVS. Plus récemment, M-DEVS [16] pose les bases d'une approche dérivée du formalisme PDEVS. Le but de M-DEVS est de définir une sémantique non-ambigüe pour la simulation de SMAs à évènements discrets satisfaisant un certain nombre de propriétés : le comportement réactif et proactif de l'agent (déjà supporté par un modèle PDEVS), la dynamique structurelle afin de modifier la structure des interactions et de créer/supprimer des entités au cours de la simulation, la concurrence pour permettre de traiter des influences simultanées et à laquelle PDEVS apporte une solution « directe » et enfin, l'instantanéité qui permet de traiter séparément les influences physiques (action des agents) des influences logiques (perception, message informationnel), qui sont instantanées.

Pour répondre à la problématique de la dynamique structurelle, l'auteur s'inspire à la fois de DSDE et de ρ-DEVS et ajoute la notion d'influences structurelles permettant de modifier la structure du système de manière réflexive, propagées via une nouvelle fonction de sortie.

Si la définition d'une unique extension permettant d'englober l'ensemble des propriétés pour la spécification d'un SMA semble attrayante, M-DEVS ne rassemble pas toutes les qualités nécessaires. Ces travaux ne peuvent être considérés comme une véritable extension du formalisme PDEVS étant donné que la preuve de fermeture de couplage, qui permet de montrer une équivalence entre un modèle atomique et un modèle couplé et de garantir que le couplage n'a pas d'influence sur les trajectoires d'états d'un modèle, n'a à notre connaissance pas été prouvée. De plus, nous pensons que le rôle d'une telle approche est de simplifier la définition de modèles pour le paradigme agent, comme c'est le cas de Cell-DEVS [17] pour les automates cellulaires. Or, les modifications apportées par M-DEVS à la structure d'un modèle atomique complexifient la définition d'un agent (ajout de trois nouvelles fonctions et modification de la sémantique de fonctions existantes).

De notre point de vue, l'approche DSDEMAS est la plus complète et la plus fidèle aux concepts de la TMS. Nos travaux, présentés dans la section suivante, sont à placer dans cette continuité.

3 Spécification DPDEMAS

Les SMAs, du fait du nombre important de sous-systèmes qui les composent, de leurs interactions, et de la diversité des domaines d'applications, font qu'il est contraignant de les exprimer à l'aide d'un seul formalisme. Le multi-formalisme PDEVS peut-être utilisé comme pivot. Proposer une extension PDEVS permettant d'exprimer le paradigme SMA et tous les aspects listés section 2.1 nous semble difficile sans complexifier le formalisme PDEVS et sans figer le SMA dans un certain point de vue conceptuel. Nous suivons donc la même approche que DSDEMAS tout en allant plus loin dans la spécification dans le but de proposer des modèles révisables et réutilisables pour faciliter la modélisation d'un SMA.

3.1 Le système multi-agent

Nous formalisons le système multi-agent à l'aide d'un modèle couplé DSDE [15] $DPDEMAS = (X_{MAS}, Y_{MAS}, \chi, M_\chi)$ dont la particularité est de définir un

modèle atomique spécifique $M_\chi = (X_\chi, Y_\chi, S_\chi, \delta_{int_\chi}, \delta_{ext_\chi} \lambda_\chi, ta_\chi, \zeta)$ appelé modèle exécutif (ou χ) qui peut agir sur la topologie du réseau de modèles. Le SMA est ainsi capable de modifier sa structure de manière dynamique afin de refléter l'évolution des interactions et la création/destruction de nouvelles entités. La définition de M_χ est identique à celle d'un modèle atomique (*cf.* section 2.2), excepté pour la fonction ζ qui permet d'accéder à la structure du réseau de composants associée à un état $s_\chi \in S_\chi$:

$$\zeta(s_\chi) = (D, \{M_d\}, \{I_d\}, \{Z_{i,d}\})$$

où D est l'ensemble des identifiants des modèles avec $\{M_d\}$ l'ensemble des modèles et $\{I_d\}$ et $\{Z_{i,d}\}$ représentent les informations de couplage et sont identiques à la définition d'un modèle couplé PDEVS.
Le modèle $DPDEMAS$ est composé de modèles d'environnements et de modèles agents (Fig. 1). Nous notons $D_{Ag} \subset D$ l'ensemble des identifiants des modèles agents et $D_E \subset D$ l'ensemble des identifiants des modèles d'environnements. L'ensemble des identifiants est donc $D = D_{Ag} \cup D_E$. Celui des modèles est défini par $\{M_d\} = \{E_d \mid d \in D_E\} \cup \{Ag_d \mid d \in D_{Ag}\}$.

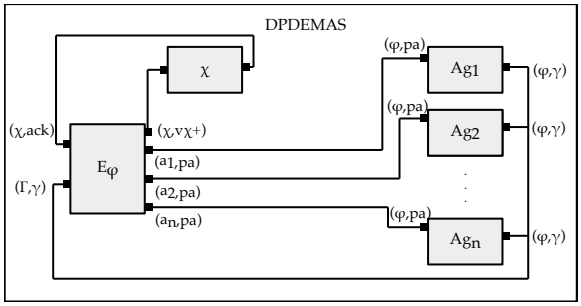


FIGURE 1 – Vue d'ensemble d'un SMA spécifié avec DPDEMAS.

Nous donnons la possibilité de coupler le modèle $DPDEMAS$ au sein d'un réseau de modèles, c'est pourquoi celui-ci peut disposer d'une interface d'entrées/sorties. Les ports d'entrée peuvent être utiles pour : (1) la réception de paramètres d'entrée afin d'initialiser la simulation ; (2) représenter des perturbations provenant de modèles exogènes au SMA. Les ports de sorties peuvent être utiles pour représenter des valeurs agrégées (comme la population saine et la population infectée dans le cadre d'un modèle épidémiologique par exemple). Le SMA peut également être vu comme un agent au sein d'un SMA de niveau hiérarchique supérieur. Si c'est le cas, le modèle SMA doit respecter l'interface que nous allons spécifier ci-dessous pour l'agent.

3.2 Les agents

Nous nous appuyons pour la spécification des agents sur l'analogie proposée dans [12, 18] entre un modèle DEVS et un agent, laquelle permet de mettre en évidence la capacité d'un modèle DEVS (couplé ou non) à satisfaire les propriétés définissant un agent selon [10]. *Autonomie* : le modèle est seul responsable de son état ; *proactivité* : à travers la transition interne ; *réactivité* : perception à travers la transition externe et action à travers la fonction de sortie après un état transitoire ; et *sociabilité* : à travers la communication via les ports.

Cependant, nous reprenons cette analogie pour définir le *système cognitif* de l'agent, étant donné que nous utilisons la séparation explicite du « corps » et de « l'esprit » de l'agent [9]. Cette approche permet de respecter la contrainte d'intégrité interne de l'agent [3] et donc l'autonomie de l'agent. Bien qu'un modèle PDEVS soit capable d'évoluer de manière autonome, séparer le corps et l'esprit de l'agent permet de rendre explicite les variables d'états considérées comme perceptibles et éventuellement modifiables par les autres agents et de les décorréler du processus décisionnel.

Système cognitif. Le système cognitif a la responsabilité de collecter les perceptions, d'en tenir compte dans son processus de raisonnement avant de concrétiser ses décisions à travers l'action. Il représente en somme, le processus de délibération à trois phases identifié par [19]. Un modèle PDEVS permet déjà de réaliser ce processus de délibération. Nous représentons donc le système cognitif de l'agent par un modèle PDEVS. En revanche, nous ne donnons aucune directive sur l'utilisation d'un modèle atomique, couplé, ou encore dynamique. Nous pensons que cette décision revient au modélisateur en fonction du type d'agent qu'il souhaite modéliser et de sa complexité. Les travaux de [12] puis de [2] montrent qu'un agent est analogue à une structure DEVS, que l'on utilise un simple modèle atomique ou un couplé dynamique. Ainsi, un agent *réactif* sera mieux représenté par un PDEVS atomique, tandis qu'un agent *cognitif* capable de raisonnement et basé sur une architecture telle que *Beliefs-Desire-Intentions* [20] sera mieux représenté par un modèle couplé

PDEVS, voir par un DSDE si l'agent est capable d'adapter sa structure pour modifier son comportement.

Afin qu'il puisse être interfacé avec les autres entités du SMA que nous définirons par la suite, un agent doit respecter une interface d'entrées / sorties spécifique.

L'ensemble des ports-valeurs d'entrée d'un agent est défini par :

$$X_{Ag} = X_{Ag_{env}} \cup X_{Ag_{agents}} \cup X_{Ag_{exo}}$$

où $X_{Ag_{env}} = \{(p, v) \mid p \in D_E, v \in Pa\}$ est l'ensemble des messages provenant des environnements dans lesquels l'agent est plongé. Chaque port est nommé d'après l'identifiant de l'environnement qui émet le message et la valeur du message représente un percept de l'agent. D_E est l'ensemble des identifiants des environnements et Pa est l'ensemble des percepts que nous définissons dans la suite de ce document. $X_{Ag_{agents}}$ représente les messages directs provenant d'autres agents, dans le cas où l'agent évolue dans un environnement social. Enfin, $X_{Ag_{exo}}$ représente les messages destinés aux modèles exogènes au système multi-agent.

De la même manière, les sorties de l'agent sont définies par : $Y_{Ag} = Y_{Ag_{env}} \cup Y_{Ag_{agents}} \cup Y_{Ag_{exo}}$ où $Y_{Ag_{env}} = \{(p, v) \mid p \in D_E, v \in \Gamma\}$ est l'ensemble des messages destinés aux environnements dans lesquels l'agent évolue, le port étant l'identifiant de l'environnement dans lequel l'agent souhaite réaliser une action, cette dernière étant représentée par une influence ; $Y_{Ag_{agents}}$ représente les messages directs destinés aux agents dans le cas où l'agent évolue dans un environnement social ; enfin, $Y_{Ag_{exo}}$ représente les messages destinés à des modèles exogènes au système multi-agent.

Corps. Pour chaque environnement physique $e \in D_E$ dans lequel un agent $a \in D_{Ag}$ évolue, il existe au sein de l'état de l'environnement un « corps » $\beta_{a,e}$ défini par un triplet : $\beta_{a,e} = (p, \psi, \theta)$ où $p \in P$ représente la position du corps de l'agent dans l'environnement ; $\psi \in \wp(P)$ est une partie de l'ensemble des positions et représente la portée de perception/action de l'agent ; enfin, θ est utilisé pour définir l'ensemble des variables d'état « publiques » de l'agent au sein de cet environnement et que les autres agents pourront percevoir et/ou modifier.

À travers son corps, l'agent entretient une étroite relation avec son environnement. Si l'agent perçoit et agit au sein de celui-ci, il ne peut le faire que de manière limitée, en fonction de sa position spatiale ou sociale et d'une certaine portée. À travers la définition de cette structure et des variables p et ψ, il est important de noter que l'agent est bien *situé* au sein de son environnement, et que la variable ψ nous permet de matérialiser la portée de ses perceptions et de ses actions. Cette définition du corps nous permet donc de satisfaire la *contrainte de localité* des perceptions/actions [3].

3.3 Spécification générique de l'environnement

Nous nous appuyons ici sur PDEVS et sur le modèle d'action IRM4S afin de permettre à l'environnement d'assumer son rôle de médiation des interactions.

Au-delà de l'interface d'entrées/sorties qu'un environnement DPDEMAS doit respecter, il est possible d'identifier les caractéristiques communes aux différents environnements (l'état) et de définir une mécanique générale (le comportement). Comme pour le système cognitif d'un agent, nous pensons que le type de modèle doit être adapté en fonction de la topologie visée (e.g. un modèle cellulaire peut être représenté par un modèle Cell-DEVS [17], un modèle continu par un atomique). Néanmoins, afin de formaliser la sémantique de l'environnement, nous spécifions l'état générique et le comportement à travers la définition d'un modèle atomique PDEVS tel que defini section 2.2) :

$$E = (X, Y, S, \delta_{int}, \delta_{ext}, \delta_{con}, \lambda, \text{ta})$$

Interface d'entrées/sorties. Le rôle de médiation des interactions de l'environnement se traduit notamment par la structure de ses entrées/sorties. Peu importe la forme que prend un environnement, ce dernier doit fournir une interface spécifique. L'ensemble des ports d'entrée de l'environnement est défini par : $IPorts = \{\chi, inf\} \cup IPorts_{exo}$ où χ est le port d'entrée permettant au modèle exécutif du modèle SMA de signaler la prise en compte des changements structuraux demandés. Le port inf est destiné à recevoir les influences envoyées par les systèmes cognitifs des agents plongés dans cet environnement ($X_{inf} = \Gamma$). Enfin, $IPorts_{exo}$ représente d'autres ports d'entrée éventuels destinés à recevoir des messages exogènes au SMA.

L'ensemble des sorties de l'environnement est défini par : $Y = Y_{mas} \cup Y_{agents} \cup Y_{exo}$ où

$Y_{mas} = \{(\chi, y_\chi) \mid y_\chi \in Y_\chi\}$ est l'ensemble des couples possibles pour le port de sortie χ, qui est destiné à être rattaché au modèle exécutif du modèle SMA et dont les valeurs représentent des demandes de modifications structurelles. Ces demandes ont pour but de répercuter sur la structure certains changements d'états de l'environnement (e.g. des naissances d'agents). $Y_{agents} = \{(p, v) \mid p \in A, v \in Pa\}$ représente l'ensemble des messages destinés aux agents plongés dans l'environnement ($A \subseteq D_{Ag}$) dont les valeurs représentent un percept. Enfin, Y_{exo} est l'ensemble de port-valeurs éventuels destinés à des modèles exogènes au SMA.

Etat. L'état de notre modèle d'environnement générique est défini par le sextuplet suivant :

$$S = \{(phase, \sigma, A, \Sigma, \Gamma, v_\chi^+)\}$$

où $phase = \{natural, perception, req\chi, await\chi\}$ est la variable qui représente le nom donné à l'état courant de l'environnement ; $A \subseteq D_{Ag}$ est l'ensemble des identifiants des agents qui sont plongés dans l'environnement ; $\Gamma = \{\Gamma_a \mid a \in A\} \cup \Gamma_e$ est l'ensemble des influences produites pour cet état par les agents et l'environnement si celui-ci dispose d'une dynamique endogène ; $\Sigma \in \Sigma^*$ est une structure associée à chaque état $s \in S$ de l'environnement qui diffère en fonction du type d'environnement (physique ou social) ; $\sigma \in \mathbb{R}^+_{0,\infty}$ est une variable introduite pour indiquer le prochain temps d'activation de l'environnement si celui-ci a une dynamique endogène (sa valeur par défaut est ∞) ; enfin, v_χ^+ est utilisé pour stocker les messages destinés à l'exécutif du modèle SMA lorsqu'une influence est susceptible d'entraîner des changements structuraux.

Mécanique générale. Nous associons à chaque environnement $e \in D_E$ deux fonctions, $natural_e$ et $reac_e$, permettant de décomposer les fonctions de transitions de l'environnement. A l'instar du modèle IRM4S, la première fonction permet à l'environnement d'avoir une dynamique endogène en produisant à partir de son propre état total un ensemble d'influences : $natural_e : Q \to \Gamma'$. La seconde correspond au calcul de réaction de l'environnement, qui à partir de son état total et de l'ensemble des influences produites par les agents et l'environnement permet de calculer un nouvel état : $reac_e : Q \times \Gamma' \to S$. Ces deux dernières fonctions doivent être définies par le modélisateur et ont vocation à être utilisées au sein même des fonctions de transition de l'environnement. Cette décomposition nous permet de définir un comportement générique aux différentes transitions de l'environnement.

Globalement, notre environnement dispose des responsabilités suivantes : réceptionner et/ou produire de manière endogène des influences avant de traiter celles-ci afin de calculer un nouvel état et éventuellement demander des changements structuraux au niveau du SMA.

La gestion des demandes de changements structuraux est effectuée de la manière suivante : après modification de l'état de l'environnement par la fonction $reac_e$, si la variable $v_\chi^+ \neq \varnothing$, l'environnement passe dans un état transitoire ($phase = req\chi$) afin de réaliser une sortie permettant d'envoyer les demandes de modifications au modèle exécutif SMA, puis passe dans un état passif ($phase = await\chi$) jusqu'à la réception d'un message de l'exécutif. Lorsque les modifications structurelles sont effectuées (réponse du modèle exécutif), l'environnement passe dans un état transitoire ($phase = perception$) permettant d'effectuer les sorties nécessaires destinées aux agents (qui correspondent aux percepts des agents) avant de repasser dans l'état $natural$ destiné à l'attente/-production d'influences. Dans le cas où le traitement des influences suite à l'exécution de la fonction $reac_e$ ne nécessitent pas de modification de la structure du SMA ($v_\chi^+ = \varnothing$), l'environnement envoie directement les percepts aux agents concernés.

Soit un environnement $e \in D_E$ dans l'état $s_e = (phase, \sigma, A, \Sigma, \Gamma, v_\chi^+)$, auquel a été associé les fonctions $natural_e$ et $reac_e$. La fonction d'avancement du temps est définie par :

$$ta(s_e) = \begin{cases} 0 & \text{si } phase \in \{req\chi, perception\} \\ \infty & \text{si } phase = await\chi \\ \sigma & \text{si } phase = natural \end{cases}$$

La sortie associée à l'état $req\chi$ correspond à l'envoi des demandes de modifications structurelles stockées dans la variable v_χ^+ au modèle exécutif :

$$\lambda(req_\chi, \sigma, A, \Sigma, \Gamma, v_\chi^+) = \{(\chi, v_\chi^+)\}$$

Nous ne détaillons pas ici la sortie associée à l'état $perception$ étant donné que celle-ci dépend du type d'environnement utilisé (et donc de la structure de Σ).

La fonction de transition interne nous permet de réaliser la dynamique endogène ainsi que le cal-

cul de réaction aux influences de l'environnement. Ci-dessous sa définition pour chaque état :

$$\delta_{int}(s_e) = \begin{cases} reac_e((s_e, 0), natural_e(s_e, 0) \cup \Gamma) \\ \qquad \text{si } phase = natural \\ (await_\chi, \sigma, A, \Sigma, \Gamma, \varnothing) \\ \qquad \text{si } phase = req_\chi \\ (natural, \sigma, A, \Sigma, \Gamma, v_\chi^+) \\ \qquad \text{si } phase = perception \end{cases}$$

Si l'environnement est dans l'état $natural$, l'environnement effectue sa dynamique endogène à travers la fonction $natural_e$ puis le nouvel état est calculé grâce à la fonction $reac_e$. La phase qui résulte de la fonction $reac_e$ peut être de deux types : $req\chi$ si $v_\chi^+ \neq \varnothing$ ou $perception$ si $v_\chi^+ = \varnothing$.

Si $phase = req\chi$, l'environnement vient d'effectuer une sortie pour le modèle exécutif. Il passe alors dans l'état passif $await\chi$ dans lequel $v_\chi^+ = \varnothing$. Enfin, si $phase = perception$, cela veut dire que l'environnement vient d'envoyer les percepts aux agents et passe donc dans l'état $natural$.

En ce qui concerne la fonction de transition externe, sa définition est moins concise, nous donnons donc son comportement pour chaque phase. Lorsque l'environnement est dans la phase $natural$, il peut recevoir à tout moment des influences de la part d'autres agents, qu'il traite grâce à la fonction $reac_e$:

$$\delta_{ext}((natural, \sigma, A, \Sigma, \Gamma, v_\chi^+), e, \\ ((inf, \gamma_1), (inf, \gamma_2), ..., (inf, \gamma_n))) = \\ reac_e(q_e, \bigcup_{i=1}^{n} \gamma_i \cup \Gamma)$$

Lorsque l'environnement reçoit une réponse concernant les changements de structure demandés à l'exécutif du SMA ($phase = await\chi$), l'environnement passe dans l'état $perception$ afin d'envoyer les percepts aux agents.

$$\delta_{ext}((await\chi, \sigma, A, \Sigma, \Gamma, v_\chi^+), 0, (\chi, x)) \\ = (perception, \sigma - e, A, \Sigma, \Gamma, v_\chi^+)$$

Enfin, la fonction de transition de confluence permet de gérer la collision entre la dynamique endogène et la réception d'influences produites par les agents au même instant :

$$\delta_{con}((natural, \sigma, A, \Sigma, \Gamma, v_\chi^+), \\ ((inf, \gamma_1), (inf, \gamma_2), ..., (inf, \gamma_n))) \\ = reac_e(q_e, \bigcup_{i=1}^{n} \gamma_i \cup \Gamma \cup natural_e(q_e))$$

L'environnement générique étant défini, nous pouvons nous intéresser de manière plus spécifique aux environnements physiques et sociaux.

Environnement physique. L'environnement *physique* fournit les règles et contraintes qui régissent et permettent l'existence des agents et des ressources. Nous nous appuyons pour définir l'environnement physique des travaux de [21]. Afin de simplifier la spécification de l'environnement physique, nous utilisons la notation « point », conventionnellement utilisée dans les langages de programmation orientés objet (*e.g.* l'élément $\beta_{a,e}.p$ fait référence à l'élément p du n-uplet $\beta_{a,e}$).

Soit $s_\varphi = (phase, \sigma, A, \Sigma, \Gamma, v_\chi^+)$, l'état d'un environnement *physique* $\varphi \in D_E$ tel que défini précédemment. La structure Σ associée à cet état $s_\varphi \in S_\varphi$ est définie par le quadruplet suivant : $\Sigma = ((P, dist), \Xi, \theta)$ Le tuple $(P, dist)$ est un espace *quasimétrique* composé de P, l'ensemble des positions dans l'espace, et de $dist$, qui est une fonction quasimétrique permettant de définir une topologie : $dist : P \times P \rightarrow \mathbb{R}_\infty^+$ L'espace quasimétrique utilisé par [21] pour formaliser la notion d'environnement SMA permet de représenter n'importe quel environnement physique continu ou discret, y compris un graphe orienté étant donné que la fonction de quasidistance ne vérifie par nécessairement la propriété de symétrie. Ainsi, si l'environnement modélisé représente un environnement continu à deux dimensions, $P = \mathbb{R}^2$ avec $dist$ la distance euclidienne par exemple.

Ξ représente l'ensemble des entités situées au sein de l'environnement. Nous distinguons deux types d'entités : les corps des agents tels que définis dans la section 3.2 et les ressources manipulables par les agents. Nous notons l'ensemble des corps situés dans l'environnement $\beta^* = \{\beta_{a,\varphi} \mid a \in A\}$. Les ressources, quant à elles, sont définies par l'ensemble $R = \{(p, \vartheta) \mid p \in P\}$, où p représente la position de la ressource au sein de l'espace et ϑ est une variable utilisée pour représenter les informations relatives à cette ressource. Ainsi, $\Xi = \beta^* \cup R$.

θ est une variable utilisée pour définir à l'environnement d'éventuelles variables d'état supplémentaires et peut avoir n'importe quelle structure définie par le modélisateur.

De la même manière que [21], nous associons à chaque environnement physique $e \in D_E$ un certain nombre de fonctions permettant de donner la position d'un agent, de calculer le voisinage d'un agent ou de donner le contenu associé à un point ou à une partie de l'espace.

La fonction pos_e permet de donner la position d'un agent. Il est possible de donner un comportement générique à cette fonction étant donné que la position de chaque agent est déjà spécifiée dans son corps :

$$\begin{aligned} pos_e : A &\to P \\ a &\mapsto \beta_{a,e}.p \end{aligned}$$

La fonction $cont_e$ permet de donner le contenu (corps ou ressource) associé à un point ou à une partie de l'environnement :

$$\begin{aligned} cont_e : \wp(P) &\to \wp(\Xi) \\ \{p\} &\mapsto \{\xi \in \Xi \mid \xi.p = p\} \end{aligned}$$

Enfin, la fonction de voisinage ν_e permet à un agent de connaître les autres agents avec qui il peut interagir : $\nu_e : A \to \wp(\beta^*)$ Il est également possible de définir un comportement générique à la fonction de voisinage étant donné que la variable ψ présente dans chaque corps détermine la portée de perception de l'agent :

$$\nu_e : a \mapsto \{\beta_{i,e} \mid i \in A \setminus \{a\}, pos_e(i) \in \beta_{a,e}.\psi\}$$

Ce type d'environnement, que [9] catégorise en tant que *centralisé*, forme une structure unique dans laquelle l'ensemble des informations sont rassemblées. Deux autres types d'environnements sont distingués : les environnements *distribués* et les environnements *agents*. Il est tout à fait possible de proposer une version distribuée de notre environnement en couplant plusieurs environnements entre eux. Cependant, il faut mettre en place un mécanisme permettant l'échange des entités entre chaque sous-environnement.

Environnement social. Bien que notre spécification prenne en compte la notion d'environnement social, nous omettons sa spécification dans cet article. Bien qu'il puisse être représenté par les structures que nous venons de définir, nous avons dissocié l'environnement social afin de tirer partie de la topologie formée par les couplages entre modèles PDEVS. L'environnement social est alors responsable de créer et/ou détruire des couplages directs entre les systèmes cognitifs des agents en fonction de leur appartenance à une structure hiérarchique (groupe) et de leurs rôles. Ces couplages sont alors support de communication directe.

3.4 Interaction

L'interaction désigne une action réciproque permettant à deux agents d'échanger des informations, de manière directe ou indirecte. Le mode direct suppose qu'un agent échange des informations directement avec un autre, et le mode indirect, passe par un intermédiaire qui est l'environnement. Les communications directes passent par des messages PDEVS. Pour les communications indirectes (ou par *stigmergie*), elles s'effectuent à travers l'environnement via les influences (qui permettent de modifier l'état de l'environnement), et la perception des agents (qui permet de découvrir ces changements d'état).

Influence. La notion d'influence fait appel au mécanisme Influence/Réaction [22], qui permet de rendre l'environnement responsable de son état en évitant aux agents de modifier de manière directe.

Dans le modèle IRM4S [3], la structure d'une influence n'a pas fait l'objet de formalisation, elle représente une intention d'action, qui n'est rien d'autre qu'un message arbitraire envoyé à l'environnement. Néanmoins, une influence peut être matérialisée par le nom de l'action que l'agent souhaite effectuer, accompagnée d'éventuels paramètres. Nous donnons la structure générique d'une influence afin d'y inclure un certain nombre de paramètres utiles à la fois pour son traitement par la fonction $reac$ de l'environnement et pour le calcul de la perception.

Une influence $\gamma \in \Gamma$ est définie par le quintuplet suivant : $\gamma = (phase, n, p, a, \theta)$ où $phase \in \{pending, execute, satisfied, rejected\}$ représente l'état courant de l'influence, qui évolue lorsque celle-ci est traitée par l'environnement (plus particulièrement par la fonction $reac$) ; n est une variable qui représente le nom donné à l'intention d'action, par exemple $move$ pour se déplacer ; $p \in P$ est une position (physique ou sociale) au sein de l'environnement afin de matérialiser la cible de l'influence ; $a \in D_{Ag}$ est

l'identifiant de l'agent qui a initié l'influence ; enfin, θ est une variable permettant de définir d'autres paramètres qui peuvent être représentés par une structure arbitraire.

Perception. La spécification d'un système multi-agent basé sur un formalisme à évènements discrets tel que PDEVS pose le problème de la perception des agents. En effet, si un agent s'active à un temps de simulation précis et effectue une action entraînant un changement d'état au sein de l'environnement, les agents inactifs à cette date peuvent manquer ce changement.

La solution que nous proposons est d'envoyer par l'environnement à chaque agent toutes les variables ayant été modifiées et qui sont susceptibles de l'intéresser, autrement dit tout ce qu'il est capable de percevoir. Etant donné qu'une influence est ciblée à travers sa variable p, il est possible de calculer l'ensemble des agents concernés par celle-ci en utilisant la portée de perception/action ψ définie dans leur corps.

Soit une influence telle que définie dans la section précédente produite pour un environnement physique $\varphi \in D_E : \gamma = (phase, n, p, a, \theta)$. Pour chaque corps $\beta_{a,e} = (p_{a,e}, \psi_{a,e}, \theta_{a,e})$, l'agent a est concerné par l'influence γ si et seulement si $p_\gamma \in \psi_{a,e}$. Autrement dit si la cible $p_\gamma \in P$ de l'influence est incluse dans la partie $\psi_{a,e} \in \wp(P)$ représentant la portée de perception/action de l'agent. Les percepts des agents, que nous avons décrit par l'ensemble Pa tout au long de cet article est constitué d'un sous ensemble des influences ainsi que d'un sous-ensemble des ressources.

Nous proposons une définition générique sous forme de pseudo-code de la fonction de sortie d'un environnement physique, dont le rôle est de calculer les perceptions des agents (*cf.* section 3.3), juste après le calcul du nouvel état de l'environnement :

```
fonction λφ(phaseφ, σφ, Aφ, ((Pφ, distφ), Ξφ, θφ), Γφ, v+χ,φ) faire
  si phaseφ = perception alors
    pour chaque γi = (satisfied, ni, pi, ai, θi) ∈ Γ faire
      :
      pour chaque a ∈ {j ∈ Aφ | pi ∈ βj,φ.ψ} faire
        Ya ← νφ(a) ∪ βa, φ∪
             {r | r ∈ contφ(βa, φ.ψ), r ∈ R})
      fin pour
    fin pour
  sinon si phase = reqχ alors
    cf section 3.3
  fin si
fin
```

Cette approche permet de s'assurer que l'ensemble des agents sont en mesure de percevoir tout changement d'état de l'environnement sans que le modélisateur anticipe l'ensemble des perceptions possible.

4 Conclusion

Dans cet article, nous avons donné la spécification formelle d'un SMA en utilisant les outils de la TMS. Cette spécification, formalisée à l'aide de PDEVS, s'appuie sur des travaux existants et intègre un modèle d'action (IRM4S) et des propriétés (cf. section 2.1) reconnues dans la littérature pour favoriser la reproductibilité des résultats de simulation. DPDEMAS constitue un support permettant le partage de modèles SMAs. La sémantique opérationnelle étant bien définie, l'implémentation non-ambigüe d'un modèle est possible sur un outil capable d'exécuter des modèles PDEVS. Ces concepts ont été mis en œuvre à travers un modèle SMA de type proie-prédateur implémenté dans une plateforme de simulation PDEVS, que nous n'avons pas détaillé dans cet article faute d'espace disponible.

En perspective de ce travail, nous souhaitons : (1) valider l'approche en réalisant une étude comparative des résultats de simulation produits par des implémentations différentes d'un modèle spécifié à l'aide de DPDEMAS. (2) de proposer une bibliothèque de modèles d'environnements qui correspondent à différentes topologies (discrètes, continues, ...) afin de laisser au modélisateur la possibilité de les réutiliser en définissant uniquement la dynamique endogène et la réaction aux influences ; (3) de proposer un outil permettant de faciliter l'utilisation de cette spécification à travers la définition d'un DSL (Domain Specific Language) permettant de construire la structure des modèles.

Références

[1] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of Modeling and Simulation, 2nd Ed.*, 2nd ed., ser. Integrating Discrete Event and Continuous Complex Dynamic Systems. Orlando, FL, USA : Academic Press, 2000.

[2] R. Duboz, D. Vesmisse, G. Quesnel, A. Muzy, and E. Ramat, "Specification

of Dynamic Structure Discret event Multiagent Systems," in *Agent-directed simulation, part of the 2006 Spring Simulation Multiconference*. Huntsville, AL, USA, : SCS, Apr. 2005, pp. 23–30.

[3] F. Michel, "Le modèle IRM4S. De l'utilisation des notions d'influence et de réaction pour la simulation de systèmes multi-agents," *Revue dIntelligence Artificielle*, vol. 21, no. 5-6, pp. 757–779, 2007.

[4] V. Stodden, D. H. Bailey, J. Borwein, R. J. LeVeque, W. Rider, and W. Stein, "Setting the default to reproducible : Reproducibility in computational and experimental mathematics." 2013.

[5] R. Duboz, B. Bonté, and G. Quesnel, "Vers une spécification des modèles de simulation de systèmes complexes," *Stud. Inform. Univ.*, vol. 10, no. 1, pp. 7–37, 2012.

[6] K. Bajracharya and R. Duboz, "Comparison of Three Agent-based Platforms on the Basis of a Simple Epidemiological Model (WIP)," in *Proceedings of the Symposium on Theory of Modeling & Simulation - DEVS Integrative M&S Symposium*. San Diego, CA, USA : Society for Computer Simulation International, 2013, pp. 7 :1–7 :6.

[7] B. G. Lawson and S. Park, "Asynchronous Time Evolution in an Artificial Society Model," vol. 3, no. 1, Jan. 2000.

[8] J. Ferber, F. Michel, and J. Baez, "AGRE : Integrating Environments with Organizations," in *Environments for Multi-Agent Systems*, D. Weyns, H. V. D. Parunak, and F. Michel, Eds. Berlin, Heidelberg : Springer Berlin Heidelberg, 2005, pp. 48–56.

[9] J.-C. Soulié, "Vers une approche multi-environnements pour les agents," Ph.D. dissertation, Université de la Réunion, Dec. 2001.

[10] M. J. Wooldridge and N. R. Jennings, "Intelligent agents : theory and practice," *The Knowledge Engineering Review*, vol. 10, no. 02, pp. 115–152, Jun. 1995.

[11] H. Vangheluwe, "DEVS as a common denominator for multi-formalism hybrid systems modelling," in *IEEE International Symposium on Computer-Aided Control System Design, 2000. CACSD 2000*. IEEE, 2000, pp. 129–134.

[12] A. M. Uhrmacher and B. Schattenberg, "Agents in discrete event simulation," in *European Simulation Symposium*, 1998.

[13] A. M. Uhrmacher, J. Himmelspach, M. Röhl, and R. Ewald, "Introducing Variable Ports and Multi-Couplings for Cell Biological Modeling in DEVS," in *Simulation Conference, 2006. WSC 06. Proceedings of the Winter*. IEEE, Dec. 2006, pp. 832–840.

[14] A. Steiniger, F. Kruger, and A. M. Uhrmacher, "Modeling agents and their environment in Multi-Level-DEVS," in *2012 Winter Simulation Conference - (WSC 2012)*. IEEE, Dec. 2012, pp. 1–12.

[15] F. J. Barros, "The Dynamic Structure Discrete Event System Specification Formalism," *Transactions on Modeling and Computer Simulation*, vol. 13, no. 1, pp. 35–46, Mar. 1996.

[16] J.-P. Müller, "Towards a Formal Semantics of Event-Based Multi-agent Simulations," in *Multi-Agent-Based Simulation IX*, N. David and J. S. Sichman, Eds. Berlin, Heidelberg : Springer, 2009, pp. 110–126.

[17] G. A. Wainer and N. Giambiasi, "Application of the Cell-DEVS Paradigm for Cell Spaces Modelling and Simulation," *SIMULATION*, vol. 76, no. 1, pp. 22–39, Jan. 2001.

[18] R. Duboz, E. Ramat, and N. Giambiasi, "Utilisation du formalisme DEVS pour la spécification de systèmes d'agents réactifs," in *Dixièmes journées francophones sur les systèmes multi-agents*, Lille, France, Oct. 2002, pp. 99–102.

[19] A. Pnueli, "Specification and development of reactive systems," in *IFIP Congress*, 1986, pp. 845–858.

[20] A. S. Rao and M. P. Georgeff, "An abstract architecture for rational agents," in *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, USA, Oct. 1992, pp. 439–449.

[21] P. Mathieu, S. Picault, and Y. Secq, "Design patterns pour les environnements dans les simulations multi-agents," *Revue dIntelligence Artificielle*, vol. 30, no. 1-2, pp. 133–158, 2016.

[22] J. Ferber and J.-P. Müller, "Influences and Reaction : a Model of Situated Multiagent Systems," in *Proceedings of the Second International Conference on Multiagent Systems*, 1996.

Contrôle par apprentissage constructiviste et régulation du trafic coopératif

M. Guériau[a,b] maxime.gueriau@ifsttar.fr
F. Armetta[a] frederic.armetta@liris.cnrs.fr
S. Hassas[a] salima.hassas@liris.cnrs.fr
R. Billot[c] romain.billot@telecom-bretagne.eu
N-E. El Faouzi[b] nour-eddin.elfaouzi@ifsttar.fr

[a]Univ. Lyon, UMR CNRS 5205 LIRIS, F-69622 Villeurbanne, France

[b]LICIT, Univ. Lyon – IFSTTAR, LICIT, F-69675 Bron – ENTPE, LICIT, F-69518 Vaulx En Velin, France

[c]IMT Atlantique, Lab-STICC, Univ. Bretagne Loire, F-29238 Brest, France

Résumé

Lorsqu'un système autonome évolue dans un environnement complexe, en partie inconnu ou dynamique, il n'est pas possible de fournir une représentation exhaustive a priori facilitant son processus de prise de décision. Pour illustrer ce problème, nous choisissons le cas du contrôle décentralisé du trafic coopératif, où une unité d'infrastructure est en charge de réguler localement le flux, en envoyant des consignes aux véhicules connectés. Ce contrôle est le fruit d'une stratégie construite par l'apprentissage d'une représentation précise (états perception-action) des différents états de trafic. Nous proposons un modèle capable, sans connaissances expertes, d'utiliser un ensemble de méthodes de classification représentées sous la forme d'une population d'agents et de les combiner dynamiquement pour construire une représentation précise de l'environnement. Notre approche s'inscrit dans une démarche d'apprentissage constructiviste où la population d'agents construit collectivement une représentation qui exploite, suivant l'usage, les discrétisations possibles de l'espace de perception proposées par les individus.

Mots-clés : *Apprentissage constructiviste, prise de décision, contrôle*

Abstract

Decision making in autonomous systems is particularly challenging in unknown and changing complex environments, where providing a complete a priori representation is not possible. To illustrate the problem, we consider a decentralized control of road traffic, where a control device of the distributed infrastructure locally controls traffic by sending recommendation messages to connected vehicles. We propose an approach able to combine, without prior domain-knowledge, a set of existing traditional unsupervised learning methods that collaborate as a population of agents in order to build an efficient representation. Our approach follows a constructivist learning perspective, where the population of agents is able to collectively build a representation that dynamically combines discretizations from the individuals.

Keywords: *Constructivist learning, decision-making, control*

1 Contexte

La pertinence de la décision prise par un système autonome repose directement sur sa capacité à discriminer ses différentes interactions sensorimotrices. Cette compétence dépend de la précision de la représentation faite de l'environnement par le système mais aussi du problème visé. Dans les approches classiques en Intelligence Artificielle (IA), cette représentation est donnée au système *a priori* sous la forme de connaissances expertes. Dans le cas d'un environnement inconnu *a priori* et/ou dynamique, il est plus difficile de fournir une telle représentation. Le problème est donc plutôt de permettre au système de construire dynamiquement une représentation assurant une prise de décision efficace.

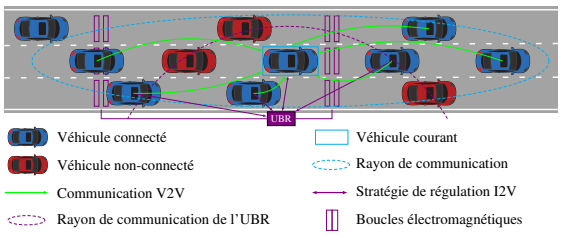


FIGURE 1 – Architecture générale des systèmes de transport intelligents coopératifs.

Comme exemple d'un environnement de prise de décision complexe, nous avons choisi le cas de la régulation du trafic routier dans le contexte des systèmes de transport intelligents coopératifs (C-ITS), illustré par la figure 1. Ces systèmes complexes s'appuient sur les avancées récentes en matière de communication et de technologies de l'information afin d'améliorer l'écoulement du flux de trafic. Dans un futur proche, l'infrastructure routière sera partagée par des véhicules connectés et non-connectés. Les véhicules connectés pourront, grâce à un protocole de communication sans fil, échanger des informations entre eux (véhicule à véhicule – V2V) et avec l'infrastructure (infrastructure à véhicule – I2V et véhicule à infrastructure V2I). Dans les C-ITS, les unités de contrôle côté infrastructure, nommées Unités de Bord de Route (UBR), ont pour rôle de collecter des données sur leur section dédiée. Plus que de simples relais d'information, une UBR peut aussi jouer le rôle d'unité de régulation décentralisée du trafic en propageant des consignes aux véhicules connectés par communication I2V.

1.1 Contributions

Le premier objectif de ce travail est de proposer un modèle permettant à un système autonome de construire une représentation à partir de ses interactions bas-niveau avec son environnement (section 2). Le modèle suit une perspective constructiviste et couple deux processus inter-dépendants : la construction d'hypothèses d'"états perception-action" et l'élaboration d'une "stratégie de contrôle". Ces processus sont sont gérés par une population d'agents en compétition pour construire une représentation précise de l'environnement (ensemble d'états perception-action) et d'un processus de prise de décision permettant de choisir une action pertinente. Ce dernier mécanisme est modélisé comme une forme d'apprentissage par renforcement où le retour sur expérience est fourni par l'environnement et utilisé pour renforcer les hypothèses d'état perception-action proposées par les agents. L'originalité de notre contribution est d'utiliser un ensemble de représentations concurrentes modélisées sous la forme d'agents autonomes qui s'adaptent dynamiquement grâce à des mécanismes d'intelligence collective, dans le but de produire une représentation émergente d'un contexte dynamique.

Le second objectif de l'article est de proposer une application innovante de notre modèle conceptuel. Nous avons choisi l'environnement du véhicule connecté où une unité décentralisée de l'infrastructure connectée (UBR) est un agent de contrôle du trafic qui implémente notre modèle. La partie 3 montre des résultats intéressants de notre travail avec une implémentation simplifiée du modèle. Les résultats montrent également des voies d'amélioration possibles du modèle.

1.2 État de l'art

Les approches classiques en IA reposent sur une abstraction de l'environnement proposée par le concepteur du système. Cette représentation dépend très fortement des spécificités du problème visé et de l'environnement associé (capteurs/effecteurs). Un système exploitant une telle représentation aura donc une capacité d'adaptation à d'autres problèmes/environnements limitée [3]. Comme l'autonomie d'un système dépend de sa capacité d'adaptation, les approches constructivistes proposent des mécanismes de construction itérative de la représentation de l'environnement par un agent, à partir de ses interactions. Ces approches s'inspirent des travaux précurseurs en sciences cognitives de Piaget [9] sur le développement de l'enfant, et ont convergé vers la définition de caractéristiques permettant de modéliser l'autonomie [12] : l'agent doit être incarné, situé, et exploiter un processus développemental épigénétique. La notion d'épigénétique à été proposée par Piaget, et décrit le processus de développement de l'individu à travers ses interactions avec son environnement. L'approche la plus prometteuse en IA développementale consiste à reproduire le processus d'acquisition de connaissance de l'être humain. Seuls quelques travaux novateurs [7] proposent de construire une représentation uniquement à partir des interactions sensorimotrices de l'agent. Ces approches se heurtent au problème de l'amorçage de la représentation, qui survient dès que l'on cherche à démarrer la construction de la représentation de l'environnement, notamment sans utiliser de connaissances expertes *a priori*. Pour contourner ce problème, quelques approches proposent d'aider le système en lui donnant quelques mécanismes de bas niveau, permettant d'amorcer le processus d'apprentissage [10]. Le modèle des auteurs utilise le gaz neuronal croissant (GNG) - une adaptation des cartes auto-organisées - afin de générer une première discrétisation de l'espace des perceptions du système. Cela permet de réduire le nombre de dimensions des données perçues (espace d'entrée) afin de simplifier l'identifica-

tion des actions les plus pertinentes correspondant à cet espace. Cependant, pour fournir une telle discrétisation, il faut réaliser un partitionnement des données perçues en amont du processus d'apprentissage du système. Cela s'oppose en partie au cadre strict du constructivisme où cette discrétisation devrait être le résultat des interactions avec l'environnement. Le système doit donc être en mesure de faire évoluer cette représentation. Une façon d'offrir cette possibilité est de fournir différentes discrétisations au système, qu'il devra combiner afin d'en déterminer les plus utiles à son usage (actions). Dans cet article, nous proposons des mécanismes permettant au système d'utiliser plusieurs représentations concurrentes, qui sont mises à jour en parallèle. Ces abstractions (discrétisations) des données brutes perçues peuvent être générées en amont de la phase d'apprentissage du système, à partir de différents algorithmes d'apprentissage (K-moyennes, GNG). D'un point de vue de l'ingénierie du trafic, les approches classiques de régulation sont définies comme un ensemble de règles expertes qui s'appliquent dans un contexte discret et prédéfini. Ce type de stratégies peut aussi bien s'appliquer au niveau du système [8] qu'à une échelle plus locale [4]. Les SMA sont largement utilisés pour traiter des problèmes de gestion et de régulation des systèmes de transport [2], mais la plupart des contributions s'intéressent à la coordination d'intersections. Le déploiement des véhicules connectés a tout de même suscité un intérêt plus récent vers le contrôle sur autoroute. Ces approches proposent des stratégies dynamiques exploitant les technologies basées sur la communication, comme les limitations de vitesse variables, et permettent des stratégies de régulation plus décentralisées.

2 Construction d'une représentation par apprentissage concurrent

L'objectif à long terme de notre modèle est de concevoir un système de prise de décision dynamique capable de définir les frontières de chaque état perception-action, afin d'identifier précisément le périmètre de chaque action étant donné un ensemble de perceptions. En effet, un tel système serait capable de mieux comprendre les dynamiques de l'environnement et aurait une probabilité plus élevée de sélectionner l'action la plus pertinente dans le contexte actuel estimé. Plusieurs problèmes sont à traiter afin d'arriver à un tel résultat. D'abord, il est nécessaire de choisir quelles sont les connaissances de base à fournir au système sans pour autant incorporer trop de connaissances expertes dans sa représentation. Ensuite, le périmètre exact des actions peut ne pas être atteignable selon le problème. La difficulté est de trouver la meilleure façon de modéliser cette notion de périmètre sans pour autant limiter l'ensemble des états possibles. Enfin, le défi le plus intéressant se situe au niveau du renforcement de la représentation, qui doit mener à un contrôle efficace de l'environnement par le système. Nous proposons un modèle conceptuel générique dont les mécanismes généraux traitent les problématiques précédentes, et qui peut être facilement adapté dans son implémentation pour traiter des problèmes spécifiques.

2.1 Aperçu

Le modèle proposé s'appuie sur des mécanismes d'intelligence collective afin d'assister le système dans sa tâche itérative de construction de sa représentation de l'environnement. Cette tâche prend la forme de deux processus couplés, perception et décision, comme illustré dans la figure 2. Les données d'entrée, collectées par

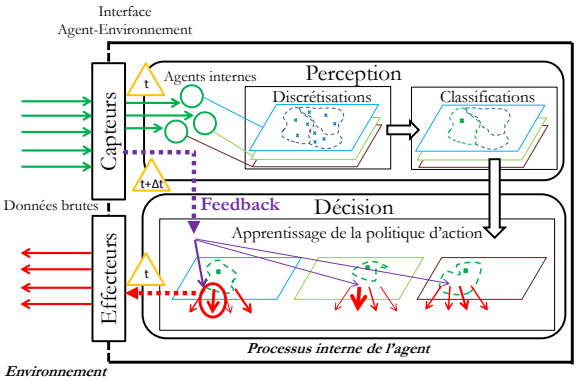


FIGURE 2 – Aperçu du modèle.

les capteurs, représentent des observations de l'environnement et constituent la perception de bas-niveau du système. Ces données sont continues et non-identifiées dans la mesure où aucune information experte n'est communiquée au système au sujet de la pertinence ou de l'importance de chaque variable perçue. Cette perception est transmise aux agents "discrétiseurs", dont le rôle est de fournir des abstractions à partir de ces informations. Ces abstractions correspondent à des façons alternatives d'interpréter le signal d'entrée. Chaque agent applique une stratégie de discrétisation interne. Les discrétisations générées peuvent évoluer en temps réel

pour adapter le périmètre des (états) classes que l'agent propose. Plusieurs mécanismes peuvent améliorer la précision des discrétisations proposées par les agents. Le processus multi-agent produit un ensemble de partitions concurrentes pour le système correspondant à différentes représentations possibles. La sélection de l'action repose sur un apprentissage par renforcement qui fait intervenir toutes les représentations concurrentes. Le modèle, présenté dans

Algorithme 1 Processus général du modèle

Instancier l'ensemble A des actions possibles
Instancier l'ensemble D des agents discrétiseurs
Instancier l'ensemble S des états sélectionnés concurrents (vide)
répéter *# boucle qui représente le processus de décision du système*
  si S n'est pas vide **alors** *# bloc non exécuté à la première itération*
    $F = feedback(P)$
    pour tout $s_i \in S$ **faire** *# récupérer le lien L entre l'état s_i et l'action a^**
      $L = getPercepActionLink(s_i, a^*)$
      $incrementReward(L, F)$
    fin pour
    $reinforce(D, a^*, F)$
  fin si
  $P = getPerception()$ *# début du processus de sélection de l'action*
  $S.removeAll()$
  pour tout $d_i \in D$ **faire**
    $p_i = discretizerPerception(P)$ *# récupération de la perception de l'agent*
    $d_i.discretize(p_i)$
    $s_i = D_i.classify(p_i)$
    $S.add(s_i)$ *# ajout de l'hypothèse d'état s_i à l'ensemble S*
  fin pour
  $a^* = actionSelection(S)$
  $execute(a^*)$
fin répéter *# fin du processus de décision avant la prochaine perception (et feedback)*

l'algorithme 1, vise à combiner dynamiquement les états perception-action précédemment appris afin de construire une représentation encore plus précise de l'environnement.

2.2 Perception de haut-niveau

Le système utilise des représentations individuelles proposées par ses agents discrétiseurs pour construire une représentation de plus haut-niveau. Chaque discrétiseur exploite son propre processus de classification exécuté dans son propre espace de perception, qui peut varier d'un agent à l'autre. Cela offre la possibilité d'évaluer des représentations dédiées à des canaux spécifiques de la perception. Ainsi, les discrétiseurs peut percevoir des données à partir des mêmes capteurs ou utiliser des dimensions différentes à l'échelle de la perception du système. Nous proposons de décrire le modèle de façon générique, ce qui permet de repousser le choix de la méthode de classification à l'étape d'implémentation du modèle. Les algorithmes en-ligne ou hors-ligne peuvent être utilisés à la conditions qu'ils soient compatible avec le type des données d'entrées et qu'ils permettent de générer une représentation sous la forme d'un ensemble d'états discrets. L'ensemble des variables perçues par chacun des agents peut être défini de manière experte ou initialisé aléatoirement, en fonction du problème visé. Le processus de perception du système peut se résumer en deux étapes consécutives (étapes 1 et 2), telles que représentées dans la figure 3. Chaque fois que le système perçoit son environnement, tous les agents discrétiseurs D mettent à jour en parallèle leur propre espace d'hypothèses d'état, en fonction de leur méthode de classification choisie (K-means, GNG, etc.). Par exemple, la discrétisation de l'agent D_i est composée de l'ensemble des hypothèses d'état : $\{S_1, S_2, ..., S_j\}$. Pour chaque pas de temps de perception du système, chaque agent D perçoit ses variables d'entrée et classe sa perception comme une hypothèse d'état discret. A partir de l'ensemble des hypothèses d'état (une par agent), le système tente de choisir l'action la plus pertinente dans le contexte défini par les différents agents.

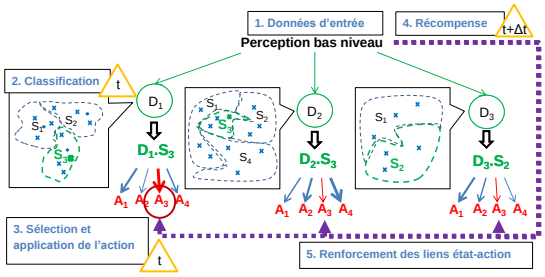


FIGURE 3 – Processus de discrétisation, de classification et de sélection d'action.

2.3 Contrôle de l'environnement

Les discrétiseurs mémorisent la probabilité de sélection de chaque action lorsqu'ils classent

leur perception en tant qu'une hypothèse d'état donnée. Ces probabilités sont modélisées sous la forme de liens état-action qui permettent de garder une trace au niveau du système du résultat des actions consécutives, évalué à travers le retour sur expérience (*feedback*) fourni par l'environnement. A l'issue de l'étape de perception, tous les discrétiseurs ont sélectionné une de leurs hypothèses d'état en réponse à la perception de bas-niveau propagée. Le système doit désormais choisir une action qui soit la plus pertinente possible en fonction des différentes hypothèses proposées (une par agent). Le système doit alors trouver un compromis entre l'exploitation (appliquer l'action générant le meilleur *feedback*) et l'exploration (tester des actions non utilisées jusque là). Ce choix est illustré par les étapes 3 à 5 dans la figure 3. Le système fait face à un dilemme classique d'exploration-exploitation [1], puisque l'espace des actions est supposé fini et qu'il n'y a pas de corrélation évidente entre des hypothèses d'état distinctes. Cette situation fait écho au problème dit du bandit à n bras, relatif à l'élaboration d'une stratégie optimale de sélection et de test de différentes machines à sous dans un casino. Plusieurs stratégies, disponibles dans la littérature, peuvent être utilisées dans notre modèle. La plupart utilisent un paramètre objectif à minimiser basé sur la notion de regret, exprimé comme la différence entre la récompense courante et la récompense optimale (théoriquement inconnue). Notre modèle ne nécessite pas que la stratégie soit optimale dans la mesure où la sous-exploration de l'espace des actions risque de nuire aux actions prises par le système. Néanmoins, cette hypothèse dépend encore une fois du type de problème visé et pourra être étudiée en comparant différentes implémentations du modèle.

Le retour sur expérience fourni au système fait partie de sa perception de bas-niveau et doit être défini lors de la formalisation du problème (*i.e.* pour l'implémentation). Cette fonction de *feedback* permet une évaluation par le système du résultat de ses actions dans l'environnement. Cette valeur est utilisée comme une récompense pour l'algorithme d'exploration-exploitation et contribue au renforcement de la représentation construite. Le résultat de la sélection est un lien état-action dont chaque état est directement lié à un agent discrétiseur. Ainsi, le processus de renforcement contribue à l'apprentissage du système, en l'aidant à identifier les hypothèses d'état les plus précises parmi l'ensemble des discrétisations proposées par la population d'agents discrétiseurs.

3 Application au trafic coopératif

Le modèle générique présenté dans cet article a été implémenté dans le cadre d'une application innovante : la régulation du trafic coopératif. Le système prend place dans une unité de bord de route, chargée de réguler dynamiquement un flux de trafic. Pour cela, il est équipé de capteurs permettant de percevoir les véhicules connectés et les informations du flux (débits/vitesses). Un agent discrétiseur est associé à chacun de ces capteurs. Le cadre de simulation utilisé pour les expérimentations est présenté dans [5]. Nous donnons ici les détails de l'implémentation de notre modèle de contrôle : description des caractéristiques des agents discrétiseurs et actions/*feedback* utilisés. Ensuite, les résultats en simulation mettent en avant le comportement du modèle et confirment les bénéfices de notre approche qui, grâce à des représentations concurrentes, permet de réguler un flux partiellement composé de véhicules connectés.

3.1 Cas d'étude

Nous avons choisi le cas d'étude du contrôle du trafic coopératif pour implémenter notre modèle et étudier les bénéfices de notre approche. Nous visons une situation de trafic mixte où le flux est composé de véhicules connectés et non-connectés. Les véhicules connectés peuvent échanger des informations entre eux (communication V2V) et avec l'infrastructure (I2V/V2I). L'infrastructure est composée d'unités de bord de route (UBR), en charge de réguler le trafic sur leur section dédiée. Une UBR perçoit des informations lui permettant d'estimer l'état de trafic courant grâce à ses capteurs (par exemple, des boucles électromagnétiques qui permettent d'estimer le débit et les vitesses sur chaque voie). L'objectif d'une UBR est de propager des consignes dans des messages à destination des véhicules connectés. Une telle stratégie de contrôle décentralisée vise à réduire ou à retarder le phénomène de congestion.

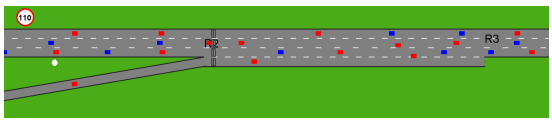


FIGURE 4 – Capture d'écran du scénario dans le simulateur.

Comme illustré par la figure 4, toutes les expérimentations sont menées dans un même scénario représentant une section autoroutière rectiligne

à trois voies avec la présence d'un convergent (voie d'insertion) vers sa fin. L'UBR (représentée par le disque blanc) perçoit les débits et les vitesses grâce aux capteurs sur la route (rectangles gris) ainsi que les vitesses des véhicules connectés dans son rayon de communication (150 mètres). L'unité de bord de route implémente le modèle afin de choisir les consignes les plus pertinentes à envoyer aux véhicules connectés pour éviter, ou au moins réduire, la congestion sur sa section. Les consignes sont propagées sous la forme de messages I2V aux véhicules connectés. La voie d'insertion, à cause de la différence de vitesse avec la section principale, déclenche, suivant les conditions de trafic dense, l'apparition de congestion qui se propage depuis la voie de droite jusqu'à l'ensemble des voies. Nous étudions l'effet de la stratégie de régulation d'une UBR sur le flux en faisant l'hypothèse d'un pourcentage fixé à 30% de véhicules connectés, mélangés de façon homogène parmi les véhicules non connectés sur la section principale.

Simulation. Le scénario choisi pour la simulation est modélisé dans le simulateur MASCAT [5] (*Multi-Agent Simulator for Connected and Automated Traffic*), une extension multi-agent du simulateur de trafic microscopique MovSim [11]. Cette plateforme de simulation permet d'instancier différents modèles microscopiques (nommés "lois de poursuite") qui décrivent le comportement longitudinal des véhicules. Le réseau routier est représenté sous la forme d'un graphe qui permet de modéliser les sections multi-voies, les limitations de vitesses et les connexions entre les sections. Notre extension permet d'ajouter la communication entre les véhicules et l'infrastructure. Un protocole basé sur l'envoi de message permet à toutes les entités, modélisées comme des agents autonomes, de partager des informations et d'adapter leur comportement.

Scénarios. Afin de générer différents états distincts pour l'unité de bord de route, nous avons déterminé les flux d'entrée (section principale et voie d'insertion) à partir de données réelles issues de boucles électromagnétiques. Le jeu de données est composé d'un mois (juin 2001) de données boucles, récoltées entre 8h et 9h sur l'autoroute A6. Nous n'avons conservé que les données de deux boucles, choisies pour la correspondance entre le réseau réel et celui de notre scénario (figure 4). Cela a permis de produire 30 simulations de 60 minutes chacune. La diversité des situations de trafic simulées permet d'étudier le comportement du modèle dans des contextes sensiblement variables (trafic fluide, régime critique ou congestionné).

3.2 Définition des comportements

Afin de reproduire les dynamiques du flux de trafic, des lois de poursuite modélisent le mouvement longitudinal des véhicules. Ce comportement de base est modifié par les messages de consignes reçus par les véhicules connectés.

Unité de Bord de Route. Le résultat de la stratégie de contrôle de l'UBR est le choix, à chaque pas de temps de décision (fixé à 120 secondes), d'une action parmi les suivantes :

- A_1 : pas d'envoi de consigne.
- A_2–A_3 : consigne de changement de voie (droite à gauche – gauche à droite).
- A_4–A_5 : consigne de modification d'interdistance (1,8 s–1,2 s).
- A_6–A_7 : consigne de limite de vitesse (110 km/h–50 km/h).

Toutes les actions mènent à la propagation d'un message de consigne (sauf A_1) contenant le type de consigne et le paramètre associé aux véhicules connectés. Ces messages définissent aussi une zone de pertinence (qui couvre l'ensemble de la section jusqu'à la voie d'insertion) ainsi qu'une date d'expiration (fixée elle-aussi à 120 secondes). La consigne contenue dans un message est appliquée par le véhicule qui l'a reçue tant qu'il se trouve dans la zone de pertinence du message et que celui n'a pas expiré.

Véhicules. Le comportement longitudinal des véhicules connectés et non-connectés est régi par une loi de poursuite. Nous avons choisi *Intelligent Driver Model* (IDM), qui est un modèle sans collision capable de reproduire les instabilités du flux de trafic [11]. La décision de changement de voie est modélisée grâce à la stratégie opportuniste nommée MOBIL [6]. Seuls les véhicules connectés sont équipés de dispositifs de communication sans fil leur permettant de percevoir les messages de consigne dans un rayon de 150 mètres, ce qui correspond aux technologies actuelles. Nous ne considérons par les messages partiellement reçus ou les pertes d'informations dans la mesure où le protocole réel est sensé les traiter. Les messages de consigne reçus par les véhicules connectés sont retransmis aux autres véhicules (à portée) jusqu'au début de la section (*i.e.* tant qu'ils se trouvent dans la zone de pertinence du message).

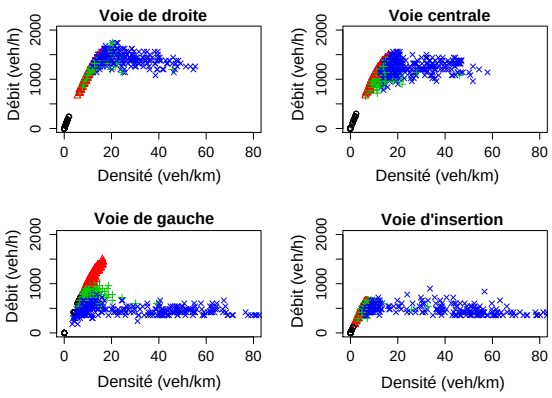
Les véhicules connectés affichent dans leur interface embarquée la consigne contenue dans les messages, afin que le conducteur puisse l'intégrer à son comportement. Nous proposons donc une interprétation par le véhicule de chacune des consignes. Pour les actions A_2 et A_3, la consigne de changement de voie modifie la stratégie opportuniste tant que le message reste pertinent (zone et durée). Le véhicule va donc essayer de rejoindre la voie la plus à gauche (respectivement à droite) en entreprenant un ou des changement(s) de voie jusqu'à atteindre cette voie cible. Le modèle de changement de voie intègre une estimation du créneau spatio-temporel disponible. Cette estimation permet au véhicule de décider si un changement de voie est suffisamment sûr (au sens de ces paramètres) pour être effectivement exécuté. Cela implique qu'une plus grande proportion de véhicules connectés changera de voie lorsque le trafic sera fluide (et inversement lorsque le trafic sera dense). Pour les actions A_4 à A_7, la seule modification consiste à changer temporairement les paramètres correspondants dans le modèle longitudinal (respectivement, l'interdistance désirée et la vitesse désirée) pour les valeurs indiquées dans le message de consigne. Toutes ces modifications sont maintenues tant que le message reste pertinent et sont automatiquement annulées dès que le message expire ou que le véhicule quitte la zone de pertinence définie.

3.3 Implémentation du modèle

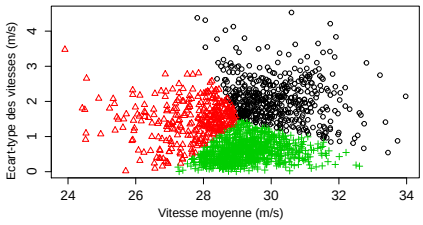
Le modèle présenté dans cet article est implémenté en tant que processus de décision d'une unité de bord de route. Cette UBR est capable de percevoir par l'intermédiaire d'un capteur de type électromagnétique le débit moyen, la concentration et les vitesses sur la voie d'insertion et sur les trois voies de la section principale. En plus, les véhicules connectés situés dans son rayon de communication partagent leurs vitesses respectives. L'ensemble de ces données constitue la perception bas-niveau du système. La perception et la décision du système sont ici discrétisées et exécutées toutes les 120 secondes. L'UBR perçoit des données à partir de ses capteurs, agrégées sur la période précédente de 120 secondes, et choisit et applique immédiatement une action (*i.e.* envoie une consigne). Après une période de 120 secondes supplémentaires, la consigne expire et l'UBR reçoit, au même moment, un *feedback* utilisé pour renforcer la représentation et une nouvelle perception de l'environnement.

Agents discrétiseurs. Afin de montrer les bénéfices du modèle, nous proposons une implémentation avec deux agents discrétiseurs. Chaque agent a pour rôle de générer une représentation (discrétisation) à partir des variables qu'il perçoit. Nous avons décidé de lier chaque discrétiseur, nommés D_1 et D_2, à un capteur spécifique. Les agents diffèrent donc principalement par leur perception. Ainsi, D_1 perçoit les données des boucles électromagnétiques (débit et densité moyens) pour chacune des trois voies et la voie d'insertion, agrégées toutes les 120 secondes. D_2 exploite les données provenant du capteur de communication de l'UBR et construit un modèle (moyenne et écart-type) de la distribution estimée des vitesses à partir des messages des véhicules connectés qui sont passés dans son rayon de communication durant son dernier pas de temps de décision. Les discrétisations des agents sont générées à partir d'une classification par l'algorithme des K-moyennes (stabilisé avec 10000 itérations à partir d'initialisation aléatoires) pour respectivement 4 et 3 classes (D_1 et D_2). Les résultats de la discrétisation par les deux agents sur les données issues des 15 premières simulations sont représentés sur la figure 5. L'apprentissage des états perception-action de chaque agent est réalisé sur les 15 mêmes simulations, en exploitant le *feedback* du système.

Renforcement de la représentation. Les récompenses recueillies par les agents sous la forme de liens état-action sont calculées à partir d'une variable de l'environnement choisie. Dans le scénario proposé, nous avons retenu la vitesse moyenne la plus faible (sur les 4 boucles électromagnétiques) en tant que *feedback* du système, puisque maximiser cette valeur correspond à l'objectif global d'amélioration de l'écoulement du flux. Ce retour sur expérience externe, est utilisé sous la forme de récompenses dans l'algorithme d'exploration-exploitation pour renforcer à la fois la stratégie de contrôle et les représentations proposées. Parmi les algorithmes de bandit à n bras de la littérature, notre choix s'est arrêté sur *Upper Confidence Bound* [1] (UCB). Son initialisation consiste à essayer chaque machine une seule fois. Ensuite, l'algorithme sélectionne systématiquement la machine j qui maximise $\ddot{x}_j + \sqrt{\frac{2 \ln n}{n_j}}$ où $\ddot{x}_j$ correspond à la récompense moyenne obtenue pour la machine j, n_j est le nombre de fois que la machine j à été jouée jusqu'ici et n est le nombre total d'essais. La mise à jour des récompenses concerne tous les liens



(a)



(b)

FIGURE 5 – Discrétisations générées par (a) D_1 [8 dimensions] et (b) D_2 [2 dimensions] sur le jeu de données d'apprentissage.

perception-action (notés L dans l'algorithme 1), et non pas seulement celui qui a été effectivement sélectionné par l'algorithme. De cette façon, le bénéfice d'une action est également propagé aux liens qui ont contribué à sa sélection. Cette récompense permet d'accélérer la phase d'exploration, en s'appliquant à autant de liens que d'agents discrétiseurs pour chaque itération.

3.4 Expérimentations

Les expérimentations sont menées en deux étapes principales. Dans un premier temps, un ensemble de données d'entrée (la perception des agents D) est nécessaire afin d'appliquer les algorithmes de classification et obtenir des discrétisations stables (sans appliquer les actions). Puis, un jeu de simulations est utilisé pour l'apprentissage des liens perception-action en utilisant le processus complet d'exploration et d'application des actions (sur les 15 simulations). La seconde étape confronte le système à des scénarios alternatifs, pour vérifier que la représentation construite par le système l'aide à appliquer une stratégie de régulation efficace sur le flux. Cette étape est menée sur la deuxième partie des simulations. La situation témoin de chaque simulation est l'observation d'un ensemble d'indicateurs dans le même scénario mais sans utiliser l'infrastructure (aucune consigne n'est envoyée). Nous comparons trois implémentations du modèle : le cas avec un seul agent (D_1 et D_2 séparément), et une combinaison dynamique des représentations de ces deux agents grâce au modèle.

Indicateurs. Afin d'étudier les effets des différentes stratégies, nous avons sélectionné trois indicateurs : le temps total passé (TTS), la vitesse moyenne sur la section et le pourcentage de congestion. Le TTS est simplement la somme des temps de trajet de tous les véhicules sur la section ; des valeurs plus faibles indiquent un meilleur écoulement. La vitesse moyenne permet d'observer, au cours du temps, l'homogénéité du flux. Le pourcentage de congestion est la proportion spatiale de la section où les véhicules circulent à moins de 30 km/h. On peut s'attendre à ce que l'implémentation du modèle profite des représentations individuelles proposées par les agents et mène à l'amélioration de l'écoulement du flux, observé au regard de ces indicateurs.

Résultats. Parmi les 15 simulations disponibles, nous en avons sélectionné 3 différentes qui présentent des situations de trafic différentes. La figure 6 présente le tracé des trois indicateurs sur ces simulations. Les indicateurs permettent d'évaluer les effets des différentes implémentations du modèle, en les comparant avec la simulation témoin (courbes rouges). Dans les simulations (a) et (c), sans régulation, les instabilités générées par les changements de voies depuis la voie d'insertion se propagent à l'ensemble du flux. Les trois simulations montrent des comportements différents du modèle. Dans la simulation (a), la stratégie de contrôle de l'agent D_1 permet d'éviter l'apparition de congestion alors que celle de l'agent D_2 donne de moins bon résultats que dans la simulation témoin. Ces résultats sont bien visibles en observant le tracé des TTS. Des valeurs faibles de temps total passé indiquent que les véhicules parcourent la section plus rapidement en moyenne. La combinaison dynamique de D_1 et D_2 produit une stratégie mixte qui donne une amélioration mitigée. La simulation (b) illustre un cas spécifique où l'un des deux discrétiseurs propose une mauvaise représentation (D_2), puisque l'on n'observe pas de congestion dans la simulation témoin. Néanmoins, le modèle est capable de

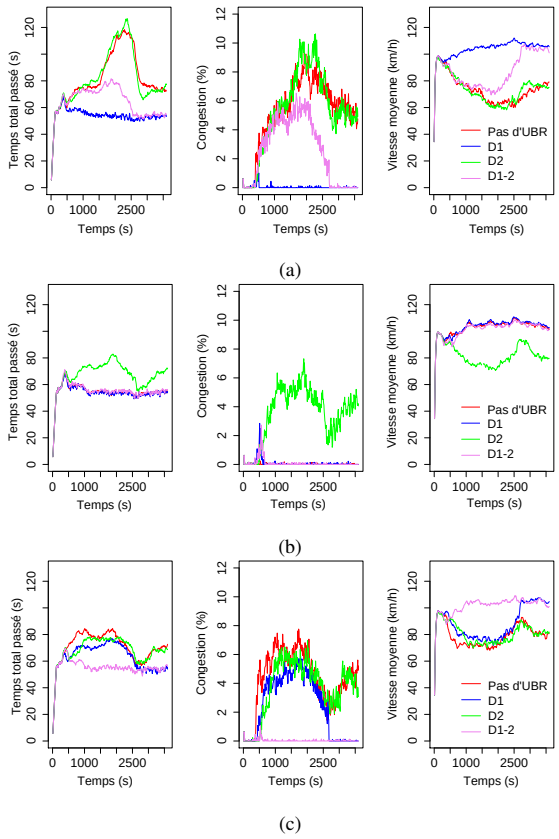


FIGURE 6 – Résultats des 3 simulations : (a) Lundi 18, (b) Jeudi 21 and (c) Mardi 28.

converger vers les meilleures représentations individuelles (D_1) dans ce contexte, ce qui permet de ne pas empirer la situation observée dans la simulation témoin. Dans la figure 6 (c), l'état de trafic modélisé se rapproche du régime critique du flux, puisque l'ampleur de la congestion est limitée et la vitesse moyenne ne diminue que peu (comparé à la simulation (a)). Dans ce contexte, les deux stratégies apprises par chaque discrétiseur ont un impact positif sur le flux, et l'amélioration de la représentation par le modèle est visible. Le système est capable, grâce aux récompenses exploitant le *feedback*, de construire une représentation de plus haut niveau à partir de celles des agents, qui permet d'éviter la propagation de congestion sur la section. La figure 7 représente de façon simplifiée la contribution de chaque discrétiseur (en reprenant le code couleur de la figure 6) à la stratégie de régulation globale du système. Le schéma montre que des états perception-action des deux agents sont utilisés par le modèle. L'action A_2, la plus utilisée, semble logique puisqu'elle permet de

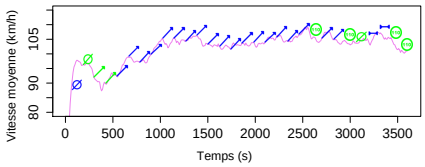


FIGURE 7 – Répartition entre les agents (D_1 en bleu, D_2 en vert) des consignes choisies par le modèle dans la simulation (c).

laisser plus de place aux véhicules entrant pour s'insérer dans le flux principal. Ainsi, les résultats confirment la capacité du modèle à utiliser différentes représentations concurrentes et à les combiner en une représentation plus précise.

Discussion. Comme dans la plupart des problème d'apprentissage par renforcement, le choix du *feedback* est critique et complexe. De plus, comme le système exploite différentes discrétisations concurrentes, le choix des données d'entrée (perception) est important. Cela est du à une problématique de fusion de capteurs : certains phénomènes sont perceptibles ou améliorables en utilisant seulement certaines dimensions. Cela signifie que le système doit percevoir des variables pertinentes pour son usage pour être capable d'adapter sa stratégie de contrôle aux variations de l'environnement. Dans les expérimentations, nous avons vu que le concepteur doit fournir un premier ensemble de discrétiseurs capables d'identifier des états pertinents, avant même de les faire combiner par le modèle. Ce problème pourrait être traité par l'ajout de mécanismes permettant au système de créer de manière autonome de nouveaux agents, voire de les faire évoluer dynamiquement, en exploitant son expérience d'interaction.

Comme les données d'entrée peuvent provenir de différentes dimensions (ou variables), une possibilité est de donner une perception complète au système. Dans ce cas, lors de l'implémentation du modèle, le temps d'exploration de l'ensemble des combinaisons de toutes les variables pour chaque discrétisation risque d'être très important. C'est pourquoi nous proposons d'utiliser différents agents qui ne travaillent que sur un sous-ensemble de la perception de bas-niveau. Les résultats obtenus en simulation laissent entrevoir le potentiel de convergence du système (en termes de profils de liens perception-action) vers les combinaisons de variables les plus pertinentes pour l'usage du système (*i.e.* au regard de la stratégie de

contrôle produite). Cela pourra être particulièrement utile pour des problèmes où la combinaison des variables d'entrée n'est pas connue *a priori*.

4 Conclusions et perspectives

Nous avons proposé un modèle générique permettant à un système de construire une représentation de son environnement à partir de ses interactions. Par un apprentissage constructiviste visant à construire itérativement le processus de prise de décision d'un système de contrôle, le modèle utilise une population interne d'agents discrétiseurs pour amorcer la construction de la représentation. La représentation qui en est le fruit se compose d'états perception-action qui évoluent dans un apprentissage par renforcement exploitant un retour de l'environnement afin d'explorer l'espace de recherche. Après avoir présenté le modèle d'un point de vue théorique, nous avons proposé une application dans le cas d'étude de la régulation de trafic coopératif. L'objectif est d'utiliser l'infrastructure et les véhicules connectés pour proposer un système d'aide à la décision autonome. Ce cadre applicatif innovant met en valeur les effets de notre approche. Les résultats obtenus en simulation montrent que la combinaison dynamique des discrétisations individuelles permet au système d'adopter une stratégie de régulation plus efficace.

La prochaine étape de notre travail se concentrera sur les évolutions du modèle. La construction du modèle conceptuel est aussi itérative, afin de pouvoir tester et valider chacun des mécanismes par l'implémentation à chaque étape. Nous cherchons désormais à concevoir un nouveau type d'agent capable d'exploiter les hypothèses d'état proposées par les discrétiseurs en proposant des associations entre ces états. Dans un premier temps, deux types d'associations seront considérées : la spécialisation ou l'agrégation. La question du compromis entre le temps nécessaire à l'exploration et le nombre d'associations reste à étudier.

Références

[1] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3) :235–256, 2002.

[2] Ana LC Bazzan and Franziska Klügl. A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review*, 29(03) :375–403, 2014.

[3] Rodney A Brooks. Intelligence without representation. *Artificial intelligence*, 47(1) :139–159, 1991.

[4] Rodrigo Castelan Carlson, Ioannis Papamichail, and Markos Papageorgiou. Local feedback-based mainstream traffic flow control on motorways using variable speed limits. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4) :1261–1276, 2011.

[5] Maxime Guériau, Romain Billot, Frédéric Armetta, Salima Hassas, and Nour-Eddin El Faouzi. Un simulateur multiagent de trafic coopératif. In *23es Journées Francophones sur les Systèmes Multi-Agents (JFSMA'15)*, pages 165–174. Cépaduès, 2015.

[6] Arne Kesting, Martin Treiber, and Dirk Helbing. General lane-changing model mobil for car-following models. *Transportation Research Record : Journal of the Transportation Research Board*, 1999(1) :86–94, 2007.

[7] Jonathan Mugan and Benjamin Kuipers. Autonomous representation learning in a developing agent. In *Computational and Robotic Models of the Hierarchical Organization of Behavior*, pages 63–80. Springer, 2013.

[8] Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12) :2043–2067, 2003.

[9] Jean Piaget. The construction of reality in the child. *Journal of Consulting Psychology*, 19(1) :77, 1955.

[10] Jefferson Provost, Benjamin J Kuipers, and Risto Miikkulainen. Self-organizing distinctive state abstraction using options. In *Proc. of the 7th International Conference on Epigenetic Robotics*, 2007.

[11] Martin Treiber and Arne Kesting. *Traffic flow dynamics : data, models and simulation*. Springer, 2013.

[12] Jordan Zlatev and Christian Balkenius. Introduction : Why "epigenetic robotics" ? In *Proceedings of the First Conference on Epigenetic Robotics. In : Balkenius, C., Zlatev, J., Kozima, H., Dautenhahn, K., Breazeal, C.(eds.) : Lund University Cognitive Science Series*, number 85, 2001.

Les identités au centre de la participation des agents à des actions collectives

E.-M. Khalfi, J.-P. Jamont, M. Occello
{prenom.nom}@lcis.grenoble-inp.fr

Univ. Grenoble Alpes, Grenoble INP, LCIS, F-26000 Valence, France

Résumé

Avec le développement des objets connectés, les agents embarqués déployés dans des environnements physiques et les applications multi-agents qui les impliquent deviennent de plus en plus populaires. Ces systèmes multi-agents sont amenés à partager le même environnement physique. Cette cohabitation d'agents de systèmes différents, qui n'ont pas nécessairement été prévus pour interagir entre eux par les concepteurs, les amène cependant à se solliciter. Un agent peut alors participer à la réalisation d'objectifs incompatibles avec les siens ou ceux de ses collectifs. Pour éviter ces situations, nous proposons un modèle d'agent basé sur les identités pour l'aider à décider de sa participation ou non à des actions collectives.

Mots-clés : *Modèle d'agents, identité personnelle, identités collectives, SMA embarqués.*

Abstract

The increasing number of smart objects and applications that collectively involve them leads multi-agent systems to collocate in the same physical environment. This agents cohabitation, that have not been planned to interact, requires the integration of cooperation mechanisms, since interactions are increasingly becoming standardized so agents of distinct systems can understand each other. Before committing to cooperate with others, an agent must take into account that it may be involved in the achievement of objectives that are incompatible with its own objectives, or with those of its teams. To avoid such situations, we propose an agent model which consists in reasoning about "identities" in participation to collective actions.

Keywords: *Agent model, embedded MAS, personal identity, social identity.*

1 Introduction

Les approches multi-agents sont de plus en plus utilisées dans la conception d'applications reposant sur l'utilisation d'objets connectés. Le développement massif de ces objets (26 milliards en 2020 selon Gartner), amène ainsi les agents de différents Systèmes Multi-Agents (SMA) à coexister dans un même environnement. L'environnement d'un SMA embarqué n'appartient donc plus à lui seul : il peut être partiellement ou complètement partagé avec d'autres systèmes.

Dans la plupart des applications multi-agents, les agents sont conçus pour interagir avec les agents d'un même système. Leurs interactions sont menées dans des cadres clairement prédéfinis lors de leur conception. Même s'ils ne partagent pas nécessairement les mêmes modèles et architectures, ils sont souvent interopérables du fait des normes et des standards, au niveau technique (interfaces et protocoles de communication), syntaxique (codage et formats de données) et sémantique (ontologies) [1].

Du fait de son autonomie, lorsqu'un agent n'a pas les ressources et les compétences nécessaires pour atteindre seul ses objectifs, il cherche à coopérer. Pourtant, la notion d'appartenance à un système n'est souvent pas explicite, il cherchera donc à coopérer avec les agents de son voisinage, qu'ils soient membres du même SMA ou pas.

Dans cet article, nous nous intéressons aux propriétés et aux éléments sur lesquels un agent peut raisonner pour décider de sa coopération avec d'autres agents dans un contexte multi-SMA. Notre objectif est de doter l'agent de la capacité de raisonner sur ses potentiels engagements à coopérer, afin d'éviter les situations d'interaction interférant négativement dans la réalisation de ses objectifs.

L'article est organisé comme suit. Nous nous intéressons dans un premier temps aux interférences qu'un agent peut rencontrer lors des interactions sociales coopératives dans un contexte multi-SMA (section 2). Ensuite, nous introduisons la théorie de l'identité qui fournira le cadre théorique de notre contribution (section 3). La section 4 s'appuie sur un scénario que nous utilisons dans le projet ANR ASAWoO

qui nous permettra de mettre en évidence des limitations de certains modèles. Nous décrivons ensuite le modèle IDEAS qui utilise différents types d'identités pour permettre aux agents de décider plus prudemment quant à leur engagement dans des actions collectives (section 5).

2 L'interférence dans le cadre de la coopération

Dans notre travail, nous considérons que "*des agents coopèrent s'ils s'engagent dans une action commune après avoir identifié et adopté un but commun*" [2]. Cette notion d'engagement est d'ailleurs souvent reprise dans la définition de concepts connexes comme la délégation de tâche [3]. Cet engagement est un acte social par lequel l'agent répond à une sollicitation pour accomplir une tâche pour un autre agent. Comme dans le cadre des réseaux de dépendances, l'engagement peut être vu comme un comportement permettant la structuration de groupes et d'organisations sociales. Un agent ne s'engage pas à participer à l'exécution de n'importe quelle action collective, mais seulement à celles qui sont réellement pertinentes pour lui et/ou pour les collectifs auxquels il appartient.

Un agent doit tenir compte du possible ajustement de ses objectifs à ceux de l'agent qui l'a sollicité. Les buts semblables ou compatibles sont en effet une condition nécessaire pour l'établissement des situations d'interactions coopératives [3]. De plus, pour que les actions et les interactions sociales menées en vue de réaliser un objectif partagé soient réellement qualifiées de coopératives [4, 5], il est important qu'elles soient intentionnelles.

Les agents déployés dans des environnements multi-SMA sont amenés à former des collectifs spontanés, à s'engager dans des coopérations temporaires avec d'autres agents qui peuvent être "inconnus". De plus, ces collectifs impromptus sont souvent formés d'une manière opportuniste, ce qui rend plus difficile l'application de schémas de coopération préétablis. Afin de prendre des décisions concernant la coopération avec d'autres agents, et s'engager à participer à des actions conjointes, les agents ont besoin de s'assurer que leurs partenaires potentiels n'interfèreront pas négativement dans la réalisation de leurs objectifs. Des interférences peuvent être cependant positives ou combiner des aspects positifs et négatifs [6].

Une interférence est dite *positive* pour un agent lorsque l'exécution des actions d'un autre agent favorise la réalisation ou la préservation de ses objectifs. Ceci peut se produire lorsqu'il participe à exécuter des actions faisant partie du plan d'un autre agent, ou lorsqu'il adopte un de ses objectifs. Les interférences sont dites *négatives* pour un agent lorsque l'exécution des actions d'un autre menace la réalisation de ses objectifs.

L'interférence *combinée* est la plus complexe à appréhender pour un agent (ou à approcher par un concepteur) car les agents ont généralement plusieurs buts de différents types et niveaux d'abstraction. La réalisation d'une action coopérative peut alors être une interférence à la fois négative et positive. Si un but est, par exemple, compatible au niveau individuel (interférence positive), il peut être incompatible au niveau collectif (interférence négative). Ainsi, un agent peut approuver une interférence négative, comme il peut désavouer une interférence positive. Ainsi un agent contrôlant l'ouverture d'une fenêtre considérerait la demande d'ouverture pour aérer une pièce comme une interférence positive vis-à-vis l'objectif de réguler la température d'une chambre, alors qu'il la considérerait comme négative vis-à-vis l'objectif d'assurer la sécurité de toute la maison.

Dans le cadre de ces interférences, un agent a intérêt à favoriser les interférences positives, à éviter les négatives, et à gérer le compromis qu'offrent des interférences combinées afin de décider au mieux de son engagement à participer à des actions collectives. Un agent doit avant tout vérifier l'ajustement de l'objectif caché derrière la sollicitation à participer à une action collective, avec ses propres objectifs (individuels et collectifs). Les objectifs collectifs et les coéquipiers n'étant pas explicitement identifiés lors d'une sollicitation, il est intéressant de définir et de maintenir des descriptions de ces entités. Pour ce faire, nous proposons donc de nous inspirer de la théorie des identités.

3 L'identité au centre de l'engagement

Un individu ne peut s'absoudre de son environnement social, car il est influencé par sa famille, ses amis, son éducation, son travail, sa classe sociale, son sexe, la politique (notamment de son pays), son histoire etc. À chaque instant de sa vie, il est à la fois dépendant et contributeur de son environnement social. D'un point de vue social, il doit définir prudemment le cadre des in-

teractions avec les membres de son réseau social. D'un point de vue individuel, son comportement et ses émotions ne sont pas simplement déterminés par la société : il a toujours l'autonomie de faire ses choix. Un individu doit à la fois, (i) interagir *dans* un milieu social et (ii) garder le contrôle de son état interne *à l'écart de* ce même milieu. Ces deux aspects peuvent apparaitre incompatibles mais ils vont naturellement ensemble dans notre vie sociale, ce qu'explique la théorie de l'identité [7].

3.1 L'identité personnelle

Plusieurs études en sociologie définissent l'*identité personnelle* comme la façon dont une personne se considère, comment elle considère les autres, et comment elle se rapporte ou se comporte avec eux [8]. Il s'agit du centre de conscience, des besoins émotionnels, des désirs et du contrôle de soi, en fonction desquels l'individu réfléchit et agit sur ses interactions sociales [9]. Les travaux de psychologie [10] et en philosophie considèrent que l'identité personnelle regroupe les conditions nécessaires et suffisantes pour qu'un individu persiste dans le temps [11, 12]. Cependant, si un individu ne pouvait pas être reconnu d'une interaction à l'autre comme un même individu, aucun lien social stable ne pourrait être établi ou maintenu : il est donc nécessaire que l'identité personnelle soit complétée par une identité sociale.

3.2 L'identité sociale

La définition la plus répandue de l'identité sociale est celle de H. Tajfel [13] qui est le fondateur de la théorie de l'identité sociale. Il s'agit de "*la part de la conception de soi qui découle de sa connaissance, de son appartenance à un groupe social, ainsi que les valeurs et la signification émotionnelle attachée à cette appartenance*". Comme pour l'identité personnelle, la théorie de l'identité sociale est un sujet de grand intérêt dans plusieurs disciplines, dont la psychologie et la sociologie.

L'approche psychologique définit l'identité sociale comme la façon dont un individu s'autoperçoit en tant que membre d'un groupe social comme une nation, une culture, une religion, un sexe, ou une profession. Les individus appartenant au même groupe social partagent certaines caractéristiques communes qui peuvent les distinguer des autres groupes (*self-catégorisation* [14]). Selon cette approche, l'identité sociale d'un individu dérive des groupes et des catégories sociales auxquels il appartient.

L'approche sociologique s'intéresse aux relations interpersonnelles et aux interactions entre des membres d'un groupe social. Selon cette approche, une identité sociale se construit autour de trois aspects fondamentaux : (i) l'appartenance d'un individu à un groupe social (i.e. son rôle dans ce groupe, ses tâches, et l'avantage derrière cette appartenance), (ii) l'appartenance des autres membres du groupe social, et (iii) la conscience de l'existence d'un groupe social par les individus et de l'objectif global partagé du groupe. Ces trois aspects sont nécessaires pour la construction d'une identité sociale, car ils proviennent de l'individu et deviennent communs entre les individus partageant la même identité sociale.

Contrairement à l'identité sociale, qui est essentiellement un concept individuel fondé sur les caractéristiques d'un groupe ou d'une catégorie à laquelle l'individu appartient, l'identité collective [15] est une notion qui se réfère aux processus par lesquels un groupe s'identifie.

3.3 L'identité collective

L'identité personnelle regroupe les attributs et les significations qu'un individu attribue à lui-même. Les identités sociales peuvent s'intégrer dans l'identité personnelle, et naitre des appartenances sociales ou des rôles que joue un individu. Les identités collectives sont distinctes des identités personnelles et sociales de plusieurs façons. Premièrement, les identités collectives ne sont pas nécessairement intégrées dans les identités sociales existantes parce qu'elles sont souvent émergentes plutôt qu'ancrées dans les groupes sociaux [16]. Deuxièmement, les identités collectives tendent à être plus éphémères que les identités sociales ou personnelles (pour plus de détails, voir [17]). Troisièmement, contrairement à l'identité personnelle et sociale, l'observation d'une identité collective se fait en considérant les relations entre groupes.

3.4 Discussion

Les théories de l'identité personnelle et sociale fournissent une perspective théorique à partir de laquelle on peut examiner les relations entre des individus participant à plusieurs groupes sociaux [18]. Dans ce travail, nous considérons que l'identité personnelle est à la fois de nature sociale et psychologique, et ne peut pas être

exclusivement sociale ou psychologique (approche à l'intersection de la psychologie et la sociologie). Le soi conditionne les interactions sociales de l'individu, et en même temps, le soi ne peut exister que dans un contexte social. Selon cette approche de psychologie sociale, l'identité sociale intègre à la fois des aspects psychologiques, lorsque le contexte social est un groupe de grande échelle, et des aspects sociaux, pour les relations entre groupes dans lesquels les membres interagissent entre eux. L'identité sociale devient ainsi un concept multi-niveau qui comprend la définition de ses groupes sociaux à différentes échelles.

Nous proposons ici une relecture de ces travaux axés sur l'identité dans un contexte multi-agent. Nous nous sommes intéressés aux liens entre les architectures et les plateformes multi-agents et les théories qui les soutiennent. Dans un premier temps, nous avons complété la classification proposée par L. Braubach [19] avec des travaux plus récents issus de différentes revues [20, 21], afin de mettre en évidence les contributions multi-agents et leurs fondements théoriques (Fig. 1). Ainsi, nous avons dans un second temps proposé une classification des éléments issus des théories de l'identité afférente utilisées dans ces mêmes architectures (Fig. 2).

On remarque ainsi que les caractéristiques essentielles de l'identité personnelle et de l'identité sociale sont intrinsèquement contenues dans plusieurs architectures, ces dernières partageant les mêmes fondements théoriques que ceux de la théorie de l'identité. Néanmoins, peu d'attention est accordée à la construction de ces identités qu'elles soient personnelles ou sociales.

À partir de cette étude illustrée, nous avons notamment constaté dans les architectures d'agent le manque d'un processus homologue au concept philosophique de *comparateur d'identité* [8] dans les architectures d'agents examinées. Dans une situation d'interaction, le processus de comparaison d'identité d'un individu évalue les significations pertinentes tirées de son environnement, et celles tirées des retours d'interaction avec autruis, et les compare avec sa propre identité de référence pour la préserver ou contrôler. Cela nous a amenés à introduire un *processus de construction d'identité* pour permettre la réalisation de cette fonction. Nous avons identifié deux processus qui permettront à un agent d'appréhender ses interactions sociales dans un environnement multi-SMA. Le premier est un processus d'engagement dans des actions collectives, qui s'appuie sur le deuxième processus de construction des identités personnelles et sociales.

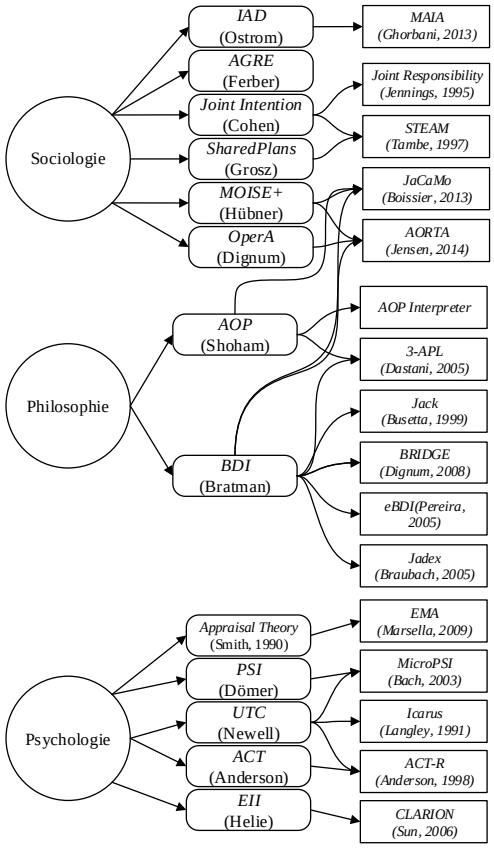


FIGURE 1 – Classification des disciplines et théories des architectures d'agents.

4 Illustration de la problématique

Nous présentons ici notre cas d'étude et nous esquissons son implantation multi-agent en utilisant le modèle BDI. Cette application nous permettra de positionner notre contribution.

4.1 Etude de cas

Considérons un scénario d'agriculture de précision que nous avons décrit plus précisément dans [22]. Des réseaux de capteurs impliquant des anémomètres, des thermomètres, et des pluviomètres sont déployés pour surveiller différents paramètres environnementaux liés à des applications agricoles. Des tags RFID sont utilisés pour identifier séparément chaque rangée de vigne. Des robots "vignerons" sont utilisés pour réaliser plusieurs tâches agricoles (fertilisation, taille, pulvérisation, désherbage, trai-

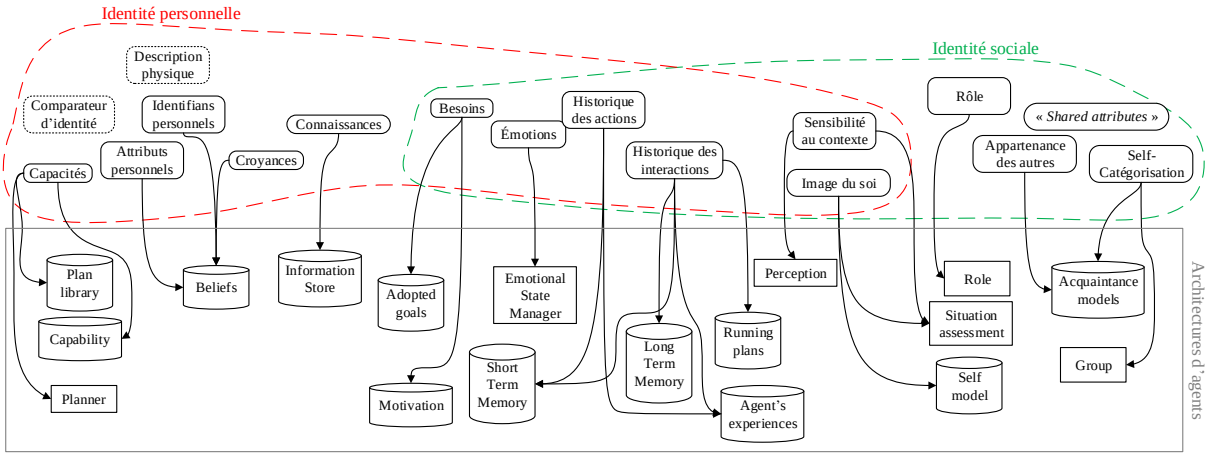


FIGURE 2 – Synthèse des éléments utilisés dans les architectures d'agents et leurs analogues dans la théorie de l'identité.

tement d'images, ...). Des drones survolent les vignes pour observer leur état et réaliser des tâches aériennes.

Par souci de simplification, nous nous focalisons sur un drone et un robot vigneron appartenant à ces groupes. Le drone a deux missions : une mission de pulvérisation, et une mission de supervision aérienne. Dans ce cadre, il désire gérer au mieux l'arrosage des vignes (tâche `ManageFieldWatering`). Pour cela (Fig. 3), il a la capacité de prendre des photos aériennes (`TakePicture`), de les transmettre à un serveur distant (`TransferPicture`) en vue d'exécuter des algorithmes de traitement d'image (`ProcessPicture`) afin de détecter si la partie concernée de la zone a besoin d'être pulvérisée (`DetectWateringNeeds`).

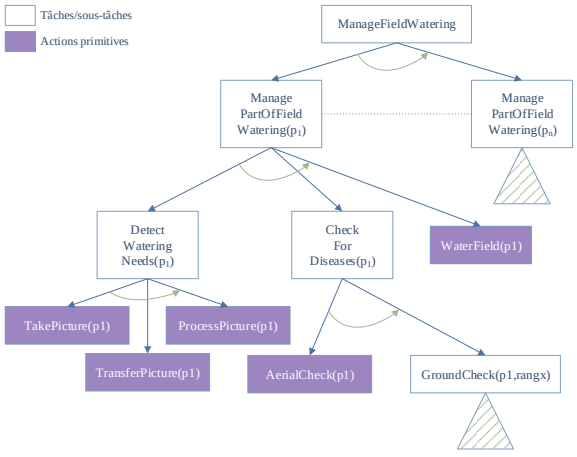


FIGURE 3 – Décomposition de la tâche *ManageFieldWatering* à accomplir par un drone.

Avant de passer à l'étape de pulvérisation d'eau (`WaterField`), le drone doit vérifier que la partie de la zone n'est pas atteinte par certaines maladies. Cette vérification commence par une analyse aérienne `AerialCheck` et, le cas échéant, une analyse au sol `GroundCheck`. Le drone identifie des rangées atteintes de la *flavescence dorée*. Cependant, il n'a pas les moyens pour accomplir la tâche `GroundCheck`, il cherche donc un robot qui pourra se déplacer pour compléter l'analyse par des prélèvements au sol. Cette sollicitation consiste à demander à un robot d'accomplir la tâche `GroundCheck(p1, rang21)`.

4.2 Description d'un solution en utilisant approche BDI

Il existe de nombreux modèles et architectures d'agents. Pour instancier ce scenario, nous avons utilisé une architecture BDI en raison de sa popularité dans notre communauté, et pour sa compatibilité avec notre cadre théorique (contexte délibératif etc.).

Un agent BDI est souvent modélisé par un tuple $\mathcal{X}_{bdi}$=<$\mathcal{E}, \mathcal{B}, \mathcal{P}_{lib}, \mathcal{I}, \mathcal{A}, \mathcal{S}_{\mathcal{E}}, \mathcal{S}_{\mathcal{O}}, \mathcal{S}_{\mathcal{I}}$>, avec $\mathcal{E}$ un ensemble d'évènements, $\mathcal{B}$ un ensemble de croyances, $\mathcal{P}_{lib}$ une librairie de plans, $\mathcal{I}$ un ensemble d'intentions, $\mathcal{A}$ un ensemble d'actions. La fonction de sélection $\mathcal{S}_{\mathcal{E}}$ extrait un évènement de l'ensemble $\mathcal{E}$ afin de chercher des plans dans $\mathcal{P}_{lib}$ qui peuvent être déclenchés pour réagir à l'évènement sélectionné. La fonction $\mathcal{S}_{\mathcal{O}}$ sélectionne un de ces plans afin de l'associer à une nouvelle intention ajoutée à l'ensemble $\mathcal{I}$,

$\mathcal{S}_{\mathcal{I}}$ sélectionne une intention de $\mathcal{I}$ qui déclenche le plan qui lui est associé.

Le cycle de décision simplifié d'un agent BDI est donné en Fig. 4 (On trouvera plus de détails sur un cycle de raisonnement complet dans [23]). L'agent génère un évènement lorsqu'il reçoit une requête d'un utilisateur (e.g. un utilisateur qui demande au drone d'accomplir la tâche `ManageFieldWatering`) ou d'un autre agent (e.g. le robot vigneron qui reçoit une sollicitation du drone pour accomplir la tâche `GroundCheck`), ou lorsqu'il détecte un changement dans l'état des capteurs de l'environnement. Ces évènements peuvent être externes dans le cas d'une sollicitation d'exécution d'une tâche, ou internes lorsque l'agent cherche à satisfaire ses propres objectifs. $\mathcal{E}_{int} \subseteq \mathcal{E}$ et $\mathcal{E}_{ext} \subseteq \mathcal{E}$ représentent respectivement l'ensemble des évènements internes et externes.

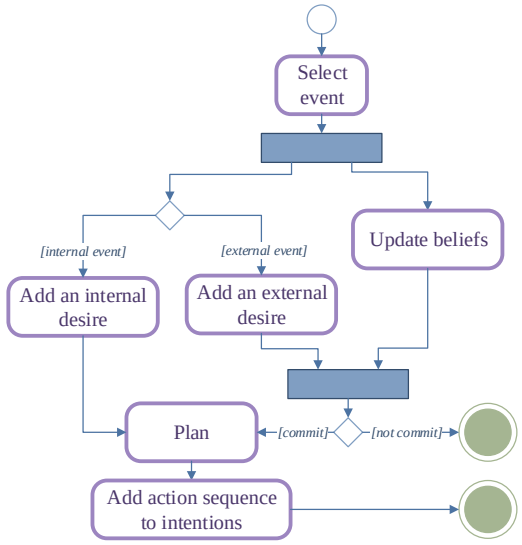


FIGURE 4 – Cycle de raisonnement simplifié associé à un modèle BDI.

Une fois l'évènement ajouté à l'ensemble $\mathcal{E}$, l'agent sélectionne un élément de cet ensemble, généralement, en fonction de sa priorité selon le domaine d'application (***Select Event***), et met à jour ses croyances $\mathcal{B}$ (***Update beliefs***). Ensuite, il traite chaque évènement selon son type (interne ou externe). Cette distinction est nécessaire car un agent s'engage automatiquement à satisfaire les désirs provenant d'évènements générés par lui-même (e.g. évitement de collision avec d'autres agents), alors qu'il doit choisir plus prudemment les évènements externes auxquels il réagit (e.g. répondre à une sollicitation de taille des pieds de vigne provenant d'un autre agent). Les évènements internes déclenchent des désirs internes $\mathcal{D}_{int} \subseteq \mathcal{D}$ (***Add an internal desire***), alors que les évènements externes déclenchent des désirs externes $\mathcal{D}_{ext} \subseteq \mathcal{D}$ (***Add an external desire***). Chaque désir externe est représenté par un couple <ϵ, α>, ϵ étant l'évènement, et α étant l'agent qui a déclenché l'ajout du désir à $\mathcal{D}_{ext}$.

Chaque agent dispose d'une librairie de plan $\mathcal{P}_{lib}$ qui contient des "recettes" pour qu'il puisse satisfaire ses objectifs (voir notre illustration en Fig. 3). Un plan est généralement structuré en un entête et un corps. L'entête spécifie les préconditions sous lesquelles le plan doit être déclenché. Le corps contient les sous-objectifs et/ou la séquence d'actions telle que, si l'agent l'exécute à partir d'un certain état initial, alors une fois la séquence d'actions finie, l'objectif sera atteint. Le plan est représenté en Fig. 3 par les feuilles de l'arbre (exécutées dans le même ordre que décrit par les flèches).

Après le processus de planification (***Plan***), l'agent ajoute le corps du plan approprié à sa liste d'intentions $\mathcal{I}$ (***Add action sequence to intentions***) pour réagir à l'évènement sélectionné. L'ensemble $\mathcal{I}$ contient une liste de plans partiels que l'agent doit suivre pour réaliser ses différents objectifs. L'agent doit donc choisir quelle intention sélectionner en fonction de ses priorités et du domaine de l'application.

4.3 Adaptation orientée-engagement de l'approche BDI

Nous avons considéré jusqu'ici la mise en oeuvre de la coopération par la délégation de tâches entre agents, l'agent sollicité s'engageant à satisfaire une requête d'un agent solliciteur. Cette notion d'engagement est un aspect important dans un modèle BDI, car elle fournit, au niveau global, un équilibre entre la réactivité et la poursuite des objectifs d'un SMA, et donne, au niveau local, une stabilité au processus de raisonnement d'un agent. Cependant, l'engagement d'un agent BDI à réaliser une sollicitation soulève plusieurs questions, liées par exemple, à l'ordre dans lequel un agent doit coordonner ses différents engagements, aux croyances qu'un agent doit générer lorsqu'il s'engage, à ce que désire un agent lorsqu'il annule un engagement etc.

Dans le cycle de décision d'un agent BDI, l'engagement intervient lors de l'étape de planification (***Plan***) ou après, lors de l'étape ***Add action sequence to intentions***.

Au niveau de l'agent, l'engagement individuel (ou interne) se réfère à la relation entre l'agent et l'action. Lorsqu'il décide d'atteindre un objectif, l'agent se prête à exécuter des tâches et à conserver ses intentions tant qu'il croit que l'objectif est possible à atteindre et qu'il n'est pas encore atteint. On identifie trois stratégies d'engagement différentes [24] :

blind l'agent s'engage à atteindre un objectif et n'abandonne son intention que lorsqu'il croit qu'elle est atteinte. C'est la stratégie d'engagement la plus forte, la plus durable.

single-minded l'agent n'abandonne son intention que s'il croit que son objectif est atteint ou qu'il est impossible à atteindre.

open-minded l'agent n'abandonne pas ses intentions tant qu'elles sont compatibles avec ses propres buts.

Au niveau social, un engagement social [25] n'est pas un engagement individuel partagé par plusieurs agents, c'est un concept relationnel qui lie plusieurs agents à une tâche, tel qu'un agent x s'engage à réaliser une tâche a pour le compte d'un agent y, en prenant en compte son engagement précédent avec un agent z [6].

Pourtant, la notion d'engagement accepte deux sens différents : elle peut signifier (i) une promesse ou une convention, comme (ii) une obligation, ou un contrat par lequel un agent exprime sa réponse à une sollicitation. La définition généralement adoptée dans les SMA [26] est la seconde.

Dans un contexte multi-SMA, plusieurs agents coexistent dans un même espace physique et coopèrent avec d'autres agents qui ne sont pas nécessairement du même système. Les agents doivent donc établir leurs interactions avec plus de prudence que dans le contexte du même SMA, d'où l'intérêt accordé par les travaux sur des agents BDI à la notion d'engagement. Dans cet article nous nous intéressons à deux cas où un agent pourrait accepter de s'engager à poursuivre des objectifs incompatibles [3] :

— la tâche déléguée par l'agent solliciteur est explicitement incompatible avec les objectifs de l'agent sollicité. Par exemple, si un agent a déjà adopté d'accomplir la tâche `WaterField(p1)`, il ne pourra pas s'engager à accomplir la tâche `Spray(sulphur, p1)`.

— la transformation d'un conflit externe inter-agents vers un conflit interne. Par exemple, un agent x accepte les sollicitations de deux agents y et z, tels que leurs tâches déléguées à x sont incompatibles. Ainsi, le conflit externe entre y et z deviendra un conflit interne à l'agent x.

5 IDEAS : Un modèle d'engagement centré identité

La définition d'engagement que nous utilisons dans cet article intervient au moment de la décision d'adopter un désir. Elle reste tout de même compatible avec la notion d'obligation *open-minded* et avec l'engagement social, car pour "honorer son contrat" ou "respecter son obligation" envers un autre agent, il faut considérer ses promesses à tenir auprès des futures sollicitations. En effet, chaque engagement contraint les relations sociales d'un agent, ses futures croyances et intentions, et par conséquent, son adoption de nouveaux désirs externes, d'où l'intérêt de proposer un processus d'engagement.

Notre conviction est que la théorie de l'identité est un bon cadre fondamental qui permet de maintenir des descriptions des collectifs auxquels un agent appartient, d'avoir des représentations des autres agents, et par conséquent d'intégrer des processus de construction de son identité personnelle et de ses identités sociales.

Pour ce faire, un agent doit avoir la capacité à prendre en compte les états mentaux des autres agents dans sa perception de l'environnement. Pourtant, il ne peut souvent pas connaître les états mentaux des autres agents, il ne peut que déduire leur existence sur la base de ce qui lui est observable, à savoir leurs actions, messages échangés, liens sociaux, et sollicitations. Nous nous intéressons en fait à un concept qui est à la base de la communication humaine, appelé l'*intersubjectivité*. Cette intersubjectivité - entre deux ou plusieurs agents en interaction - consiste à prendre en considération la reconnaissance de la signification intentionnelle des actions et sollicitations. Du point de vue multi-agent, l'intersubjectivité se réfère à l'interconnexion du *soi*, des *autres* et de l'*environnement*. Ce mécanisme, que nous traduisons par un algorithme de reconnaissance d'intention/plan, nous permettra de faire des hypothèses sur les représentations des autres agents.

5.1 Un processus d'engagement basé sur les identités

Nous proposons d'ajouter un processus d'engagement avant d'adopter un désir ex-

terne. Pour ce faire, nous complétons le tuple décrit dans la section précédente. Ainsi, un agent IDEAS (IDEntity-based Agent System) sera modélisé par le tuple $<\mathcal{X}_{bdi}, \mathcal{D}, \mathcal{C}_{\mathcal{P}}, \mathcal{P}, \mathcal{R}, \mathcal{P}_{\mathcal{I}}, \mathcal{S}_{\mathcal{I}}, \mathcal{P}_{\mathcal{PI}}, \mathcal{P}_{\mathcal{SI}}>$ (présenté sur Fig. 6). $\mathcal{D}$ est un ensemble de désirs, $\mathcal{C}_{\mathcal{P}}$ est un processus d'engagement, $\mathcal{P}$ est un planificateur, $\mathcal{R}$ est un processus de reconnaissance de plan, $\mathcal{P}_{\mathcal{I}}$ représente l'identité personnelle de l'agent, $\mathcal{S}_{\mathcal{I}}$ représente l'ensemble des identités sociales de l'agent, $\mathcal{P}_{\mathcal{PI}}$ et $\mathcal{P}_{\mathcal{SI}}$ sont des processus de construction de ces identités. La dynamique de l'intégration du processus d'engagement est illustrée sur la Fig. 5.

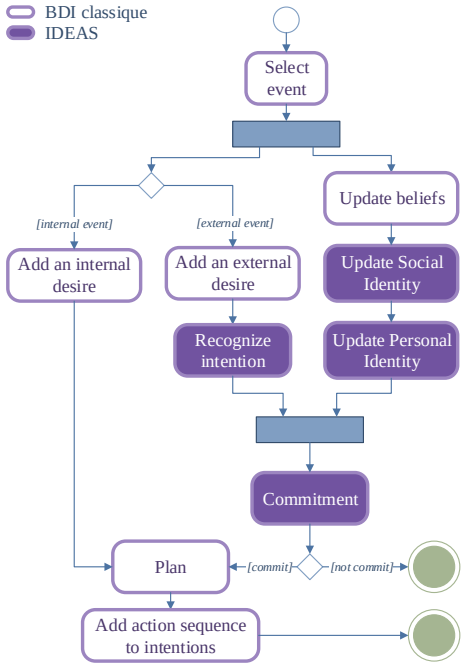


FIGURE 5 – Intégration de l'engagement et de l'identité dans un diagramme d'activité d'un agent BDI.

Recognize intention. Comme nous avons vu pour ***Plan***, l'agent exploite un processus de planification $\mathcal{P}$ pour réaliser ses objectifs (répertoriés dans $\mathcal{D}$). La reconnaissance d'intention est le processus inverse de la planification : la planification consiste à rechercher un plan pour atteindre un objectif donné, la reconnaissance de plan recherche le plan appliqué à partir d'actions observées et la reconnaissance d'intention le sens de cette volonté de satisfaire les objectifs. En plus de $\mathcal{P}_{lib}$, le processus de reconnaissance de plan $\mathcal{R}$ se sert aussi de la trace des actions et des interactions des agents observés, et donne en sortie les plans qui expliquent les actions observées d'un agent ou d'un ensemble d'agents.

Update social identity. Ce processus applique $\mathcal{R}$ pour identifier (i) les structures des équipes et (ii) les comportements des agents, en expliquant l'ensemble d'actions observées par la construction d'un plan qui les anime. L'agent pourra donc définir son identité sociale car il identifiera les équipes auxquelles il appartient, et les objectifs collectifs qu'il participe à réaliser.

Update personal identity. Ce processus se sert principalement des croyances $\mathcal{B}$ et d'un sous processus fortement inspiré du comparateur d'identité décrit dans le modèle de l'identité personnelle proposé par P. Burke [8]. Ce comparateur adapte l'identité personnelle en fonction des interactions avec les autres agents.

Commitment. Pour décider de s'engager ou non à adopter un désir externe, le processus $\mathcal{C}_{\mathcal{P}}$ prend en compte l'identité personnelle $\mathcal{P}_{\mathcal{I}}$, l'ensemble des identités sociales $\mathcal{S}_{\mathcal{I}}$, et aussi le résultat du processus de reconnaissance de plans $\mathcal{R}$.

5.2 Illustration du processus d'engagement

Dans le scénario décrit précédemment, lorsque le drone a sollicité le robot vigneron pour exécuter la tâche `GroundCheck(p1,rang21)`, ce dernier, ajoute cette tâche à ses désirs externes (***Add an external desire***). Il déclenche ensuite un processus d'engagement (***Commitment***) pour décider s'il adoptera ce désir ou pas. Le robot commencera le processus $\mathcal{C}_{\mathcal{P}}$ par la mise à jour de ses identités $\mathcal{S}_{\mathcal{I}}$ et $\mathcal{P}_{\mathcal{I}}$ en déclenchant les deux processus $\mathcal{P}_{\mathcal{SI}}$ (***Update social identity***) et $\mathcal{P}_{\mathcal{PI}}$ (***Update personal identity***).

Lorsqu'un agent est sollicité, c'est la dimension sociale de ses identités qui conditionne son processus d'engagement. Pour cette raison, même si nous adoptons une approche de psychologie sociale pour définir les identités d'un agent, nous ne couvrons pas la dimension psychologique de ces identités. Cette dimension psychologique est cependant importante lorsqu'un agent est solliciteur.

Pour mettre à jour ses identités sociales, le processus $\mathcal{P}_{\mathcal{SI}}$ du robot se sert de $\mathcal{R}$ pour identifier les plans (en cours d'exécution) dans lesquels il participe. Le robot `robot1` fait les observations suivantes :

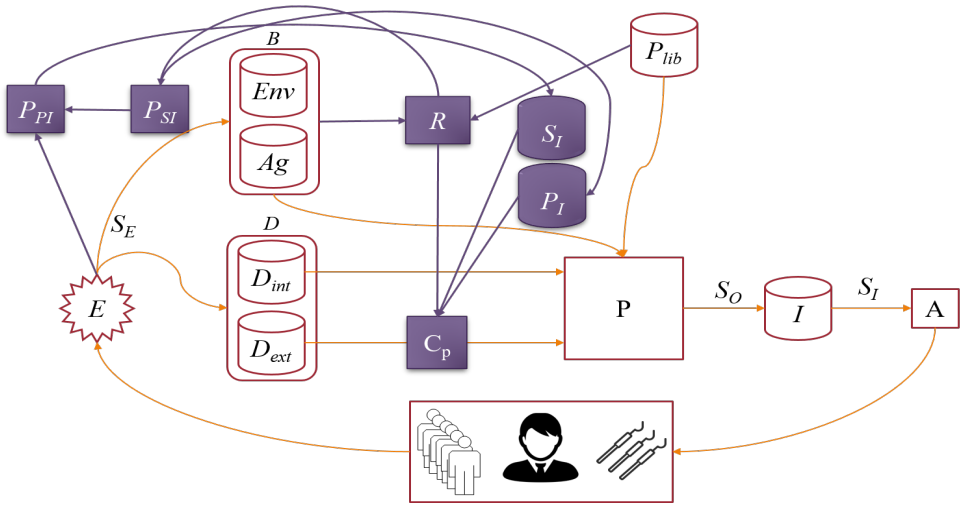


FIGURE 6 – Le modèle abstrait d'IDEAS.

t	robot1	robot2	drone2	robot4
1	Weed(p1,rang12)	GroundCheck(p5,rang38)	TakePicture(p1)	null
2	Weed(p1,rang13)	Weed(p2,range1)	TransferPicture(p1)	null
3	Weed(p1,rang14)	null	TakePicture(p2)	prune(p1, rang12)

À partir de ces observations, le processus $\mathcal{R}$ est capable d'inférer (i) les équipes d'agents avec lesquelles `robot1` coopère, (ii) les objectifs collectifs qu'il est en train de satisfaire, et (iii) sa participation dans chaque équipe qui suit les plans en cours d'exécution.

Pour mettre à jour son identité personnelle $\mathcal{P}_{\mathcal{I}}$, nous nous limitons dans cette illustration à mettre à jour la base des croyances du `robot1` et à y intégrer $\mathcal{S}_{\mathcal{I}}$ (de par sa définition l'identité personnelle inclut l'ensemble des identités sociales). Le processus $\mathcal{P}_{\mathcal{PI}}$ se sert des capteurs déployés dans son voisinage pour mettre à jour l'état de l'environnement et la représentation des autres agents (traces).

Dans cette étape, `robot1` a mis à jour ses identités personnelles et sociales. Avant de décider de son engagement, il doit vérifier leur compatibilité avec l'intention qui motive la sollicitation du drone (le dessein dans lequel s'inscrit la sollicitation). Pour ce faire, il se sert du processus $\mathcal{R}$ pour reconnaitre son intention (***Recognize plan***). Il observe que l'état de plusieurs champs est devenu `is-watered`, et que le drone a exécuté les actions `TakePicture`, `TransferPicture`, `ProcessPicture`, `AerialCheck`, `WaterField` sur plusieurs parties du champ.

À partir de ces observations, le robot infère que le drone a comme intention d'accomplir la tâche `ManageFieldWatering`.

Le robot a ainsi toutes les données nécessaires à une prise de décision quant à l'engagement qu'il doit prendre à réaliser la tâche `GroundCheck(p1,rang21)` sollicitée par le drone. Le processus d'engagement consiste à comparer l'objectif (inféré) du drone, avec les objectifs individuels et collectifs du robot, tout en prenant en compte d'autres critères non fonctionnels liés aux identités sociales du robot, comme la priorité des plans en cours d'exécution, par exemple.

6 Conclusion

Les agents embarqués déployés dans des environnements physiques sont de plus en plus interopérables et amenés à coexister avec des agents d'autres systèmes dans un même espace physique. Un environnement n'est plus la propriété d'un unique SMA. La coopération ne se limite alors plus aux agents d'un même SMA : un agent doit donc être capable de répondre à des sollicitations exprimées de manière opportuniste par des agents extérieurs à son système. Pour doter l'agent de la capacité de raisonner sur son engagement à participer à une action collective et d'appréhender ses interactions sociales avec les agents de son voisinage, nous nous basons sur la théorie de l'identité pour proposer le modèle d'agent IDEAS. Un agent utilisant ce modèle prend en compte les notions d'identité personnelle et d'identité sociale dans sa décision de coopérer ou non avec un agent solliciteur. Ce processus de jugement utilise aussi un modèle de reconnaissance de plan pour avoir une représentation de ses coéquipiers et de ses partenaires

potentiels.

Plusieurs perspectives de ce travail sont envisagées. Sur le plan technique, nous adaptons le modèle proposé à une architecture OSGi [27]. Cette intégration nous fournira les ontologies (fonctionnalités, modèles d'actions, ...) et les modules nécessaires (raisonneur sémantique, planificateur, gestionnaire d'interopérabilité matérielle, gestionnaire de contexte, ...) pour implémenter notre architecture d'agent. Sur le plan théorique, un travail similaire à ce que nous avons effectué pour un agent sollicité est à effectuer pour les agents solliciteurs. Il nous parait intéressant d'étudier les travaux de la *saillance d'identité* [28] pour permettre à un agent de choisir quelle identité « activer » lors de l'initiation d'une nouvelle coopération.

Remerciements

Ce travail est financé par l'ANR ASAWoO <ANR-13-INFR-012>.

Références

[1] D. Guinard, V. Trifa, F. Mattern, and E. Wilde, "From the internet of things to the web of things : Resource-oriented architecture and best practices," in *Architecting the Internet of Things*, pp. 97–129, Springer, 2011.

[2] J. Ferber and J.-F. Perrot, *Les systèmes multi-agents : vers une intelligence collective*. InterEditions, 1995.

[3] C. Castelfranchi and R. Falcone, "Conflicts within and for collaboration," in *Conflicting agents*, pp. 33–61, Springer, 2002.

[4] L. E. Parker, "Distributed intelligence : Overview of the field and its application in multi-robot systems," *Journal of Physical Agents*, vol. 2, no. 1, pp. 5–14, 2008.

[5] P. R. Cohen and H. J. Levesque, "Teamwork," *Nous*, vol. 25, no. 4, pp. 487–512, 1991.

[6] C. Castelfranchi, "Modelling social action for ai agents," *Artificial Intelligence*, vol. 103, no. 1, pp. 157–182, 1998.

[7] P. J. Burke and J. E. Stets, *Identity theory*. Oxford University Press, 2009.

[8] P. J. Burke, *Contemporary social psychological theories*. Stanford University Press, 2006.

[9] D. Layder, *Social and personal identity : Understanding yourself*. Sage, 2004.

[10] R. Martin and J. Barnes, "Personal identity," 2008.

[11] B. Garrett, *Personal identity and self-consciousness*. Routledge, 2002.

[12] G. Strawson, *Locke on personal identity : Consciousness and concernment*. Princeton University Press, 2014.

[13] H. E. Tajfel, *Differentiation between social groups : Studies in the social psychology of intergroup relations.* Academic Press, 1978.

[14] J. C. Turner, "Social categorization and the self-concept : A social cognitive theory of group behavior," *Advances in group processes*, vol. 2, pp. 77–122, 1985.

[15] A. Melucci, *Challenging codes : Collective action in the information age*. Cambridge University Press, 1996.

[16] J. D. DeLamater and A. Ward, *Handbook of social psychology*. Springer, 2006.

[17] D. Snow, "Collective identity and expressive forms," *Center for the Study of Democracy*, 2001.

[18] D. Abrams and M. A. Hogg, *Social identifications : A social psychology of intergroup relations and group processes*. Routledge, 2006.

[19] L. Braubach, A. Pokahr, and W. Lamersdorf, "A universal criteria catalog for evaluation of heterogeneous agent development artifacts," *From Agent Theory to Agent Implementation (AT2AI-6)*, pp. 19–28, 2008.

[20] K. Kravari and N. Bassiliades, "A survey of agent platforms," *Journal of Artificial Societies and Social Simulation*, vol. 18, no. 1, p. 11, 2015.

[21] T. Balke and N. Gilbert, "How do agents make decisions ? a survey," *Journal of Artificial Societies and Social Simulation*, vol. 17, no. 4, p. 13, 2014.

[22] L. Médini, M. Mrissa, E.-M. Khalfi, M. Terdjimi, N. Le Sommer, P. Capdepuy, J.-P. Jamont, M. Occello, and L. Touseau, "Building a web of things with avatars," in *Managing the Web of Things : Linking the Real World to the Web* (L. Y. a. B. B. Michael Sheng, Yongrui Qin, ed.), ch. 6, pp. 1–30, Morgan Kaufmann, Elsevien, 2017.

[23] R. H. Bordini, J. F. Hübner, and M. Wooldridge, *Programming multi-agent systems in AgentSpeak using Jason*, vol. 8. John Wiley & Sons, 2007.

[24] A. Haddadi, *Communication and cooperation in agent systems : a pragmatic theory*, vol. 1056. Springer Science & Business Media, 1996.

[25] C. Castelfranchi, "Commitments : From individual intentions to groups and organizations.," in *ICMAS*, vol. 95, pp. 41–48, 1995.

[26] M. Baldoni, C. Baroglio, F. Capuzzimati, and R. Micalizio, "Commitment-based agent interaction in jacamo+," *Fundamenta Informaticae*, 2017.

[27] M. Mrissa, L. Médini, J.-P. Jamont, N. Le Sommer, and J. Laplace, "An avatar architecture for the web of things," *IEEE Internet Computing*, vol. 19, no. 2, pp. 30–38, 2015.

[28] S. Stryker and R. T. Serpe, "Commitment, identity salience, and role behavior : Theory and research example," in *Personality, roles, and social behavior*, pp. 199–218, Springer, 1982.

Une approche multi-agent basée sur la confiance pour évaluer la performance des plateformes de crowdsourcing d'idées

Amine Louati
amine.louati@telecom-em.eu

Christine Balagué
christine.balague@telecom-em.eu

Mehdi Elmoukhliss
mehdi-alexandre.elmoukhliss@telecom-em.eu

Télécom École de Management, LITEM, Institut mines Télécom, France

Résumé

Avec l'expansion du phénomène de crowdsourcing d'idées, les entreprises s'appuient de plus en plus sur la simultanéité de la coopération et de la compétition, appelée «coopétition» pour générer de nouvelles idées de produits et améliorer leur processus d'innovation. Des études antérieures ont montré que cette simultanéité offre des avantages pour la performance des plateformes de crowdsourcing d'idées. Cependant, ces études présentent trois limites majeures au niveau de (i) la modélisation mono-relationnelle du réseau de coopétition, (ii) l'absence de la dimension sociale dans les interactions et (iii) le manque d'analyse de la dynamique sociale des utilisateurs dans le réseau de coopétition. Pour surmonter ces limites, nous modélisons le réseau de coopétition comme un réseau multi-relationnel où les relations sont de types différents. Nous proposons un modèle de confiance qui permet aux utilisateurs de qualifier les types de relations et qui guide leur comportement lorsqu'ils décident d'interagir. Nous proposons également l'utilisation d'un système multi-agent car les agents supportent les interactions et offrent des capacités développées de raisonnement, d'extraction et de représentation de connaissance utiles pour évaluer la confiance. Enfin, à l'aide d'une simulation multi-agent, nous évaluons la performance d'une plateforme homogène et donnons des pistes sur la façon de l'améliorer pour générer plus d'idées.

Mots-clés : *Réseau de coopétition, crowdsourcing d'idées, confiance, simulation multi-agent, coopération, compétition.*

Abstract

With the expansion of the idea crowdsourcing phenomenon, companies are increasingly relying on the simultaneity of cooperation and competition namely "coopetition" to generate new ideas of products and to improve their innovation process. Some studies have shown that this simultaneity creates benefits in terms of performance of idea crowdsourcing platforms. However, these studies present three main limits in (i) the mono-relational modeling of the coopetitive network, (ii) the absence of the social dimension in interactions and (iii) the lack of analysis of the social dynamics of users in the coopetition network. To overcome these limits, we model the coopetition network as a multi-relational network where relationships are of different types. We propose a model of trust that allows users to qualify the type of relationships and guides their behavior when they decide to interact. We also propose the use of a multi-agent system as agents endorse interactions and offer developed capacities of reasoning, extraction and representation of knowledge useful to evaluate trust. Finally, using a multi-agent simulation, we evaluate the performance of a homogeneous platform and give insights on how to enhance it to generate more ideas

Keywords: *Coopetition Networks, Idea Crowdsourcing, Trust, Multi-Agent Simulation, Cooperation, Competition.*

1 Introduction

Au cours de la dernière décennie, de nombreuses sociétés ont expérimenté des plateformes de crowdsourcing en ligne pour générer de nouvelles idées de produits ou de services et pour améliorer leur processus d'innovation. Par exemple, la société pharmaceutique Merck [1] a travaillé avec Kaggle [2], un site collaboratif d'analyse prédictive, pour uniformiser son processus de découverte de médicaments. Aujourd'hui, les plateformes de crowdsourcing sont utilisées non seulement par des entreprises privées mais aussi par des gouvernements ou des organisations publiques sur des causes humanitaires ou environnementales mondiales. Beau-

1. http ://www.merck.com/index.html
2. https ://www.kaggle.com/

coup d'études ont mis l'accent sur l'identification des facteurs influençant la performance de ces plateformes en termes de créativité ou de quantité d'idées générées, comme la compétition et l'engagement des utilisateurs [2], la rétroaction en temps réel [14], la motivation [17], le nombre d'utilisateurs [4], les connaissances antérieures [8], le climat de coopétition [21] et la comparaison entre les incitations externes et internes [20]. D'autres études ont examiné l'influence des rôles des utilisateurs sur la qualité de la contribution et comment "*la coopétition*" [6, 18] peut améliorer les résultats d'innovation des entreprises. Toutes ces études ont été menées par des analyses sur *des données statiques* utilisant des méthodes quantitatives telles que l'analyse des réseaux sociaux et/ou des méthodes qualitatives telles que l'analyse interprétative du contenu. Bien que les résultats d'analyses aient répondu à des questions importantes et fourni des éléments d'amélioration pour la conception des plateformes de crowdsourcing d'idées, trois limites doivent être surmontées.

— Tout d'abord, peu de recherches ont considéré *l'aspect multi-relationnel* dans la modélisation du réseau de coopétition. Hu et al. [11] ont modélisé ce réseau par un graphe dirigé et signé dans lequel les arcs positifs et négatifs représentent respectivement les relations coopératives et compétitives. Bien que l'aspect multi-relationnel soit pris en compte dans la modélisation, il n'inclut pas la relation coopétitive. Pour combler cette lacune, nous modélisons le réseau de coopétition comme un réseau multi-relationnel distinguant une relation coopérative, compétitive ou coopétitive.

— Deuxièmement, à notre connaissance, aucune étude n'a considéré *la dimension sociale* incluant la perception de la plateforme et l'historique interactif dans la définition des interactions. Pour surmonter cette limite, nous proposons l'utilisation d'un modèle de confiance comme un mécanisme qui articule les comportements coopératif, compétitif et coopétitif entre les utilisateurs et qui gère leur processus décisionnel lorsqu'ils interagissent. Dans ce papier, notre modèle de confiance est bâti sur deux dimensions, *la dimension environnementale* et *la dimension dyadique*. La dimension environnementale de la confiance se compose de deux mesures : la confiance climatique et la confiance sociale. Elle permet aux utilisateurs de qualifier leur perception de la plateforme qui peut être coopérative, compétitive ou coopétitive. La dimension dyadique de la confiance permet aux utilisateurs d'évaluer la confiance entre eux. Elle est calculée en se basant sur l'historique interactif des agents tout en tenant compte de l'aspect temporel dans le processus de construction.

— Troisièmement, peu d'études intègrent *la dynamique sociale* des utilisateurs ainsi que l'évolution de leur relations dans la plateforme [9, 12, 16]. Pour résoudre ce problème, nous proposons l'utilisation d'un *système multi-agent* car les agents ont démontré leur capacité à supporter les interactions tout en utilisant le raisonnement, l'extraction et la représentation de connaissances ainsi que des métaphores sociales comme *la confiance*. Ainsi, chaque utilisateur dans la plateforme est associé à un agent autonome caractérisé par un rôle capable d'évaluer la confiance des autres agents avant d'interagir avec eux. Dans le présent travail, nous considérons trois rôles différents : *le coopérateur* celui qui préfère coopérer ou ne rien faire, *le compétiteur* celui qui préfère concurrencer ou ne rien faire et *le coopétiteur* celui qui peut coopérer et concurrencer en même temps.

Le but de ce travail est d'explorer à l'aide d'une simulation multi-agent basée sur la confiance la dynamique micro-sociale des agents pour mieux comprendre leur dynamique macro-sociale et améliorer par la suite la performance d'une plateforme homogène. Une plateforme homogène est une plateforme dans laquelle les trois rôles sont distribués d'une manière égale entre les agents. La suite de ce papier est structurée de la manière suivante. La section suivante présente le cadre théorique de ce travail. La section 3 présente une description détaillée de notre modèle de confiance et montre comment il articule la dynamique sociale des agents dans la plateforme. La section 4 montre quelques résultats expérimentaux. La section 5 conclut le papier et identifie les travaux futurs.

2 Cadre théorique

Dans cette section, nous définissons les principaux concepts utilisés pour construire et valider notre recherche c'est à dire, l'architecture des agents et le réseau de coopétition.

2.1 Architecture d'agent

À chaque utilisateur de la plateforme, nous attribuons un agent autonome qui agit en son nom

en utilisant trois types d'actions :

— Les actions positives (ac^+) : sont des actions de soutien telles que la publication d'un commentaire positif, aimer un commentaire ou aimer une idée ;

— Les actions négatives (ac^-) : sont des actions conflictuelles telles que la publication d'un commentaire négatif ou de ne pas aimer une idée ;

— Les actions neutres ($ac^\perp$) : comme la soumission d'une nouvelle idée.

Pour simplifier la conception de l'architecture, nous dotons tous les agents du même ensemble d'actions, mais avec des probabilités d'exécution différentes en fonction du rôle caractérisant chacun. Dans ce travail, nous considérons trois rôles :

— *le coopérateur :* ce type d'agent suit un comportement sociable et amical caractérisé par un niveau élevé d'actions positives envers les idées des autres, car il préfère coopérer et non pas concurrencer.

— *le compétiteur :* ce type d'agent est motivé par le fait de remporter les prix de la plateforme. Par conséquent, il exprime un comportement agressif et hostile caractérisé par un niveau élevé d'actions négatives car il préfère rivaliser plutôt que coopérer.

— *le coopétiteur :* ce type d'agent montre simultanément des comportements amicaux et hostiles caractérisés par un grand nombre d'actions positives et négatives car il préfère concurrencer et coopérer en même temps.

TABLE 1 – Un exemple de règles d'action indépendantes de la confiance entre un agent a_k et un agent a_j

Rôle du a_k \ Type d'action	probabilité de ac^+	probabilité de ac^-
coopérateur	p^+=1	p^- = 0
compétiteur	p^+=0	p^- = 1
coopétiteur	p^+=0.5	p^- = 0.5

Au début de la simulation, les agents n'ont pas d'historique interactif et donc ne peuvent pas évaluer la confiance entre eux. Ce problème de démarrage à froid ne devrait pas les empêcher d'interagir. Par conséquent, il est nécessaire d'avoir des règles d'action permettant aux agents d'interagir indépendamment de la confiance. La table 1 représente un exemple de règles d'action indépendantes de la confiance qui gèrent les interactions entre un agent a_k et un autre agent a_j. Ces règles sont exprimées, pour les différents rôles, en termes d'une probabilité d'exécution d'une action positive ou négative. Comme nous pouvons le voir, nous partons de l'hypothèse que les coopérateurs (resp. compétiteurs) suivent un *comportement uni-modal* car ils ne peuvent effectuer que des actions positives (respectivement négatives). Cependant, les coopétiteurs suivent un *comportement bi-modal* car ils peuvent effectuer aussi bien des actions positives que négatives avec une probabilité égale. Maintenant, nous décrivons les différents composants de notre architecture d'agent qui s'appuie sur la confiance.

Définition 1 *(Architecture d'agent) Un agent a_k est une entité autonome définie par un tuple $a_k =< \mathcal{BR}, \mathcal{GR}, \mathcal{RM}, \mathcal{TM}, \mathcal{IM} >$ où :*

— *$\mathcal{BR} =< Role_k, PIT_k >$ est la base de croyances où $Role_k \in$ {coopérateur, compétiteur, coopétiteur}est le rôle de l'agent, et PIT_k est la Table Personnelle d'Interaction. Chaque enregistrement dans PIT_k contient les éléments suivants : un agent $a_j \in V$ et un ensemble d'actions Ac_k effectuées par a_j envers a_k tout au long de la simulation.*

— *$\mathcal{GR}$ est la base de buts qui contient un but de motivation de l'agent. Celui-ci représente le degré d'engagement d'un agent au sein de la plateforme exprimé en termes de nombre d'actions effectuées. Nous supposons qu'un but est atteint que si le nombre d'actions effectuées dépasse un seuil prédéfini.*

— *$\mathcal{RM}$ est le module de raisonnement qui contient les règles d'action dépendantes de la confiance et indépendantes de la confiance utilisées par l'agent pour interagir avec d'autres et atteindre son but.*

— *$\mathcal{TM}$ est le module de confiance qui permet à l'agent d'évaluer la fiabilité des autres agents et guide son processus décisionnel avant d'interagir.*

— *$\mathcal{IM}$ est le module d'interaction qui structure les actions envoyées par l'agent et traite celles reçues.*

2.2 Réseau de coopétition

La majorité des modèles existants représentant les utilisateurs des plateformes de crowdsourcing d'idées et leurs interactions reposent sur un modèle de réseau mono-relationnel et statique. Ce modèle échoue d'abord, à considèrer les différents types de relations liant les

utilisateurs (i.e. relation cooperative, compétitive....) et deuxièmement, à considérer les évolutions qui peuvent survenir au cours du temps. Pour combler ces lacunes, nous modélisons le réseau de coopétition comme un réseau *multi-relationnel* [19] et *dynamique* [3]. Un réseau multi-relationnel permet aux agents de développer différents types de relations à partir d'un ensemble R. Dans ce travail, nous distinguons trois types de relations tel que $R = \{R_{coo}, R_{com}, R_{cop}\}$, où R_{coo}, R_{com} et R_{cop} correspondent respectivement aux relations de coopération, de compétition et de coopétition. Un réseau dynamique permet de suivre en temps réel la dynamique sociale des agents. Dans la littérature, deux modèles de réseaux dynamiques sont proposés : les modèles discrets et les modèles continus. Dans un modèle discret, des réseaux statiques appelés "snapshots" sont régulièrement enregistrés à chaque période de temps fixe (par exemple, toutes les 30 minutes ou tous les jours). Ce modèle fournit une cartographie complète des états du graphe à des intervalles de temps réguliers. En revanche, le modèle continu conserve la trace de tous les changements à des instants irréguliers représentant chacun d'eux en un graphe valide. Dans ce travail, nous avons décidé d'utiliser le modèle discret car il nous permet de stocker un nombre prédéfini de snapshots et nous donne la possibilité d'interroger et de visualiser des snapshots en fonction du temps. Formellement, un snapshot est défini comme suit :

Définition 2 *(Snapshot) Un snapshot est défini par un triplet* $G_i =< V, E_i, t_i >$ *tel que :*

- $V = \{a_1, a_2, ..., a_n\}$ *est un ensemble de* n *nœuds ;*
- $E_i = \{E_{i,pos}, E_{i,neg}\}$ *est un ensemble d'arcs où* $E_{i,l} \subseteq V \times V \; \forall l \in \{pos, neg\}$ *est le sous-ensemble d'arcs du* l*-ème type d'action ;*
- t_i *est la période de temps.*

Selon cette définition, un snapshot est un graphe dirigé où les nœuds sont les agents et les arcs sont les interactions entre eux effectuées pendant la période t_i. Une interaction résulte de deux types d'actions : positive ou négative. Nous supposons que toutes les actions qui se produisent pendant la même période sont considérées comme instantanées et apparaissent dans le même snapshot.

Un réseau de coopétition est un réseau dynamique obtenu à partir de l'agrégation d'une séquence de snapshots $G_{[t_0,t_z]} = \{G_0, G_1, G_2, \ldots, G_z\}$. Il est défini comme suit :

Définition 3 *(Réseau de coopétition) Un réseau de coopétition est défini par un triplet* $G_{[t_0,t_z]} =< V, E, W >$ *tel que :*

- $V = \{a_1, a_2, ..., a_n\}$ *est un ensemble de* n *nœuds ;*
- $E = \{E_{coo}, E_{com}, E_{cop}\}$ *est un ensemble d'arcs* $E_l \subseteq V \times V \; \forall l \in \{coo, com, cop\}$ *est le sous-ensemble d'arcs par rapport à la relation* R_l *;*
- $W : E \mapsto [0, 1]$ *est une fonction de pondération qui associe chaque arc à une valeur de confiance* [3] *;*

Selon cette définition, un réseau de coopétition est un graphe dirigé et pondéré où les nœuds représentent les agents, les arcs représentent des relations de confiance entre les agents et les poids sur les arcs correspondent aux valeurs de confiance entre chaque paire d'agents. Une relation de confiance entre deux agents est une *relation de long terme* établie sur la base de leur historique interactif au cours des snapshots précédents. Elle peut être de trois types : coopérative, compétitive et coopétitive.

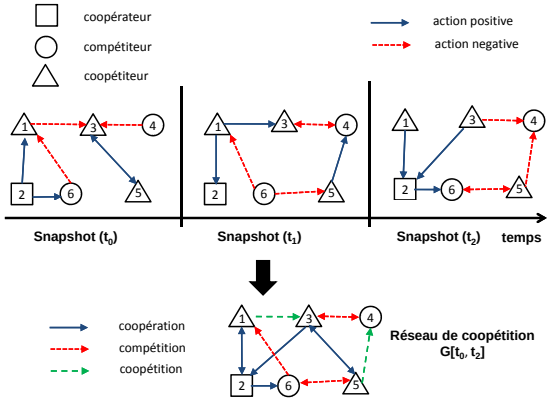


FIGURE 1 – Un exemple de construction d'un réseau de coopétition avec six agents sur trois périodes

Pour mettre en œuvre les concepts d'agent et du réseau de coopétition, nous représentons dans

3. Dans la section suivante, nous allons décrire comment ces valeurs de confiance entre les agents sont calculées.

la figure 1 un exemple de construction d'un réseau de coopétition avec six agents et nous montrons sa visualisation sur trois périodes t_0, t_1 et t_2. Comme nous pouvons le voir, chaque graphe sur l'axe du temps représente un snapshot enregistré à une période différente. Chaque snapshot contient un coopérateur, deux compétiteurs et trois coopétiteurs qui interagissent en utilisant deux types d'actions : positive et négative. La construction du réseau de coopétition à la période t_2 est basée sur la juxtaposition de tous les snapshots précédents allant de t_0 à t_2 où les arcs représentent les relations de confiance entre eux.

3 Description du modèle de confiance et de la dynamique sociale des agents

La confiance est un phénomène complexe qui est la base de toutes les interactions sociales [1]. Ainsi, tout modèle computationnel de confiance doit être conçu sur la base des caractéristiques réelles de la confiance entre les individus dans la société. En informatique, et en particulier dans les systèmes multi-agents, la confiance est définie comme une condition nécessaire qui provoque la coopération entre les agents [5]. Ainsi, c'est le principal mécanisme qui guide le processus décisionnel des agents quand ils veulent interagir. Concrètement, à chaque période de la simulation, chaque agent doit prendre les décisions suivantes basées sur l'évaluation de la confiance : (1) Quel est son potentiel d'action ?[4], (2) avec qui interagir ? et (3) comment interagir (i.e. action positive ou négative) ?. Pour répondre à ces questions, nous présentons formellement dans cette section notre modèle de confiance composé de deux dimensions ; *la dimension environnementale* et *la dimension dyadique* qui guident le processus décisionnel des agents avant d'interagir.

3.1 La dimension environnementale

Outre les caractéristiques internes de la personnalité, le comportement d'un individu est influencé par des facteurs sociaux externes issus de l'environnement dans lequel il évolue [15]. En psychologie cognitive, cette relation «individu-environnement » est appelée *affordance*. A l'origine, le concept d'affordance émergeait du travail de Gibson en psychologie écologique [10] pour expliquer l'adaptation immédiate d'un individu à son environnement. Dans un contexte de crowdsourcing d'idées, l'affordance représente le champ d'actions possibles d'un agent appelé «potentiel d'action ». Il résulte de la combinaison de sa perception de l'environnent et de son rôle. Par exemple, un coopérateur qui évolue dans un environnement coopératif sera plus motivé à interagir et donc plus actif. Cependant, dans un environnement concurrentiel, il sera démotivé et probablement réticent à participer et à interagir avec les autres. Pour permettre aux agents de qualifier leur perception de l'environnement et donc de déterminer leur potentiel d'action, nous introduisons la dimension environnementale de la confiance. Celle-ci se compose de deux mesures : la mesure de confiance climatique et la mesure de confiance sociale. Ensuite, nous décrivons comment les agréger en une seule valeur de confiance et indiquons comment calculer le potentiel d'action.

La confiance climatique L'un des deux facteurs sociaux qui contribue à la construction de la perception est *la tendance observée* régnant dans la plateforme [21]. C'est une vision collective qui décrit ce que les agents observent à un moment donné comme actions sur la page d'accueil de la plateforme. Cet ensemble d'actions noté $Pset_i$ observé à la période t_i représente une fraction de l'ensemble total d'actions E_{i-1} effectuées dans la plateforme à la période précédente t_{i-1}. Pour permettre aux agents de qualifier la page d'accueil à une période t_i, nous introduisons la mesure de confiance climatique notée $CTrust(Pset_i)$. Elle est définie pour tous les agents comme étant le rapport entre le nombre des actions positives et le nombre total des actions observées.

$$\forall a_k \in V \; CTrust(Pset_i) = \frac{cardinal(\{ac^+ \in Pset_i\})}{cardinal(Pset_i)} \quad (1)$$

La confiance sociale L'introduction de la confiance climatique nous a permis de mesurer la tendance observée sur la plateforme, qui est le premier facteur social requis pour qualifier la perception. Celle-ci devrait engendrer le même impact sur le comportement des agents ayant le même rôle. Cependant, si nous voulons comprendre la variation spécifique du comportement de chaque agent, nous devons examiner de plus près sa situation locale. Dans un réseau de coopétition, la situation locale

4. Le potentiel d'action d'un agent représente son niveau d'engagement en termes du nombre d'actions qu'il peut effectuer dans la plateforme.

d'un agent correspond à son réseau égocentrique. De nombreuse études ont souligné la pertinence de l'attitude du réseau égocentrique, comme l'un des principaux facteurs sociaux qui influent sur le comportement de l'individu quand il décide d'interagir. Pour permettre à un agent d'interpréter l'attitude de son réseau égocentrique, nous introduisons la mesure de confiance sociale. Puisque les anciennes interactions peuvent devenir rapidement obsolètes, nous supposons que la mémoire des agents est de taille limitée, notée S. Par conséquent, le réseau égocentrique d'un agent ne comprend que les agents ayant interagi avec lui pendant les S dernières périodes. Il est définie comme suit :

Définition 4 *(Réseau égocentrique) Soit $G_{[t_0,t_i]}$ un graphe de coopétition et soit $S \in \mathbb{N}$ la taille de la mémoire, alors le réseau égocentrique d'un agent $a_k \in V$, noté $H_{[t_{i-S},t_i]} =< V', E' >$, est un sous-graphe orienté de $G_{[t_{i-S},t_i]}$ tel que $V' = \{a_j \in V \mid (a_k, a_j) \in G_{[t_{i-S},t_i]}\} \cup \{a_k\}$ et $E' = \{\bigcup_{x=i-S}^{i} E_x \cap a_k\}$.*

Pour un agent a_k dont le réseau égocentrique est $H_{[t_{i-S},t_i]}$, la mesure de la confiance sociale, notée $STrust(a_k, H_{[t_{i-S},t_i]})$, est définie par :

$$STrust(a_k, H_{[t_{i-S},t_i]}) = \frac{cardinal(\{ac^+ \in E'\})}{cardinal(E')} \quad (2)$$

Contrairement à la mesure de la confiance climatique, la mesure de la confiance sociale est un point de vue individuel spécifique à chaque agent et dépend de son expérience personnelle dans la plateforme.

L'agrégation de la confiance Maintenant, nous procédons à une étape d'agrégation qui combine la confiance climatique et sociale dans un score global, à savoir la confiance environnementale. Il s'agit d'une somme pondérée donnée par la fonction suivante :

$$\begin{aligned} ETrust(a_k, Pset_i \cup H_{[t_{i-S},t_i]})) = w \times CTrust(Pset_i) \\ +(1-w) \times STrust(a_k, H_{[t_{i-S},t_i]})) \end{aligned} \quad (3)$$

où $w \in [0..1]$ est un paramètre de pondération spécifiant l'importance accordée à la valeur de la confiance climatique.

Comme mentionné ci-dessus, la valeur de la confiance environnementale reflète la perception de la plateforme. Selon la valeur obtenue, elle peut être compétitive, coopétitive ou coopérative. Nous divisons l'échelle de mesure en trois intervalles égaux sans exclusion ni préférence, en considérant que lorsque la confiance environnementale est faible, la perception est compétitive, lorsqu'elle est forte elle est coopérative et si elle n'est ni trop faible ni trop forte, la perception est coopétitive. Par conséquent, la perception est qualifiée comme suit :

- Si $ETrust(a_k, Pset_i \cup H_{[t_{i-S},t_i]})) \in [0, \frac{1}{3}[$, alors perception= compétitive ;
- Si $ETrust(a_k, Pset_i \cup H_{[t_{i-S},t_i]})) \in [\frac{1}{3}, \frac{2}{3}]$, alors perception= coopétitive ;
- Si $ETrust(a_k, Pset_i \cup H_{[t_{i-S},t_i]})) \in]\frac{2}{3}, 1]$, alors perception= coopérative.

La qualification de la perception permet à chaque agent a_k de calculer son potentiel d'action en fonction de son rôle. Nous indiquons par $\mathcal{N}_k$ le nombre maximum d'actions qu'un agent a_k peut effectuer à chaque période et par Pa_k^i son potentiel d'action à la période t_i tel que $Pa_k^i \leq \mathcal{N}_k$. La table 2 montre un exemple de calcul du Pa_k^i pour les différents types de perception. Comme vous pouvez le constater, le potentiel d'action est proportionnel à $\mathcal{N}_k$ avec la relation $\frac{1}{2^{r-1}}$ où $r \in \mathbb{N}^*$ est le rang de la perception courante dans l'ordre des préférences de l'agent. Pour le coopérateur, l'ordre des préférences est : coopérative $\succ$ coopétitive $\succ$ compétitive. Pour le compétiteur, l'ordre des préférences est : compétitive $\succ$ coopétitive $\succ$ coopérative. Enfin, pour le coopétiteur, l'ordre des préférences est : coopétitive $\succ$ coopérative, compétitive. Ainsi, lorsque la perception courante est de même nature que le rôle de l'agent, le potentiel d'action est égal à $\mathcal{N}_k$. Sinon, il est égal à $\frac{\mathcal{N}_k}{2}$ ou $\frac{\mathcal{N}_k}{4}$ selon l'ordre des préférences.

TABLE 2 – Un exemple du calcul du Pa_k^i

La perception / Rôle de a_k	compétitive	coopétitive	coopérative
compétiteur	$\mathcal{N}_k$	$\frac{\mathcal{N}_k}{2}$	$\frac{\mathcal{N}_k}{4}$
coopétiteur	$\frac{\mathcal{N}_k}{2}$	$\mathcal{N}_k$	$\frac{\mathcal{N}_k}{2}$
coopérateur	$\frac{\mathcal{N}_k}{4}$	$\frac{\mathcal{N}_k}{2}$	$\mathcal{N}_k$

Le calcul du potentiel d'action Pa_k^i permet à chaque agent de déterminer le nombre d'agents à contacter pendant la période t_i. Nous partons du principe que dans une plateforme de crowdsourcing d'idées, les agents ont à la fois tendance à interagir avec des agents inconnus (i.e. absence d'historique interactif) et des agents connus (i.e présence d'historique interactif). Ainsi, l'agent a_k va interagir avec deux listes d'agents :

— La première liste, notée $LUA_k \subset V \setminus \{PIT_k\}$, sera composée de $(1-\alpha) * Pa_k^i$ agents inconnus sélectionnés au hasard à partir de la plateforme.

— La deuxième liste, notée $LKA_k \subset PIT_k$, sera constituée de $\alpha * Pa_k^i$ agents connus sélectionnés à partir de PIT_k.

où $\alpha \in [0,1]$ est un paramètre de pondération qui indique la proportion attribuée à LKA_k du potentiel d'action.

L'agent doit être capable à la fois (1) de remplir chacune de ces deux listes et (2) de savoir comment interagir avec leurs membres. L'agent a_k remplit la liste LUA_k par des agents inconnus sélectionnés d'une manière aléatoire dans la plateforme. Pour interagir avec eux, il va appliquer les règles d'action décrites précédemment (voir tableau 1).

Pour la deuxième liste LKA_k, a_k doit évaluer la confiance de chacun des agents connus $a_j \in PIT_k$ basée sur leur historique interactif pour déterminer avec qui interagir et comment interagir. Dans la section suivante, nous décrivons comment la confiance dyadique est calculée et comment les agents interagissent les uns avec les autres.

3.2 La dimension dyadique

L'intérêt de la dimension environnementale de la confiance consiste à doter les agents de la capacité d'évaluer la perception utile pour déterminer leur potentiel d'action. Cependant, cela ne leur permet pas de spécifier avec qui interagir et comment interagir. Pour résoudre ce problème, nous considérons la dimension dyadique de la confiance. Basée sur l'historique interactif, elle guide le "trustor" (i.e., l'agent qui évalue la confiance) dans le choix du "trustee" (i.e., l'agent dont la confiance est évaluée) et, également dans le choix de l'action à entreprendre. Lors de l'évaluation de la confiance, il est important de savoir si cette évaluation est fiable. Bien qu'il existe plusieurs facteurs qui puissent être pris en compte pour mesurer la fiabilité, nous nous concentrerons sur l'un d'entre eux : le nombre d'interactions utilisées pour calculer la valeur de confiance dyadique. L'intuition derrière le facteur «nombre d'interactions » est que dans une société réelle, une expérience isolée (ou quelques unes) ne suffit pas pour bâtir un jugement correct sur quelqu'un. Nous avons besoin d'un certain nombre d'expériences avant de pouvoir dire comment est cet individu. Si ce n'est pas le cas, il y a une incertitude à considérer. Jøsang a décrit la confiance dans [13] par des probabilités incertaines. Une valeur de confiance est modélisée par une opinion qui se compose de trois valeurs dans l'intervalle $[0,1]$ représentant la croyance (b), le doute (d) dans la confiance d'un agent ainsi que l'incertitude (u). La somme de ces valeurs doit toujours égale à 1 (b + d + u = 1). Le modèle de confiance de Jøsang semble approprié pour nos exigences, car on peut distinguer si le manque de confiance résulte d'actions négatives ou d'une connaissance manquante d'un agent. Le calcul de la confiance dyadique qu'un agent a_k possède en un agent a_j à une période t_i est basé sur ces métriques :

$$b = \frac{p}{p+n+1} \qquad d = \frac{n}{p+n+1} \qquad u = \frac{1}{p+n+1} \quad (4)$$

où $p = \sum_{l=0}^{l=i} ac_{jl}^{+}$ et $n = \sum_{l=0}^{l=i} ac_{jl}^{-}$ correspondent respectivement au nombre d'actions positives et d'actions négatives.

Cependant, ces métriques n'incluent pas l'aspect temporel dans leur processus de construction. Intuitivement, une action plus récente est censée être plus importante qu'une action ancienne. Ainsi, nous combinons la mesure de confiance dyadique avec une fonction de temps, la fonction de la courbe d'oubli de Hermann [7], pour donner plus d'importance aux actions récentes. Celle-ci réduit les nombres p et n des actions positives et négatives dans le temps comme montré dans la formule 5, conduisant à un degré plus élevé d'incertitude.

$$p = \sum_{l=0}^{l=i} (ac_{jl}^{+} * e^{-\frac{t_i - t_l}{Z}}) \qquad n = \sum_{l=0}^{l=i} (ac_{jl}^{-} * e^{-\frac{t_i - t_l}{Z}}) \quad (5)$$

Maintenant, nous définissons la confiance dyadique entre le "trustor" a_k et le "trustee" a_j à la periode t_i, notée $DTrust(a_k, a_j, t_i)$, de la manière suivante :

$$DTrust(a_k, a_j, t_i) = \begin{cases} b & si \;\; u < \theta \\ \begin{cases} T_s & \text{si } Role_k = \text{coopérateur} \\ T_w & \text{si } Role_k = \text{compétiteur} \\ T_r & \text{si } Role_k = \text{coopétiteur} \end{cases} & sinon \end{cases} \quad (6)$$

Le paramètre θ est un seuil d'incertitude indiquant la fiabilité de l'évaluation de la confiance. Si l'incertitude est inférieure à ce seuil, l'évaluation de la confiance est fiable. Dans le cas contraire, une valeur de confiance est attribuée en fonction du rôle du "trustor". Si le "trustor" est un coopérateur, alors il préfère coopérer avec les autres impliquant l'existence d'une relation de confiance forte $T_s = 1$ avec le "trustee". Inversement, si le "trustor" est un compétiteur, il

préfère concurrencer les autres, ce qui implique l'existence d'une relation de confiance faible $T_w = 0$ avec le "trustee". Et enfin, si le "trustor" est un coopétiteur, alors nous assignons une valeur de confiance aléatoire $T_r \in [0, 1]$.

La capacité d'évaluer la confiance permet aux agents de raisonner sur les motifs sous-jacents des autres et, en conséquence, de dériver les différents types de relations entre eux. Cependant, une seule valeur de confiance peut être interprétée différemment d'un agent à un autre. Pour unifier et automatiser le processus d'interprétation, nous proposons une fonction de caractérisation pour les relations définie comme suit :

Définition 5 *(Caractérisation des relations) Soit λ_{inf} et λ_{sup} les seuils supérieur et inférieur de confiance, respectivement. Soit $DTrust(a_k, a_j, t_i)$ la confiance dyadique entre l'agent a_k et l'agent a_j à la période t_i et soit $\rho : [0, 1] \mapsto R$ la fonction de caractérisation qui associe les valeurs de confiance aux relations qu'elles représentent. Ainsi, les relations sont caractérisées comme suit :*

- *si $DTrust(a_k, a_j, t_i) \geq \lambda_{sup}$ alors, $\rho((a_k, a_j)) = R_{coo}$*
- *si $DTrust(a_k, a_j, t_i) \in]\lambda_{inf}, \lambda_{sup}[$ alors, $\rho((a_k, a_j)) = R_{cop}$*
- *si $DTrust(a_k, a_j, t_i) \leq \lambda_{inf}$ alors, $\rho((a_k, a_j)) = R_{com}$*

Par conséquent, une valeur de confiance élevée (resp. faible) entre deux agents indique une relation de coopération (resp. compétition) entre eux. Une valeur de confiance moyenne entre deux agents indique une relation de coopétition entre eux. Rappelons que cette caractérisation de relations résulte d'une évaluation de long terme de la confiance qui peut évoluer au cours du temps. Cela signifie qu'une relation coopétitive entre deux agents peut changer et devient compétitive ou coopérative.

Une fois qu'un agent a_k a évalué la confiance dyadique, et par la suite a déterminé le type de relations de tous les agents connus de son PIT_k, il doit décider avec quel(s) agent(s) interagir. La prise de décision sera basée sur un processus de classement de ces valeurs de confiance qui diffère d'un rôle à un autre. Pour un compétiteur, les valeurs de confiance sont classées par ordre croissant donnant la priorité d'interaction aux agents les moins dignes de confiance. Inversement, pour le coopérateur, les valeurs de confiance sont classées par ordre décroissant donnant la priorité d'interaction aux agents les plus dignes de confiance. Toutefois, pour le coopétiteur, nous assumons qu'il n'y a pas de processus de classement car ce dernier peut interagir avec les agents les moins et les plus dignes de confiance. Ainsi, les agents sont classés d'une manière aléatoire. Dans les trois cas, a_k sélectionne les $\alpha * Pa_k^i$ premiers agents à interagir avec. Pour le guider dans le choix de l'action à entreprendre, nous définissons pour chaque rôle un ensemble de règles d'action dépendantes de la confiance autrement dit, du type de la relation (a_k, a_j).

TABLE 3 – Règles d'action dépendantes de la confiance entre un agent a_k et un agent a_j

Type de (a_k, a_j) / Rôle de a_k	R_{com}	R_{cop}	R_{coo}
compétiteur	$p^+ = 0, p^- = 1$	$p^+ = DTrust, p^- = 1 - DTrust$	
coopérateur	$p^+ = DTrust, p^- = 1 - DTrust$		$p^+ = 1, p^- = 0$
coopétiteur	$p^+ = 0, p^- = 1$	$p^+ = DTrust, p^- = 1 - DTrust$	$p^+ = 1, p^- = 0$

Comme le montre la table 3, il existe deux classes de règles d'action pour chaque rôle. Dans la première classe de règles, les agents suivent un comportement uni-modal respectant la principe de la réciprocité directe. Cela veut dire que l'action effectuée par l'agent a_k envers l'agent a_j est dérivée directement du type de la relation (a_k, a_j). Cependant, dans la deuxième classe de règles, le type d'action effectuée par l'agent a_k envers l'agent a_j est conditionnelle où la probabilité d'effectuer une action positive correspond à la valeur de la confiance dyadique. Notons ici que les agents ne changent pas leur rôle, mais ils adaptent leur comportement en fonction de l'historique interactive. Dans la section suivante, nous présentons les résultats expérimentaux de l'évaluation de la performance de la plateforme.

4 Évaluation de la performance d'une plateforme homogène

Définissons tout d'abord notre environnement expérimental ainsi que la configuration des paramètres du système multi-agent. Les expériences ont été menées sur un réseau de coopétition simulé où la taille de la population $n = 1000$ agents, le nombre maximum d'actions possibles $\mathcal{N}_k = 4$, le nombre d'actions dans la page d'accueil $cardinal(Pset_i) = \frac{cardinal(E_{i-1})}{10}$ [5] et la valeur de $\alpha = 0.5$. Ainsi, chaque agent a la même chance d'interagir avec un agent connu et un agent inconnu dans la pla-

5. Le EdgeRank d'une page sur Facebook ne dépasse pas 10% des fans.

teforme. Les proportions des rôles ont été distribués de manière homogène dans la plateforme donnant lieu à trois groupes d'agents de même taille notés C_{coo}, C_{com} et C_{cop}. Cette distribution vise à analyser la performance d'une plateforme homogène pour voir si cela induit un équilibre entre les agents ou si un groupe surmonte les autres. Ainsi, nous ne mesurons pas la performance d'un agent, mais plutôt la performance cumulative de chaque groupe d'agents.

Dans ce travail, la performance est une mesure quantitative évaluant le niveau de motivation des agents dans une plateforme homogène. Rappelons que chaque agent a_k est doté d'un but de motivation reflétant son niveau d'engagement au sein de la plateforme (voir définition 1). Ainsi, pour atteindre son but à une période t_i, a_k doit effectuer un nombre minimum d'actions tel que son potentiel d'action $Pa_k^i > \frac{\mathcal{N}_k}{2}$. Par la suite, la performance d'un groupe est définie comme étant un ratio entre le nombre d'agents qui ont réussi à atteindre leurs buts et le nombre total d'agents dans le groupe.

$$performance_i(C_l) = \frac{|\{a_k \in C_l \,|\, Pa_k^i > \frac{\mathcal{N}_k}{2}\}|}{|C_l|} \quad \forall l \in \{coo, com, cop\} \qquad (7)$$

La figure 2 présente l'évolution de la performance relative des trois groupes d'agents d'une plateforme homogène sur 50 périodes. Il est intéressant de noter que, contrairement à ce qui était prévu, l'évolution de ces performances relatives est différente. Cela indique que les trois rôles ne suscitent pas le même niveau de motivation chez les agents. L'inspection visuelle de la figure nous permet d'identifier trois points importants. Tout d'abord, nous remarquons que le groupe des coopérateurs est totalement démotivé car la valeur de sa performance relative est presque nulle tout au long de la simulation. Cela est dû au fait que la dynamique sociale a connu une domination précoce du groupe des compétiteurs provoquant un manque de motivation chez les coopérateurs. Deuxièmement, nous remarquons que la performance relative du groupe des coopétiteurs est importante et finit par être la meilleure de la plateforme tout en conservant une valeur stable. Cela signifie que cette plateforme a convergé vers un état dans lequel le groupe des coopétiteurs domine. Troisièmement, il est intéressant de noter que la performance du groupe des compétiteurs comporte deux phases d'évolution. Dans l'intervalle des valeurs $[0, 30]$ l'évolution est positive. Cependant, dans l'intervalle de valeurs $[31, 50]$, l'évolution devient négative ce qui reflète une baisse de motivation chez les compétiteurs.

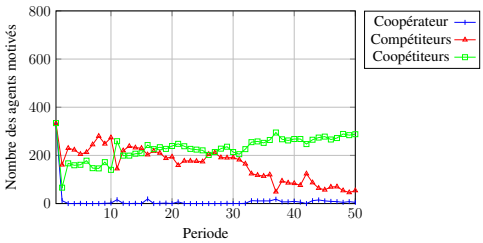


FIGURE 2 – L'évolution de la performance d'une plateforme homogène

Cela s'explique par le fait que dans le monde réel, il est difficile de construire rapidement des relations de confiance favorables à la coopération. Mais cette difficulté est progressivement surmontée lorsque les agents interagissent et établissent des relations de confiance des uns envers les autres. Selon ces observations, nous arrivons à la conclusion qu'une plateforme homogène est une plateforme efficace car elle est propice au développement d'un climat coopétitif et du coup, à la génération de bonnes idées.

5 Conclusion et perspectives

Dans ce travail, nous avons proposé une approche multi-agent basée sur la confiance pour évaluer la performance des plateformes de crowdsourcing d'idées. Pour ce faire, nous avons développé un modèle de confiance dynamique composé de deux dimensions : la dimension environnementale qui qualifie la perception et la dimension dyadique qui caractérise les relations entre les agents. Nous avons aussi défini pour chaque rôle d'agent (i.e., coopérateur, compétiteur et coopétiteur) un ensemble de règles d'action dépendantes et non dépendantes de la confiance pour guider leur processus décisionnel quand ils interagissent. Pour évaluer la performance de notre approche, nous avons mené des simulations multi-agents dans une plateforme avec une distribution homogène de rôles. Les résultats ont montré qu'une plateforme homogène est propice au développement d'un climat coopétitif et ainsi, à la génération de bonnes idées.

Une perspective de ce travail est de simuler d'autres distributions de rôles pour identifier la plateforme la plus performante. Nous envisageons aussi d'analyser statistiquement et d'interpréter visuellement les snapshots obtenus

pour mieux comprendre la dynamique sociale des agents. Par ailleurs, nous avons l'intention d'étudier l'influence des paramètres de la simulation sur le modèle et d'enquêter sur plusieurs questions telles que la façon de déterminer les valeurs appropriées des seuils de confiance utilisés dans la prise de décision.

Références

[1] A. Abdul-Rahman and S. Hailes. Supporting trust in virtual communities. In *Proceedings of the 33rd Hawaii International Conference on System Sciences - Volume 6*, pages 6007–6016, 2000.

[2] S. Adamczyk, A.C. Bullinger, and K.M. Möslein. Innovation contests : A review, classification and outlook. *Creativity and Innovation Management*, 21(04) :335–360, 2012.

[3] F. Beck, M. Burch, S. Diehl, and D. Weiskopf. The State of the Art in Visualizing Dynamic Graphs. In *EuroVis - STARs*. The Eurographics Association, 2014.

[4] K. J. Boudreau, K. R. Lakhani, and M. Menietti. Performance responses to competition across skill levels in rank-order tournaments : field evidence and implications for tournament design. *RAND Journal of Economics*, 47(1) :140–165, 2016.

[5] C. Castelfranchi and R. Falcone. Principles of trust for mas : Cognitive anatomy, social importance, and quantification. In *ICMAS*, pages 72–79. IEEE Computer Society, 1998.

[6] G. Dagnino, F. Le Roy, and S. Yami. La dynamique des stratégies de coopétition. *Revue Française de Gestion*, 33(176) :87–98, 2007.

[7] H. Ebbinghaus. *Memory : A contribution to experimental psychology*. Teachers college, Columbia university, 1913.

[8] K. Frey, C. Lüthje, and S. Haag. Whom should firms attract to open innovation platforms ? The role of knowledge diversity and motivation. *Long Range Planning*, 44(5) :397–420, 2011.

[9] C.-H. et al. Fu. A kind of collaboration-competition networks. *Physica A : Statistical Mechanics and its Applications*, 387(5-6) :1411–1420, 2008.

[10] J. J. Gibson. *The ecological approach to visual perception*. Houghton Mifflin, 1979.

[11] J. Hu and H. Zhu. Adaptive bipartite consensus on coopetition networks. *Physica D Nonlinear Phenomena*, 307 :14–21, 2015.

[12] K. Hutter, J. Hautz, J. Füller, J. Mueller, and K. Matzler. Communitition : The tension between competition and collaboration in community-based design contests. *Creativity and Innovation Management*, 20(1) :3–21, 2011.

[13] A. Jøsang. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(3) :279–311, 2001.

[14] J. H. Jung, C. Schneider, and J. Valacich. Enhancing the motivational affordance of information systems : The effects of real-time performance feedback and goal setting in group collaboration environments. *Manage. Sci.*, 56(4) :724–742, 2010.

[15] R. Kurzban and D. Houser. Experiments investigating cooperative types in humans : A complement to evolutionary theory and simulations. *Proceedings of the National Academy of Sciences of the United States of America*, 102(5) :1803–1807, 2005.

[16] S.S. Levine and M.J. Prietula. Open collaboration for innovation : Principles and performance. *Organization Science*, 19(5) :1414–1433, 2014.

[17] L. Muhdi and R. Boutellier. Motivational factors affecting participation and contribution of members in two different swiss innovation communities. *IJIM*, 15(03) :543–562, 2011.

[18] E. Pelegrin-Boucher and G. Guegen. Stratégies de "coopétition" au sein d'un écosystème d'affaires : une illustration à travers le cas de sap. *Finance Contrôle Stratégie*, 8(1) :109–130, 2005.

[19] M. Szell, R. Lambiotte, and S. Thurner. Multirelational Organization of Large-scale Social Networks in an Online World. In *Proceedings of the National Academy of Science*, pages 13636–13641, 2010.

[20] G. Tellis, J. Füller, K. Hutter, and C. Riedl. Crowdsourcing innovations : external versus internal incentives in activity and performance. In *EMAC*, page ., 2015.

[21] Z. Zhao, D. Renard, M. Elmoukhliss, and C. Balagué. What affects creative performance in idea co-creation : competitive, cooperative or coopetitive climate ? *International Journal of Innovation Management*, 20(4) :1–24, 2016.

Formation de coalitions pour une composition de services Web fondée sur la confiance dans les réseaux sociaux

Amine Louati[a] amine.louati@telecom-em.eu
Joyce El Haddad[b] elhaddad@lamsade.dauphine.fr
Suzanne Pinson[b] pinson@lamsade.dauphine.fr

[a]Institut mines Télécom, Télécom École de Management, LITEM, France

[b]PSL, Université Paris-Dauphine, LAMSADE CNRS UMR 7243, France

Résumé

Avec le nombre croissant de services Web publiés dans les réseaux sociaux, beaucoup d'approches de composition de services ont été proposées dans la littérature. Cependant, elles ne prennent pas en considération le contexte social incluant la confiance et les relations sociales entre les demandeurs et les fournisseurs ainsi que l'autonomie des fournisseurs pour décider avec qui collaborer. Pour relever ce défi, nous proposons l'utilisation des systèmes multi-agents, car ils ont la capacité de former des coalitions de partenaires dignes de confiance. Nous proposons un nouveau processus de formation de coalitions pour la composition de services fondé sur la confiance dans les réseaux sociaux. En particulier, notre processus de formation de coalitions est incrémental, dynamique et recouvrant. Les agents sont équipés d'un ensemble de services et coopèrent pour répondre à la requête du demandeur en se basant sur un processus de prise de décision décentralisé guidé par la confiance. Nous présentons les résultats de l'évaluation des premières expériences pour démontrer la validité de notre approche.

Mots-clés : *Système multi-agent, composition de services Web, modèle de confiance, formation de coalitions, réseaux sociaux.*

Abstract

With the growing number of published Web services in social networks, a lot of service composition approaches have been proposed in the literature. However, they often fail to take into consideration the social context including trust and social relationships between requesters and providers as well as the providers autonomy in deciding with whom to collaborate. To address this challenge, we propose the use of Multi-Agent Systems, as they have the ability to form coalitions of trusted partners. We propose a new coalition formation process for service composition based on trust in social networks. In particular, our coalition formation process is incremental, dynamic and overlapping. Agents are equipped with a set of services and cooperate to fulfill the requester query based on a decentralized decision-making process guided by trust. We present the evaluation results of first experiments to demonstrate the validity of our approach.

Keywords: *Multi-agent system, Web services composition, Trust model, Coalition formation, Social networks.*

1 Introduction

Les derniers progrès réalisés dans le domaine de l'informatique orientée services ont permis la publication, la localisation et l'invocation des services dans *les réseaux sociaux*. De plus en plus d'utilisateurs expriment l'intérêt de se servir des réseaux sociaux pour proposer et/ou chercher des services. Prenons l'exemple d'un utilisateur qui recherche un voyage dans son réseau social. La routine de son voyage comprend : le transport du domicile à l'aéroport, le vol, le transport de l'aéroport à l'hôtel, l'hébergement en hôtel et les visites touristiques. Dans le cas où aucun service atomique ne pourrait satisfaire le besoin complexe du demandeur de services, il devrait être possible de composer plusieurs services pour y parvenir. La composition de services Web est le processus qui crée une application à valeur ajoutée en combinant les services existants dans un service composite qui a le potentiel de réduire l'effort et le temps de développement.

Le problème de composition de services Web a suscité l'intérêt de nombreux travaux de recherche (par exemple, [1, 2, 4]). Cependant, les techniques proposées ne prennent pas en considération la dimension sociale incluant *la confiance* et les relations sociales entre les demandeurs et les fournisseurs et les informations issues de leurs expériences précédentes. Comme les agents ont démontré la capacité de promouvoir la représentation de connaissances et les interactions ainsi que des métaphores sociales comme la confiance, nous proposons l'utilisa-

tion d'une approche multi-agent pour réaliser *une composition de services Web fondée sur la confiance*. Nos agents sont autonomes et équipés d'un ensemble de services avec les valeurs de leurs attributs de QoS. Étant donnée que plusieurs agents peuvent fournir les mêmes services avec différentes valeurs de QoS, la coopération entre eux est essentielle pour déterminer toutes les compositions de services possibles. Nous pensons que la formation de coalitions est bien adaptée à la modélisation de la coopération entre des agents ayant un objectif commun qui est la composition de services dans un environnement distribué comme le réseau social. Dans ce contexte, trois défis se présentent :

1) Intégrer *la dimension sociale* dans le processus de composition : généralement, les demandeurs préfèrent les fournisseurs qui proposent non seulement les services requis mais qui sont aussi dignes de confiance. La plupart des travaux antérieurs sur la formation de coalitions pour la composition de services [5, 7, 12] ne prennent pas en compte la contribution de la dimension sociale au cours du processus de formation de coalitions (PFC). Le choix des membres se fait en utilisant les attributs de QoS sans aucune importance accordée à la confiance entre le demandeur et les fournisseurs.

2) Accorder de *l'autonomie aux fournisseurs* pour décider avec qui ils coopèrent : comme les demandeurs, les fournisseurs de services sont des agents qui doivent avoir la possibilité de sélectionner localement leurs partenaires dans le service composite afin de satisfaire leur propre bien-être. Griffiths et Luck ont présenté dans [6] un modèle de formation de coalitions qui combine la confiance et la motivation pour générer des coopérations réussies entre des agents égoïstes. Bien que ce modèle prenne en compte la dimension sociale, son processus de prise de décision associé à la sélection d'un candidat manque d'une vision globale. Les auteurs accordent de l'autonomie aux candidats pour décider de participer ou non à la coalition en se basant sur leur confiance envers l'initiateur de la coalition. Cependant, les membres de la coalition ne peuvent pas décider d'accepter ou de refuser l'adhésion d'un candidat.

3) Incorporer de *la dynamique* dans le PFC : une coalition dynamique signifie qu'elle peut évoluer au cours du temps en fonction de l'état de ses membres. Bourdon et al. ont proposé dans [3] un modèle multi-agent basé sur un broker capable de trouver des coalitions dignes de confiance centrées fournisseurs pour accomplir une composition de services Web. Bien que leur modèle intègre la dimension sociale en considérant la confiance entre les agents et accorde de l'autonomie aux fournisseurs dans le choix de leurs partenaires, il ne permet pas aux agents de joindre ou de quitter les coalitions dynamiquement. Un problème se pose quand un membre est insatisfait de la présence d'un agent spécifique et ne peut pas quitter sa coalition courante pour en rejoindre une autre.

Cet article est une version traduite et actualisée d'un article présenté à la conférence WISE [10]. Dans ce travail, nous proposons un nouveau modèle multi-agent basé sur un broker pour une composition dynamique de services. Le processus de composition est réalisé par des agents qui coopèrent dans des coalitions pour fournir collectivement plusieurs services composites répondant à la requête complexe de l'utilisateur. Pour relever les défis susmentionnés, nous intégrons la dimension sociale en utilisant un mécanisme de confiance permettant à chaque agent d'évaluer la fiabilité des autres agents impliqués avant de coopérer avec eux [11, 13]. Nous accordons l'autonomie aux fournisseurs pour décider avec qui ils participent dans le service composite en se basant sur la confiance en la coopération. La satisfaction de tous les membres donne lieu à une coalition stable et efficace. Si certains membres ne sont pas satisfaits, nous devons disposer d'un mécanisme qui permet de retrouver la stabilité de la coalition. Enfin, nous intégrons la dynamique dans notre PFC pour permettre aux membres insatisfaits de quitter la coalition.

La suite de cet article est structurée de la manière suivante. La section suivante présente les principaux concepts utilisés dans ce travail et décrit notre modèle multi-agent basé sur un broker. La section 3 décrit notre processus de formation de coalitions avec ses trois phases. La section 4 montre quelques résultats expérimentaux. La section 5 conclut le papier et identifie les travaux futurs.

2 Modèle multi-agent

Cette section est divisée en deux parties. Nous définissons dans un premier temps les concepts que nous allons utiliser pour la description du processus de formation de coalitions puis nous donnons un aperçu général de notre modèle multi-agent basé sur un broker en précisant les différents rôles considérés.

2.1 Définition des concepts

Un réseau social est modélisé par un graphe orienté où chaque noœud est représenté par un agent autonome, et chaque arc entre deux utili-

sateurs correspond à une relation de confiance entre eux.

Définition 1 (Réseau de confiance) *Soit* A=$\{a_1, .., a_s\}$ *un ensemble d'agents et* $E \subseteq A \times A$ *un ensemble d'arcs, un réseau social* $G =< A, E >$ *est un graphe orienté et connecté où chaque arc* $(a_k, a_j) \in E$ *représente une relation sociale de confiance entre* a_k *et* a_j.

Ce réseau social mono-relationnel est considéré comme un système multi-agent dans lequel les agents sont équipés d'un ensemble de services.

Définition 2 (Agent) *Un agent* $a_k \in A$ *est défini par un tuple* $a_k =< S_k, Trust, CC, EC, \lambda Inf_k, \lambda Sup_k, \gamma_k, Blist_k >$ *où :*

- S_k *est l'ensemble des* m_k *services offerts avec les valeurs de QoS,*
- *Trust*(a_k, a_j) *est la confiance de l'agent* a_k *en un autre agent* a_j*. Elle est définie comme étant l'agrégation de la confiance en la crédibilité sociale et la confiance en la recommandation (pour plus de détail, voir [9]),*
- $CC(a_k, a_j)$ *représente la confiance en la coopération de l'agent* a_k *en un autre agent* a_j *(voir définition 7),*
- $EC(a_k, a_j, s)$ *désigne la confiance en expertise que l'agent* a_k *accorde à un service* s *offert par un fournisseur* a_j *qui est définie comme l'agrégation des trois attributs de qualité de services à savoir : la spécialisation*[1]*, la disponibilité*[2] *et la qualité*[3] *(pour plus de détail, voir [9]),*
- λInf_k *et* $\lambda Sup_k \in [0, 1]$ *sont respectivement, les seuils inférieur et supérieur de la confiance en la coopération,*
- $\gamma_k \in [0, 1]$ *est le seuil de la confiance en une coalition.*
- $Blist_k$ *est une liste noire formée par un ensemble d'agents avec lesquels* a_k *fait assez peu confiance pour ne pas avoir envie de coopérer avec eux.*

Définition 3 (Service) *Un service* s *est un n-uplet* $\{in, out, f, q^1, \ldots, q^d\}$ *où* in *est un ensemble d'entrées requises pour utiliser le service,* out *est un ensemble de sorties prévues à la fin de l'exécution du service,* f *est la fonctionnalité offerte, et* $q^1, \ldots, q^d$ *sont les valeurs des* $d \in \mathbb{N}$ *attributs de QoS.*

1. La spécialisation d'un service s est son pourcentage d'utilisation par rapport aux autres services offert par le même agent.

2. La disponibilité d'un service s est la probabilité qu'il soit opérationnel au moment de l'invocaion.

3. La qualité d'un service s est la moyenne des notes qui lui sont attribuées par les utilisateurs après utilisation.

Définition 4 (Requête utilisateur) *Soit* F *le domaine de définition des fonctionnalités atomiques disponibles. Une requête utilisateur* $Q = \{f_1, f_2, \ldots, f_n \,|\, \forall i \in [1, n], f_i \in F\}$ *est un ensemble fini de fonctionnalités requises.*

Nous désignons par $A_p \subseteq A$ l'ensemble des fournisseurs dignes de confiance, par $A_i = \{a_k \,|\, a_k \in A_p \text{ et } a_k.s.f \equiv f_i\}$ l'ensemble des fournisseurs offrant un service avec la fonctionnalité f_i et par $A_Q = \bigcup_{i=1}^{n} A_i$ l'ensemble de fournisseurs offrant des services pour toutes les fonctionnalités requises dans Q.

Comme plusieurs agents peuvent offrir différents services i.e. avec différentes valeurs de QoS, pour les mêmes fonctionnalités requises, la formation de coalitions peut résoudre le problème de la composition.

Définition 5 (Coalition) *Soit* Q *une requête utilisateur. Une coalition* $c = \{(f_1, x_1), \ldots, (f_i, x_i), \ldots, (f_n, x_n) \,|\, \forall i \in [1, n], \exists k \in [1, s]$ *tel que* $x_i = a_k$ *et* $a_k \in A_i\}$ *est un ensemble d'agents satisfaisant* Q.

Durant le processus de formation de coalitions (PFC), les agents s'organisent dans des coalitions où chacun est en mesure de satisfaire une ou plusieurs fonctionnalités requises. Une coalition qui ne contient pas un ensemble d'agents satisfaisant toutes les fonctionnalités requises dans la requête Q est appelée *coalition intermédiaire*. Une coalition intermédiaire, notée c_z, est une instanciation partielle de Q telle que $c_z = \{(f_1, x_1), (f_2, x_2), \ldots, (f_n, x_n) \,|\, \exists i \in [1, n]$ tel que $x_i = \emptyset\}$. Durant le PFC, le contenu d'une coalition intermédiaire évolue. La transition d'une coalition intermédiaire c_z à une autre c_{z+1} est effectuée à l'aide d'un *proposal*.

Définition 6 (Proposal) *Un proposal* $\phi = \{(f_1, x_1), (f_2, x_2), \ldots, (f_n, x_n)\}$ *tel que* $x_i \in \{a_k, \emptyset\}$ *et* $a_k \in A_i$ *représente soit une demande d'adhésion soit une offre d'adhésion. Dans le cas d'une demande d'adhésion,* $\phi = \{(f_1, x_1), (f_2, x_2), \ldots, (f_n, x_n)\}$ *tel que* $\exists! i \in [1, n]$ *tel que* $x_i = a_k \in A_i$ *et* $\forall j \neq i, x_j = \emptyset$*. Dans le cas d'une offre d'adhésion,* $\phi = c_z$.

La demande d'adhésion contient un seul candidat a_k satisfaisant une fonctionnalité f_i. Cette demande sera envoyée par le broker à chacun des membres d'une coalition intermédiaire c_z. L'offre d'adhésion contient l'ensemble des membres d'une coalition intermédiaire c_z. Cette offre sera envoyée par le broker à un candidat a_k pour rejoindre cette coalition intermédiaire c_z.

2.2 Modèle multi-agent basé sur un broker

S'appuyant sur les travaux de Klusch et Sycara [8], notre modèle multi-agents est un modèle où intervient un broker comme représenté sur la Figure 1. Il englobe trois rôles différents :

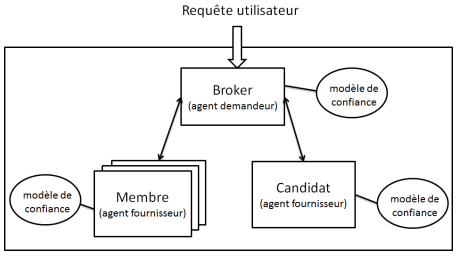


FIGURE 1 – Notre modèle multi-agents basé sur un broker

— Le rôle *candidat :* un candidat est un fournisseur de services susceptible de rejoindre une coalition.
— Le rôle *membre :* un membre est un fournisseur de services qui est attribué à une ou à plusieurs coalitions.
— Le rôle *broker :* un broker est un demandeur de services qui se charge de la mise en relation des candidats avec les membres durant le PFC et orchestre les interactions entres eux. Dans le cas de plusieurs coalitions, il choisit la coalition qui a la meilleure valeur d'expertise.

3 Description du processus de formation de coalitions

Habituellement, la composition de services est effectuée à la suite d'une phase préalable de découverte de services. Les phases de découverte et de sélection de services décrites dans un travail antérieur [9] permet à l'agent demandeur a_r d'identifier à partir *d'un réseau social multi-relationnel* (voir Fig. 2(a)) un ensemble A_p de fournisseurs dignes de confiance sur lequel il va lancer le processus de formation de coalitions (PFC). Cet ensemble de fournisseurs est situé dans un arbre (voir Fig. 2(b)), appelé *réseau social de confiance (RSC)*, où chaque couple de valeurs d'un arc indique la confiance en la sociabilité et la confiance en la recommandation entre deux agents. Si nous considérons l'agent demandeur a_r comme la racine de l'arbre, chaque fournisseur $a_k \in A_p$ est localisé par une chaîne d'agents dignes de confiance appelée *chaîne fournisseur-recommandeur*. Une chaîne fournisseur-recommandeur est une séquence père-fils d'agents commençant à l'agent demandeur a_r et conduisant à un agent fournisseur a_k dans laquelle les agents intermédiaires sont soit des fournisseurs soit des recommandeurs [4]. La figure 3 illustre cette structure arborescente en utilisant une représentation en multi-couche où a_r est la racine et chaque couche représente un niveau dans l'arbre.

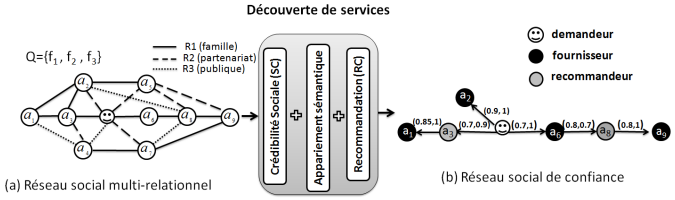


FIGURE 2 – Représentation plane du réseau social de confiance

Dans le RSC, les agents coopèrent dans une formation de coalitions pour fournir des services composites qui répondent à la requête complexe de l'utilisateur. Notre PFC est original car il est :
— *Incrémental :* Le PFC se fait d'une manière itérative où à chaque itération un seul agent peut rejoindre la coalition en favorisant en premier lieu les fournisseurs découverts dans la première couche du RSC. Le processus se poursuit en cherchant des fournisseurs dans la seconde couche et ainsi de suite.
— *Dynamique :* Les agents peuvent rejoindre et quitter leurs coalitions de façon autonome à tout moment en fonction des messages reçus et de leurs décisions locales fondées sur la confiance.
— *Overlapping :* Étant donné qu'un fournisseur peut offrir plusieurs services, il peut appartenir à plusieurs coalitions en même temps. Par contre, il ne peut pas participer à plusieurs coalitions pour un même service offert.

Le PFC se compose de trois phases séquentielles : la phase de génération des coalitions initiales, la phase de sélection des membres et la phase de sélection de la meilleure coalition.

3.1 Phase de génération de coalitions initiales

Le but de cette phase est de générer un ensemble de coalitions initiales $\mathcal{C}$ dans le RSC. Les entrées de l'algorithme 1 sont l'ensemble des fonctionnalités requises Q, l'ensemble des fournisseurs A_p et le réseau social de confiance RSC. Initialisant $\mathcal{C}$ à $\emptyset$ et son rôle à $brok$, a_r commence par l'identification des fournisseurs dans la première couche RSC(1) qui est à une profondeur 1. Ensuite, il attribue chaque fournisseur a_k à une nouvelle coalition initiale c_z (par exemple,

4. Un recommandeur est un agent qui n'a pas de services utiles mais peut être bien connecté en recommandant un fournisseur pertinent.

dans la Fig. 3, $\mathcal{C} = \{c_1 = \{a_2\}, c_2 = \{a_6\}\}$) et envoie à a_k un message INFORM contenant l'indice de la coalition dans laquelle il est membre (voir Algo. 1 lignes $4-5$). Nous faisons l'hypothèse que les agents sont de bonne volonté pour rejoindre la coalition initiale vide. Cependant, dans la suite ils seront complètement autonomes dans leur prises de décisions de rejoindre ou de quitter une coalition intermédiaire ou d'accepter une demande d'adhésion. Sur réception d'un message INFORM contenant l'indice d'une coalition (voir Algo. 3 ligne 12), un agent a_k prend le rôle de membre (voir Algo. 3 ligne 13). Chaque nouvelle coalition initiale est ajoutée à $\mathcal{C}$ (voir Algo. 1 ligne 6).

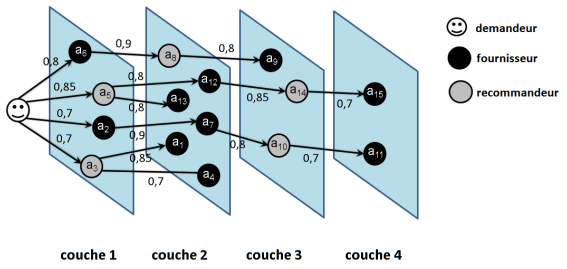


FIGURE 3 – Représentation multi-couche du réseau social de confiance

À la fin de cette phase, le broker détermine un ensemble $\mathcal{C}$ de coalitions initiales sur lesquelles il lancera la deuxième phase du PFC qui est la phase de sélection des membres.

Algorithm 1: Algorithme de génération de coalitions initiales

Input: Q est la requête utilisateur, A_p est l'ensemble des fournisseurs, RSC est le réseau social de confiance où RSC(1) est la première couche.
Variables: $\mathcal{C} = \{c_z\}$ est l'ensemble des coalitions, $role_r$ est un tableau de $|\mathcal{C}|$ valeurs $\in \{brok, mem, cand\}$ initialement fixées à $brok$.

1 $\mathcal{C} \leftarrow \emptyset$;
2 $z \leftarrow 1$;
3 **for all** $(a_k \in RSC(1) \cap A_p)$ **do**
4 c_z = new initial_coalition(a_k, Q);
5 Inform$(a_r, a_k, (z, mem))$; /* voir Algo. 3 ligne 12 */
6 $\mathcal{C}.add(c_z)$;
7 $z \leftarrow z + 1$;
8 **return** $\mathcal{C}$;

3.2 Phase de sélection des membres

L'objectif de cette phase est de compléter séquentiellement chacune des coalitions initiales $c_z \in \mathcal{C}$ par des membres fournissant des services nécessaires à la réalisation du service composite Q. Pour plus de simplicité, nous limitons la description de cette phase à une seule coalition c_z.

La phase de sélection des membres est une phase itérative gérée par le broker a_r où à la fin de chaque itération, un seul membre peut rejoindre c_z. Elle est basée sur une stratégie de recherche qui consiste à donner la priorité d'adhésion aux fournisseurs appartenant aux couches supérieurs de l'arbre i.e., plus proches du demandeur a_r, car une plus grande proximité implique une plus grande confiance. Rappelons que notre PFC est dynamique par conséquent, il est essentiel d'appliquer un mécanisme de timeout $timer_r[z]$ associé à c_z qui permet au broker d'arrêter le processus s'il dure trop longtemps (voir Algo. 2 ligne 4).

Algorithm 2: Algorithme de sélection des membres

Input: A_p est l'ensemble des fournisseurs, $\mathcal{C} = \{c_z\}$ est l'ensemble des coalitions résultat de l'Algo 1, RSC est le réseau social de confiance, l_{max} est la couche maximale dans RSC.
Variables: $timer_r$ est un tableau de $|\mathcal{C}|$ entier, $candP_r$ est un tableau de $|\mathcal{C}|$ ensembles de fournisseurs candidats initialement $\emptyset$, $func_r$ est un tableau de $|\mathcal{C}|$ ensembles de fonctionnalités initialement $\emptyset$, $count_r$ est un tableau de $|\mathcal{C}|$ entier initialement fixés à 0, $reply_r$ est un tableau de $|\mathcal{C}|$ boolean initialement fixés à False.

1 $z \leftarrow 1$;
2 **for all** $(c_z \in \mathcal{C})$ **do**
3 $l \leftarrow 1$;
4 Activate$(timer_r[z])$;
5 **while** $((timer_r[z])$ *and* $(\exists x \in c_z \mid x \equiv f)$ *and* $(l \leq l_{max}))$ **do**
6 $func_r[z] \leftarrow$ functionalities_identification(c_z);
7 $candP_r[z] \leftarrow$ candidates_identification$(func_r[z], RSC(l) \cap A_p)$;
8 **while** $((candP_r[z] \neq \emptyset)$ *and* $(\exists f_i \in func_r[z]))$ **do**
9 $a_j \leftarrow Argmax_{a_t \in candP_r[z] \cap A_i} Trust(a_r, a_t)$;
10 **for all** $(a_k \in c_z)$ **do**
11 Propose$(a_r, a_k,$ MembershipRequest$(z, \phi(a_j)))$; /* voir Algo. 3 ligne 1 */
12 wait $(count_r[z] == |c_z|)$;
13 $count_r[z] \leftarrow 0$;
14 **if** $(|OK| > \frac{|c_z|}{2})$ **then**
15 Propose$(a_r, a_j,$ MembershipOffer$(z, \phi(c_z)))$; /* voir Algo. 4 ligne 1 */
16 **else**
17 $candP_r[z] \leftarrow candP_r[z] \setminus \{a_j\}$;
18 $reply_r[z] \leftarrow True$;
19 $count_r[z] \leftarrow |c_z|$;
20 wait $(reply_r[z] == True)$;
21 $reply_r[z] \leftarrow False$;
22 wait $(count_r[z] == |c_z|)$;
23 $count_r[z] \leftarrow 0$;
24 $l \leftarrow l + 1$;
25 $z \leftarrow z + 1$;
26 **return** $\mathcal{C}$;

Pour chaque couche RSC(l), a_r identifie un ensemble de fournisseurs susceptibles de rejoindre c_z. Ceci se fait en déterminant d'abord l'ensemble des fonctionnalités non satisfaites $func_r[z]$ dans c_z et ensuite en identifiant les fournisseurs $candP_r[z]$ qui offrent des services avec ces fonctionnalités non satisfaites (voir Algo. 2 lignes $6-7$). Dans le cas où il n'y a que des recommandeurs dans cette couche et/ou tous les fournisseurs identifiés ne proposent pas de services avec les fonctionnalités requises dans $func_r[z]$ ($candP_r[z] = \emptyset$), a_r étend l'identification des fournisseurs à la couche suivante RSC$(l+1)$. Le passage d'une couche l à une autre couche $l+1$ se fait si et seulement si toutes les fonctionnalités requises dans $func_r[z]$ sont satisfaites ou l'ensemble des fournisseurs identifiés $candP_r[z]$ est vide (voir Algo. 2 lignes 8 et 24). Dans le cas où a_r dispose d'un ensemble d'agents susceptibles de rejoindre c_z

($candP_r[z] \neq \emptyset$), il sélectionne un candidat parmi cet ensemble en se fondant sur sa valeur de confiance et transmet sa demande d'adhésion aux membres de c_z.

Plus précisément, a_r ordonne les fournisseurs identifiés dans $candP_r[z]$ en se basant sur leurs valeurs de confiance et choisit le plus digne de confiance pour être candidat (voir Algo. 2 ligne 9). Puis, il déclenche une conversation de sélection d'un membre en engageant les membres de c_z et le candidat choisi.

Algorithm 3: Algorithme de demande d'adhésion

Variables: $role_k$ est un tableau de $|\mathcal{C}|$ valeurs $\in \{brok, mem, cand\}$ initialement fixées à $cand$, $Blist_k$ est une liste noire d'agents initialement vide, $NbSoll_k[|A|]$ est un tableau d'entier.

```
1  Procedure (RECEIVE_PROPOSE(a_r, a_k, MembershipRequest(z, φ(a_j))))
2      if (CC(a_k, a_j) > λSup_k and a_j ∉ Blist_k) then
3          Accept_Proposal(a_k, a_r, a_j);
4          NbSoll_k[j] ← NbSoll_k[j] + 1;
5      else
6          Reject_Proposal(a_k, a_r, a_j);
7  Procedure (RECEIVE_ACCEPT_PROPOSAL(a_k, a_r, a))
8      count_r[z] ← count_r[z] + 1;
9      |OK| ← |OK| + 1;
10 Procedure (RECEIVE_REJECT_PROPOSAL(a_k, a_r, a))
11     count_r[z] ← count_r[z] + 1;
12 Procedure (RECEIVE_INFORM(a_r, a_k, (z, mem))
13     role_k[z] ← mem;
```

Après avoir choisi un candidat a_j, le broker a_r envoie à chaque membre a_k de c_z un message PROPOSE contenant une demande d'adhésion de a_j (voir Algo. 2 lignes $10-11$). Sur réception d'un message PROPOSE (voir Algo. 3 ligne 1), un membre a_k décide localement d'accepter ou de rejeter la demande d'adhésion de a_j.

Comme mentionné ci-dessus, le processus de prise de décision d'un agent est fondé sur la confiance. Il pourrait également se baser sur le résultat de la vérification de la liste noire $Blist_k$(voir Algo. 3 ligne 2) : un membre a_k peut refuser de coopérer avec un agent candidat a_j si ce dernier est présent dans sa liste noire. Ainsi, une demande d'adhésion d'un candidat contenu dans cette liste noire est directement rejeté suivi de l'envoi d'un message REJECT_PROPOSAL au broker a_r (voir Algo. 3 lignes $5-6$). Sinon, a_k évalue la confiance en la coopération qu'il a en a_j en se basant sur l'historique des coalitions antérieures dans lesquelles ils étaient tous les deux impliqués. La confiance en la coopération $CC(a_k, a_j)$ d'un agent a_j selon le point de vue d'un autre agent a_k est donnée par la définition 7 comme suit :

Définition 7 (Confiance en la coopération (CC)) *Soit $NbSoll_k[j]$ le nombre total de sollicitations de a_k auprès de a_j pour qu'il rejoigne sa coalition intermédiaire et soit $NbMem_k[j]$ le nombre total d'adhésions de a_j à la coalition intermédiaire dans laquelle a_k est un membre.*

$$CC(a_k, a_j) = \begin{cases} 1 & \text{si} \quad NbSoll_k[j] = 0 \\ \frac{NbMem_k[j]}{NbSoll_k[j]} & \text{sinon} \end{cases} \quad (1)$$

Algorithm 4: Algorithme d'offre d'adhésion

Variables: $evalC_k$ est un tableau de $|\mathcal{C}|$ évaluation de la confiance des coalitions, $NbMem_k[|A|]$ est un tableau d'entier.

```
1  Procedure (RECEIVE_PROPOSE(a_r, a_k, MembershipOffer(z, φ(c))))
2      if (∃a_t ∈ c and a_t ∈ Blist_k) then
3          Reject(a_k, a_r, (c, z));                    /* voir ligne 28 */
4      else
5          evalC_k[z] ← evalC(a_k, c);
6          if (evalC_k[z] > β_k) then
7              Accept_Proposal(a_k, a_r, (c, z));       /* voir ligne 19 */
8              role_k[z] ← mem;
9          else
10             Reject_Proposal(a_k, a_r, (c, z));       /* voir ligne 28 */
11 Procedure (RECEIVE_FAILURE(a_k, a_r, (a_k ∉ c)))
12     c_z.remove(a_k);
13     for all (t | a_k ∈ A_t) do
14         func_r[z] ← func_r[z] ∪ {f_t};
15         v ← 1;
16         for all (v ≤ l) do
17             candP_r[z] ← candP_r[z] ∪ {RSC(v) ∩ A_t \ {a_k}};
18     count_r[z] ← count_r[z] + 1;
19 Procedure (RECEIVE_ACCEPT_PROPOSAL(a_k, a_r, (z, c)))
20     for all (a_t ∈ c_z) do
21         Inform(a_r, a_t, (z, a_k ∈ c_z));            /* voir ligne 32 */
22     c_z.add(a_k);
23     func_r[z] ← func_r[z] \ {f_i};
24     candP_r[z] ← candP_r[z] \ {a_k};
25     for all (a_t ∈ candP_r[z] ∩ A_i) do
26         candP_r[z] ← candP_r[z] \ {a_t};
27     reply_r[z] ← True;
28 Procedure (RECEIVE_REJECT_PROPOSAL(a_k, a_r, (z, c)))
29     candP_r[z] ← candP_r[z] \ {a_k};
30     count_r[z] ← |c_z|;
31     reply_r[z] ← True;
32 Procedure (RECEIVE_INFORM(a_r, a_k, (z, a_j ∈ c))
33     if ((answer_k == KO) and (CT(a_k, a_j) < λInf_k)) then
34         role_k[z] ← cand;
35         Blist_k.add(a_j);
36         Failure(a_k, a_r, a_k ∉ c_z);                /* voir ligne 11 */
37     else
38         NbMem_k[l] ← NbMem_k[l] + 1;
39         Confirm(a_k, a_r, Yes);                      /* voir ligne 40 */
40 Procedure (RECEIVE_CONFIRM(a_k, a_r, Yes))
41     count_r[z] ← count_r[z] + 1;
```

La décision d'un membre a_k d'accepter ou de rejeter la demande d'adhésion d'un candidat a_j repose sur sa valeur de confiance en la coopération : si elle dépasse λSup_k alors a_k accepte la demande d'adhésion de a_j en envoyant au broker a_r un message ACCEPT_PROPOSAL indiquant le résultat de sa décision. Il incrémente aussi la valeur de $NbSoll_k[j]$ (voir Algo. 3 lignes $2-4$) sinon, il rejette la demande d'adhésion et envoie un message REJECT_PROPOSAL au broker (voir Algo. 3 lignes $5-6$).

Sur réception d'un message ACCEPT_PROPOSAL indiquant une réponse positive à la demande d'adhésion, a_r incrémente son compteur de messages $counter_r[z]$ et le nombre de réponses positives (voir Algo. 3 lignes $7-9$).

Sur réception d'un message REJECT_PROPOSAL indiquant une réponse négative à la demande d'adhésion, a_r incrémente uniquement son compteur de messages $counter_r[z]$ (voir Algo. 3 lignes $10-11$).

Le broker attend un message de chaque membre de c_z (voir Algo. 2 ligne 12). Une fois qu'il a reçu tous les messages (un message par membre), il met $counter_r[z]$ à 0 et établit une décision finale reposant sur la règle de la majorité (voir Algo. 2 lignes $13-14$). Dans le cas où la majorité a accepté la demande d'adhésion du candidat a_j, a_r initie une conversation avec l'agent candidat a_j en lui envoyant un message PROPOSE contenant une offre d'adhésion (voir Algo. 2 ligne 15). Sinon, a_r retire a_j de $candP_r[z]$ et met les paramètres de synchronisation $counter_r[z]$ à 0 et $reply_r[z]$ à $True$ pour débloquer son état d'attente (voir Algo. 2 lignes $16-19$).

Sur réception d'un message PROPOSE contenant une offre d'adhésion à une coalition intermédiaire c_z, un agent a_k vérifie si c_z contient un membre présent dans sa liste noire. Si c'est le cas, il rejette l'offre d'adhésion (voir Algo. 4 lignes $2-3$). Sinon, il évalue la fiabilité de c_z en utilisant la mesure décrite dans la définition 8 (voir Algo. 4 ligne 5).

Définition 8 (Confiance en la coalition (evalC)) *Soit c_z une coalition intermédiaire et soit a_k un candidat. La confiance en la coalition est définie comme étant la moyenne arithmétique de la confiance en la coopération de ses membres.*

$$evalC(a_k, c_z) = \frac{\sum_{a_t \in c_z} CC(a_k, a_t)}{|c_z|} \qquad (2)$$

Une fois l'évaluation terminée, l'agent a_k accepte l'offre d'adhésion si sa confiance en la coalition c_z est supérieure à γ_k. Par conséquent, il prend le rôle de membre ($role_k[z] = mem$) et répond au broker avec un message ACCEPT_PROPOSAL indiquant le résultat de la décision (voir Algo. 4 lignes $6-8$). Sinon, il répond au broker avec un message REJECT_PROPOSAL (voir Algo. 4 lignes $9-10$).

En cas de rejet (voir Algo. 4 ligne 28), le broker a_r retire a_k de $candP_r[z]$ et met $counter_r[z]$ à $|c_z|$ (voir Algo. 4 lignes $29-30$) pour débloquer son état d'attente (voir Algo. 2 ligne 22). En cas d'acceptation (voir Algo. 4 ligne 19), le broker a_r envoie à tous les membres de c_z un message INFORM annonçant le succès de la sélection de a_k (voir Algo. 4 lignes $20-21$). Ensuite, il ajoute a_k à c_z, supprime la fonctionnalité satisfaite f_i de $func_r[z]$ et met à jour $candP_r[z]$ en supprimant tous les fournisseurs qui offrent un service avec la fonctionnalité f_i (voir Algo. 4 lignes $22-26$).

Dans les deux cas, a_r met $reply_r[z]$ à $True$ (voir Algo. 4 lignes 27 and 31) pour débloquer son état d'attente (voir Algo. 2 ligne 20).

Sur réception d'un message INFORM contenant une sélection réussie (voir Algo. 4 ligne 32), un membre $a_k \in c_z$ soit a déjà accepté la demande d'adhésion, dans ce cas il incrémente la valeur de $NbMem_k[j]$ et envoie un message CONFIRM à a_r pour exprimer son engagement (voir Algo. 4 lignes $37-39$) soit n'a pas accepté la demande d'adhésion alors, il peut décider de quitter c_z. Cela arrive en cas de mécontentement de a_k concernant la sélection du nouveau membre i.e., sa confiance en la coopération est inférieure à λInf_k.

Le mécontentement d'un membre peut conduire à un groupe malsain et instable. Par conséquent, a_k quitte c_z en prenant le rôle de candidat ($role_k[z] = cand$), ajoute le nouveau membre à sa liste noire et envoie au broker un message FAILURE indiquant le résultat de sa décision (voir Algo. 4 lignes $33-36$). La mise à jour dynamique de la liste noire des agents évite à notre PFC *l'effet ping-pong* dans lequel la coalition reste instable i.e. les agents quittent et rejoignent la coalition indéfiniment [5]. Sur réception d'un message CONFIRM contenant un engagement (voir Algo. 4 ligne 40), le broker a_r incrémente $counter_r[z]$ (voir Algo. 4 ligne 41).

Sur réception d'un message FAILURE d'un agent a_k pour quitter la coalition intermédiairec_z (voir Algo. 4 ligne 11), a_r retire a_k de c_z puis, ajoute les fonctionnalités de ses services éliminés à l'ensemble des fonctionnalités non satisfaites dans c_z (voir Algo. 4 lignes $12-14$).

Enfin, il enrichit l'ensemble des fournisseurs candidats $candP_r$ avec les fournisseurs offrant les fonctionnalités ajoutées de toutes les couches précédentes et incrémente $counter_r[z]$ (voir Algo. 4 lignes $15-19$).

Une fois que a_r a reçu tous les messages, (voir Algo. 2 lignes $21-22$), il débloque son état d'attente et met $counter_r[z]$ à 0. Le processus de sélection des membres d'une coalition c_z se répète et continue jusqu'à ce qu'une des trois conditions de terminaison soit remplie :

- La coalition est complète i.e. toutes les fonctionnalités requises sont instanciées,
- Le timeout $timer_r[z]$ du PFC a expiré,
- La couche maximale de l'arbre est atteinte.

Le résultat de cette phase est un ensemble $\mathcal{C}$ de coalitions qui seront utilisées comme entrée de la phase suivante.

5. La liste noire est une liste technique, type tabou, qui a pour but d'éviter une boucle infinie. Elle est vidée périodiquement.

Algorithm 5: Best Coalition Choice Algorithm

Input: $\mathcal{C} = \{c_z\}$ is the set of coalitions.
Variables: Exp_r est un tableau d'évaluation de confiance en l'expertise, c est la meilleure coalition.

1 **for all** $(c_z \in \mathcal{C})$ **do**
2 **if** $(\exists x \in c_z \mid x \equiv f)$ **then**
3 $\mathcal{C}.remove(c_z)$;
4 **for all** $(c_z \in \mathcal{C})$ **do**
5 $Exp_r[z] \leftarrow \frac{\sum_{a_t \in c_z} EC(a_t)}{|c_z|}$;
6 $c \leftarrow Argmax_{1 \leq z \leq |\mathcal{C}|} Exp[z]$;
7 **return** c ;

3.3 Phase de sélection de la meilleure coalition

Notre approche permet de trouver plusieurs coalitions et de choisir la plus appropriée. Pour ce faire, le broker élimine en premier lieu les coalitions non complètes (voir Algo. 5 lignes $1-3$) et établit un classement en fonction de leur niveau de confiance en l'expertise. La confiance en l'expertise d'une coalition est définie comme étant la moyenne de l'expertise de ses membres (voir Algo. 5 lignes $4-5$). Rappelons que la confiance dans l'expertise d'un membre est un score calculé à partir des valeurs de QoS de(s) service(s) qu'il offre dans la coalition sous-jacente. La coalition qui a la meilleure confiance en l'expertise constitue le service composite qui sera transmis au demandeur.

4 Évaluation de la performance de notre approche de composition de services

Dans cette section, nous présentons les résultats expérimentaux de l'évaluation de la performance de notre approche de composition de services. Les expériences ont été menées sur les données réelles de Facebook [6] contenant 4039 agents et 88234 arêtes. Nous avons créé un bassin de services Web contenant 6 catégories de fonctionnalités, dans chacune d'elles nous avons considéré 5 services différents résultant en 30 services au total. Nous avons équipé chaque agent de trois services différents, choisis au hasard dans le bassin.

Initialement, nous fixons la taille de la blacklist $Blist_k$ de chaque agent a_k à 100 peuplée aléatoirement. Par ailleurs, nous fixons les seuils de confiance comme suit : $\lambda\text{Sup}_k = 0.7$, $\lambda\text{Inf}_k = 0.3$ et $\gamma_k = 0.7$. Finalement, nous initialisons les paramètres $NbSoll_k[A]$ et $NbMem_k[A]$ à des valeurs aléatoires avec $NbSoll_k[A] > NbMem_k[A]$.

Cinq scénarios de test ont été créés comme représenté dans le tableau 1 où la ligne *requête*

6. http ://snap.stanford.edu/data/egonets-Facebook.html

Configuration \ Scénario	A	B	C	D	E
requête utilisateur	2	3	4	5	6
services	5	5	5	5	5
timeout (ms)	3000	3000	3000	3000	3000

TABLE 1 – Définition de scénarios de test

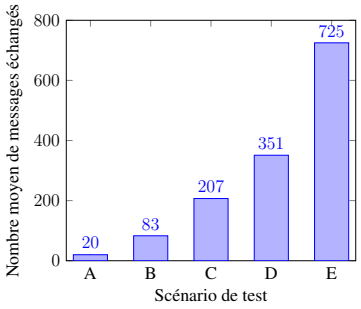


FIGURE 4 – Nombre moyen de messages échangés par scénario de test.

utilisateur désigne le nombre de fonctionnalités requises dans sa requête, la deuxième ligne *service* définit le nombre de services par catégorie de fonctionnalité et la troisième ligne *timeout* spécifie la durée en milliseconde du timeout appliqué. 100 exécutions de formation de coalitions ont été effectuées par scénario de test. Dans la première série d'expériences, nous étudions le coût de communication de notre approche pour les différents scénarios. Dans la deuxième série d'expériences, nous discutons la complétion de notre processus de formation de coalitions pour les différents scénarios. Dans la troisième série d'expériences, nous étudions le comportement des agents durant le PFC en notant la fréquence moyenne des agents qui quittent une coalition.

La figure 4 représente le nombre moyen de messages échangés par scénario de test. Comme prévu, plus le nombre de fonctionnalités requises est élevé, plus le nombre de messages échangés est grand (avec une moyenne de 20 messages dans le scénario A et une moyenne de 725 messages dans le scénario E). Le coût de communication est particulièrement élevé dans le scénario E où plus d'interactions sont nécessaires pour satisfaire les fonctionnalités requises. L'écart du nombre de messages est enregistré dans la phase de sélection des membres du PFC. En effet, comme nous l'avons décrit dans la section précédente, l'adhésion d'un nouveau membre dans une coalition doit garantir la satisfaction en termes de confiance de ce dernier et de la majorité des membres existants également. Ainsi, une grande coalition est intuitivement plus vulnérable à l'instabilité étant donné qu'il y a plus de chances qu'un membre de cette coalition refuse l'adhésion d'un nouveau

membre et quitte par la suite la coalition. Cette intuition sera confirmée plus tard (voir Fig. 6 scénarios D et E). À cause de l'instabilité, le processus requiert plus d'interactions pour compenser la perte des anciens membres et continuer à chercher les membres manquants à la complétion de la coalition.

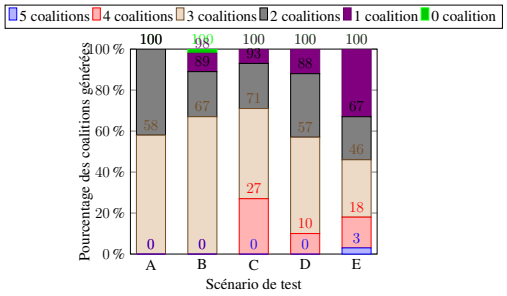


FIGURE 5 – Pourcentage de coalitions générées par scénario de test

L'un des critères clés reflétant la performance d'un processus de composition de services est sa capacité à fournir un service composite complet qui satisfait la requête du demandeur de services. La figure 5 représente le pourcentage du nombre des coalitions générées par scénario de test. Tout d'abord, notons que le nombre maximal de coalitions générées dans les scénarios A et B est de 3 coalitions, dans les scénarios C et D, il est de 4 coalitions et enfin dans le scénario E, il est de 5 coalitions. Bien que le nombre de fonctionnalités requises dans le scénario A soit plus petit que celui du scénario B, les résultats obtenus montrent que le scénario A génère dans 58% des exécutions 3 coalitions alors que le scénario B génère les 3 coalitions dans 67% des exécutions. Cela peut être expliqué par le fait que l'évaluation de la confiance en la coopération se base sur des valeurs générées aléatoirement, ce qui affecte par la suite la prise de décision des agents. Nous notons également que lorsque le nombre de fonctionnalités requises augmente (comme dans les scénarios C, D et E), le pourcentage du nombre maximal de coalitions générées diminue. Par exemple, les exécutions du scénario E donnent 5 coalitions seulement dans 3% du temps. Cela explique notre intuition que plus la taille d'une coalition est grande plus le risque d'insatisfaction chez ses membres existants est élevé, ce qui réduit le nombre de coalitions réussies. Le troisième point à interpréter concerne le cas d'absence de coalitions. La figure 5 montre que le scénario B échoue dans 2% des exécutions à générer une coalition. Cela est dû au fait qu'aucune des 3 coalitions initiales n'a réussi à garantir la confiance requise par les membres et chacun des candidats proposés par le broker.

Les résultats obtenus prouvent que l'approche que nous avons proposée est efficace pour sa capacité à produire de multiples services composites dignes de confiance (pour une configuration de seuils de confiance $\lambda\text{Sup}_k = 0.7$, $\lambda\text{Inf}_k = 0.3$ et $\gamma_k = 0.7$) à faible coût de communication.

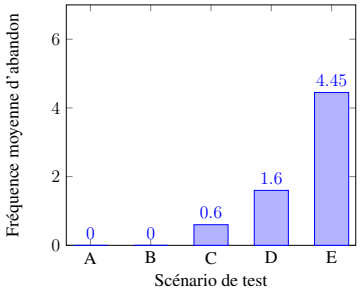


FIGURE 6 – La fréquence moyenne d'abandon par scénario de test

Afin de confirmer empiriquement l'intuition évoquée dans l'expérimentation précédente (la corrélation potentielle entre la stabilité d'une coalition et sa taille), nous allons examiner de plus près comment ces coalitions se forment et plus précisément comment les agents se comportent durant le PFC. La figure 6 illustre la fréquence moyenne d'abandon des agents sur 100 simulations par scénario de test.

La première observation à noter est que les scénarios A et B demeurent stables durant les 100 exécutions et qu'aucun agent ne quitte sa coalition. Pour le scénario A cela n'est pas envisageable car une seule sélection réussie complète la coalition intermédiaire singleton. De la même façon, dans le scénario B l'abandon d'une coalition n'est théoriquement pas possible, étant donné que dans ce cas la majorité correspond aux deux membres de la coalition. Ainsi, l'adhésion d'un nouveau membre n'entraîne pas l'instabilité du groupe.

La deuxième observation à noter concerne l'évolution de la fréquence d'abandon des agents. La figure 6 montre que la fréquence d'abandon des agents est proportionnelle au nombre de fonctionnalités requises. Par exemple, dans les scénario de test C, D et E, en moyenne 0.6, 1.6 et 4.45 agents quittent respectivement leurs coalitions durant le PFC. Cette capacité de quitter une coalition accorde aux agents de l'autonomie pour décider avec qui coopérer assurant ainsi une satisfaction intra-groupe ce qui améliore la qualité du service composite correspondant.

5 Conclusion et perspectives

Dans ce travail, nous avons proposé un processus original de formation de coalitions

fondé sur la confiance pour une composition de services dans les réseaux sociaux. Le processus de formation de coalitions est incrémental, dynamique et recouvrant engageant des agents équipés d'un ensemble de services. Les agents coopèrent en se basant sur un processus de prise de décision décentralisé guidé par la confiance en la coopération. Celle-ci permet aux membres (resp. aux candidats) d'une coalition d'évaluer la fiabilité d'un candidat (resp. d'une coalition) et de décider d'accepter ou pas sa demande d'adhésion (resp. l'offre d'adhésion). L'approche proposée permet la génération de coalitions multiples qui répondent à la requête du demandeur. Elle accorde de l'autonomie aux fournisseurs pour décider avec qui coopérer dans un service composite. Ainsi, ils peuvent quitter leur coalition s'ils sont insatisfaits de l'adhésion d'un nouveau membre.

Les résultats expérimentaux montrent que le modèle offre certains avantages aux agents, mais à un coût important pour établir la stabilité. Les travaux à venir sont nécessaires pour explorer davantage les implications de notre approche sur la stabilité des coalitions et pour réviser et optimiser le modèle en conséquence. Nous devons approfondir plusieurs points tels que la façon de déterminer les valeurs appropriées des seuils utilisés dans la prise de décision ou les contraintes qui conditionnent l'abandon d'une coalition. Une des solutions possibles pour réduire le nombre d'abandons est d'élaborer un mécanisme de négociation engageant le broker et un membre voulant quitter la coalition. Doté de stratégies de persuasions, le broker devrait négocier avec ce membre pour le convaincre de changer d'avis et de rester dans la coalition. Par ailleurs, nous avons l'intention d'étendre le modèle actuel de la confiance en la coopération en incluant d'autres sources d'information telles que la réputation. Finalement, nous prévoyons d'étudier le système dans sa dynamique et surveiller l'évolution du nombre d'abandons au cours du temps. Intuitivement, la fréquence moyenne d'abandon devrait être initialement plus importante (lorsque peu d'agent ont confiance entre eux) puis progressivement diminuer.

Références

[1] R. Aggarwal, K. Verma, J. A. Miller, and W. Milnor. Constraint driven web service composition in meteor-s. In *IEEE SCC*, pages 23–30. IEEE Computer Society, 2004.

[2] B. Benatallah, Q. Z. Sheng, and M. Dumas. The self-serv environment for web services composition. *Internet Computing, IEEE*, pages 40–48, 2003.

[3] J. Bourdon, L. Vercouter, and T. Ishida. A multiagent model for provider-centered trust in composite web services. In *PRIMA*, pages 216–228. Springer, 2009.

[4] F. Casati, M. Sayal, and M.-C. Shan. Developing e-services for composing e-services. In *CAiSE*, pages 171–186. Springer, 2001.

[5] V. Ermolayev, N. Keberle, and S. Plaksin. Towards agent-based rational service composition - racing approach. In *ICWS-Europe*, pages 167–182. Springer, 2003.

[6] N. Griffiths and M. Luck. Coalition formation through motivation and trust. In *AAMAS*, pages 17–24. ACM, 2003.

[7] T. Hongxia, C. Jian, Z. Shensheng, and L. Minglu. A distributed agent coalition algorithm for web service composition. *2014 IEEE World Congress on Services*, pages 62–69, 2009.

[8] M. Klusch and K. P. Sycara. Brokering and matchmaking for coordination of agent societies : A survey. In *Coordination of Internet Agents : Models, Technologies, and Applications*, pages 197–224. Springer-Verlag, 2001.

[9] A. Louati, J. El Haddad, and S. Pinson. A Multi-Agent Approach for Trust-based Service Discovery and Selection in Social Networks Principles and Practices in Multi-Agent Systems. *Scalable Computing : Practice and Experience*, 16(4) :381–402, 2015.

[10] A. Louati, J. El Haddad, and S. Pinson. Trust-based coalition formation for dynamic service composition in social networks. In *Web Information System Engineering–WISE*, pages 570–585, 2015.

[11] D. G Mikulski, F. L Lewis, E. Y Gu, and G. R Hudas. Trust-based coalition formation in multi-agent systems. *The Journal of Defense Modeling and Simulation*, 11(1) :19–32, 2014.

[12] I. Muller, R. Kowalczyk, and P. Braun. Towards agent-based coalition formation for service composition. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*, pages 73–80. IEEE Computer Society, 2006.

[13] L. G . Nardin and J.S. Sichman. Trust-based coalition formation : A multiagent-based simulation. In *Proceedings of the 4th World Congress on Social Simulation*, 2012.

Vers une aide au débuggage des SMA par l'exploitation d'échecs de preuve

Bruno Mermet[a] Gaële Simon[a]
Bruno.Mermet@unicaen.fr Gaele.Simon@unicaen.fr

[a]Laboratoire GREYC - UMR 6072,
Université du Havre, France

Résumé

Cet article se place dans le contexte de la validation de SMA grâce à la preuve de théorèmes. Il présente une étude de cas préliminaire faisant partie d'un travail plus vaste dont l'objectif à long terme est d'analyser dans quelle mesure de tels outils de preuve peuvent être utilisés, pour aider à mettre au point un SMA. L'article décrit comment une erreur liée à un problème de synchronisation entre agents peut être caractérisée grâce à un échec de preuve. L'exploitation des échecs de preuve permet ainsi de mettre en évidence un bug qui ne survient que dans un contexte particulier d'exécution et qui n'aurait donc pas été évident à détecter avec des techniques de debuggage classiques.

Mots-clés : *systèmes multi-agents, échec de preuve, débuggage*

Abstract

This paper is situated in the context of multi-agent systems validation using theorem proving technics. It describes a preliminary case study that is part of a larger work whose main goal is to analyse whether such technics could be used to help the developer to detect and characterise errors in the MAS specification. In the case study presented in the paper, two errors result from a synchronisation problem between agents and it is shown how these errors and the problem itself can be characterised using proof failures. Hence, in the presented case study, it has allowed to highlight a bug that can happen only in specific execution contexts. It would have been rather more difficult to detect it using only classical debugging technics.

Keywords: *Multi-Agent Systems, Proof failure, debugging*

1 Introduction

Dans cet article, on s'intéresse de manière générale à la validation des systèmes multi-agents et plus spécifiquement à la phase de mise au point du système. En effet, depuis plusieurs années maintenant, nous étudions la validation des SMA par la preuve. Pour cela nous avons développé le modèle GDT4MAS [7] qui propose d'une part des outils formels pour spécifier un système multi-agents et d'autre part un système de preuve qui permet, une fois le modèle spécifié, de générer automatiquement les obligations de preuve devant être prouvées pour garantir la correction du système. En parallèle, nous avons commencé une étude visant à répondre à la question suivante : que se passe-t-il si le démonstrateur de théorèmes ne parvient pas à effectuer la preuve ? Plus précisément, est-il possible de tirer des enseignements de cet échec, appelé "échec de preuve", afin d'aider à mettre au point le SMA ? La réponse à cette question dans le cas général est assez délicate. En effet, la première remarque que l'on peut faire est qu'un échec de preuve peut survenir dans trois cas :

— cas 1 : un théorème vrai n'est pas prouvable dans l'absolu (théorème d'incomplétude de Gödel). En effet, les théorèmes à prouver sont exprimés dans la logique du premier ordre auquel est adjointe l'arithmétique. Cela rend le langage utilisé indécidable.

— cas 2 : un théorème vrai n'est pas prouvable automatiquement par le démonstrateur en raison de la semi-indécidabilité de la logique du premier ordre. Cela signifie qu'il se peut qu'un théorème soit prouvable sans pour autant que le démonstrateur soit capable d'en faire la preuve sans intervention de l'utilisateur, par exemple car il nécessite l'utilisation d'une méthode de preuve particulière.

— cas 3 : une erreur dans la spécification du SMA a conduit à la génération d'une obligation de preuve et donc d'un théorème faux qui ne peut donc pas être prouvé.

Une difficulté importante est d'être capable de détecter dans quel cas on se trouve suite à un échec de preuve donné. Pour des raisons de place, ces aspects ne seront pas détaillés dans cet article. Toutefois, on peut signaler que le

système de preuve a été conçu afin de générer des théorèmes sous une forme qui, dans le cas général, permet leur preuve automatique par le prouveur en utilisant des stratégies générales. En effet, d'une part, les contextes logiques où un théorème vrai ne peut être prouvé automatiquement correspondent à peu de cas réels. D'autre part, les cas où le théorème est prouvable mais où l'intervention humaine au niveau du prouveur est nécessaire restent marginaux. Dans la plupart des cas, un échec de preuve sera donc très certainement lié à une erreur dans la spécification et c'est dans ce contexte que l'on se place dans la suite de l'article.

La question à laquelle nous nous intéressons est alors la suivante : si certaines obligations de preuve générées ne sont pas prouvées automatiquement, est-il possible d'en tirer des enseignements afin de corriger la spécification du SMA ? L'idée générale ici est donc d'évaluer la possibilité de tirer partie d'échecs de preuve afin d'aider à détecter, voire à résoudre, des erreurs dans la spécification du SMA. Ainsi, contrairement à ce qui est présenté dans [2], où les auteurs considèrent que les approches de preuve doivent être réservées à la validation du SMA tandis que d'autres approches doivent être utilisées pour les aspects debuggage et mise au point, nous envisageons d'exploiter les échecs de preuve pour détecter très tôt les erreurs de conception d'un SMA et d'aider à sa mise au point.

Dans la suite de l'article, nous commencerons donc par présenter rapidement les différents travaux existant concernant le débuggage de SMA. Dans la partie 3, nous présenterons le modèle GDT4MAS, le principe de preuve, et les outils supports. Le coeur des travaux présentés dans l'article fera l'objet de la partie 4. Enfin, un bilan de l'article et une présentation des travaux à poursuivre feront l'objet de la dernière partie.

2 État de l'art

Garantir la correction d'un SMA est un problème crucial mais très difficile, comme cela a pu être établi à plusieurs reprises [4]. Comme pour les logiciels classiques, il existe principalement deux méthodes pour vérifier la correction d'un SMA : le test et la preuve. Nous invitons le lecteur intéressé à se référer à de précédents articles [7, 8] pour un état de l'art plus complet sur le sujet. Dans cette partie, nous allons nous focaliser sur les travaux portant sur la correction de systèmes erronés.

Test L'essentiel des travaux concernant le débuggage de SMA sont en fait axés sur le test de SMA, afin de détecter un éventuel problème, et de disposer d'un (ou plusieurs) cas de test permettant de reproduire le problème. Les tests peuvent alors ce situer à différents niveaux, comme cela est notamment présenté dans [12] :

— au niveau unitaire : C'est par exemple le but des travaux décrits dans [17]. Il s'agit d'adaptation directe des techniques de test classiques ;

— au niveau agent : De nombreux travaux fort différents existent. Certains proposent des outils "xUnit" pour spécifier des tests unitaires [15], éventuellement en ajoutant au systèmes des agents Mocks [1]. D'autres proposent de rajouter au système des agents dédiés au test [9]. D'autres travaux proposent encore de traiter la multitude de solutions initiales possibles en utilisant des techniques évolutionnistes pour évaluer le système dans des conditions différentes [11].

— au niveau système : Les principes de test au niveau SMA sont encore peu nombreux. Ceci est notamment dû à la difficulté du problème, bien détaillée dans [10], mais il existe toutefois certains travaux allant dans ce sens [15].

Analyse de trace Une autre catégorie de travaux sur le débuggage de SMA concerne l'analyse de traces. Ces travaux sont centrés essentiellement sur trois des tâches liées au débuggage : détecter un problème, déterminer les causes possibles et retrouver la cause originelle [14]. Ils analysent les traces selon deux principes différents : soit en étudiant l'ordonnancement détecté entre des messages échangés entre agents, soit en utilisant des connaissances données par les concepteurs sur le système en utilisant des techniques de fouille de données pour vérifier si ces connaissances se retrouvent dans le système, et pour ainsi détecter ou expliquer d'éventuels bugs. La combinaison de ces deux techniques est d'ailleurs étudiée dans [5].

Visualisation Les outils de visualisation sont peu étudiés dans ce domaine, même s'ils ne sont pas dénués d'intérêt. Mais concevoir de bonnes vues est un problème très difficile à cause de la potentielle multitude d'entités en interaction. Certains travaux existent sur des vues extraites des traces [16], tandis que d'autres font des propositions sur une visualisation "en-ligne" [9].

Échecs de preuve L'exploitation des échecs de preuve reste un domaine prospectif encore peu défriché. Certains travaux ont essayé

d'équiper des prouveurs d'outils susceptibles d'utiliser des échecs de preuve [6]. Quant à l'utilisation de ces échecs, des idées sont proposées dans [3], mais elles n'ont pas été exploitées.

3 Le modèle GDT4MAS

Cette approche, qui intègre un modèle et un système de preuve associé, présente plusieurs caractéristiques intéressantes pour la conception de systèmes multi-agents : un langage formel pour exprimer le comportement des agents et les propriétés attendues, une utilisation de la logique du premier ordre, connue et expressive, et un processus de preuve automatisable. Nous présentons rapidement dans la suite la méthode GDT4MAS, détaillé dans [7, 8].

3.1 Concepts principaux

La méthode GDT4MAS nécessite de spécifier plusieurs notions décrites évoquées ci-dessous.

Environnement L'environnement est spécifié par un ensemble de variables typées et une propriété d'invariance $i_{\mathcal{E}}$.

Types d'agents Chaque type d'agent est spécifié par un ensemble de variables internes typées, un invariant et un comportement. Le comportement d'un agent est principalement défini par un *arbre de décomposition des buts* (GDT). Un GDT est un arbre de buts, dont la racine correspond au but principal de l'agent. Un plan est associé à chaque but : lorsque ce plan est exécuté avec succès il établit le but auquel il est associé. Un plan peut être constitué soit d'une simple action, soit d'un ensemble de buts reliés par un *opérateur de décomposition*. Un but B est principalement décrit par un nom n_B, une condition de satisfaction sc_B et une propriété garantie en cas d'échec gpf_B. La condition de satisfaction (SC) d'un but est spécifiée formellement par une formule qui doit être satisfaite lorsque l'exécution du but réussit. Dans le cas contraire, la propriété garantie en cas d'échec (GPF) d'un but spécifie ce qui est malgré tout garanti lorsque l'exécution d'un plan associé à un but échoue (ceci n'a pas de sens dans le cas d'un but dont le plan réussit toujours, qualifié de but NS). Les SC et les GPF sont des *formules de transition d'état* (STF) car elles expriment la relation entre deux états, appelés état initial et état final (états avant et après la résolution du but). Une STF peut être non déterministe quand, pour un état initial donné, plusieurs états finaux peuvent la satisfaire, comme la STF $x' > x$: si x vaut 0 dans l'état initial, des états finaux avec $x' = 2$ ou $x' = 10$ la satisferont.

Opérateurs de Décomposition Plusieurs opérateurs de décompositions sont définis dans GDT4MAS, permettant d'implanter tout type de comportement. Dans cet article nous n'en utiserons que 2 : l'opérateur `SeqAnd` permet de préciser que l'agent devra d'abord tenter de résoudre le sous-but gauche. En cas de succès, il essaiera alors de résoudre le sous-but droit. Si le comportement de l'agent est correct, alors si les 2 sous-buts sont réussis, le but parent l'est nécessairement. L'autre opérateur que nous utilisons est l'opérateur `SyncSecAnd`. Cet opérateur fonctionne comme l'opérateur `SeqAnd`, mais il permet en plus de mettre des verrous sur certaines variables de l'environnement. Aussi, même si l'agent perd la main, aucun autre agent ne pourra modifier les variables verrouillées.

Actions Les actions sont spécifiées par une précondition qui précise dans quels états elles peuvent être exécutées, et par une postcondition, qui est une STF précisant leur effet.

Agents Les agents sont spécifiés comme des instances de types d'agents, avec des valeurs effectives pour les éventuels paramètres.

3.2 Exemple de GDT

La figure 3 montre le GDT d'un agent composé de trois buts (représentés dans une ellipse par leur nom et leur SC) : le but A est le but racine, décomposé, via un opérateur `SeqAnd` en 2 sous-buts B et C auxquelles des actions sont associées. Aux buts feuilles sont associées des actions représentées par une flèche.

3.3 Principes de preuve

3.3.1 Présentation générale

Le mécanisme de preuve a pour but de prouver les propriétés suivantes :

— Les agents préservent leurs propriétés d'invariance [8] ;

— Les agents préservent les propriétés d'invariance de l'environnement ;

— Le comportement des agents est consistant (i.e. les plans associés aux buts sont corrects) ;

— Les agents établissent leurs propriétés de vivacité, qui formalisent certaines caractéristiques dynamiques attendues de l'agent.

De plus, le mécanisme de preuve repose sur des *obligations de preuve*, qui sont les propriétés à

prouver pour garantir la correction du système. Celles-ci peuvent être générées automatiquement à partir d'une spécification GDT4MAS. Elles sont exprimées en logique du premier ordre et peuvent être vérifiées par tout prouveur adéquat. Enfin, le système de preuve est compositionnel : la preuve de la correction d'un type d'agent est décomposée en plusieurs petites obligations de preuves indépendantes, et la preuve d'un type d'agent peut être réalisée la plupart du temps indépendamment des autres.

3.3.2 Notion de contexte

Une notion clé que nous avons introduite pour le processus de preuve est celle de contexte d'exécution d'un but. Il s'agit d'un prédicat résumant les états dans lesquels un but du GDT d'un agent est susceptible d'être exécuté. Le contexte d'un nœud peut être inféré automatiquement selon une démarche descendante en partant du *triggering context* (TC) de l'agent (prédicat spécifiant quand l'agent est susceptible de s'exécuter).

Nous allons illustrer le principe de calcul du contexte d'un nœud sur le petit exemple de GDT de la figure 1.

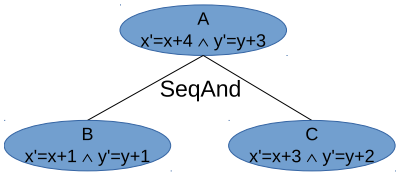


FIGURE 1 – GDT arithmétique

Nous supposons que le TC associé à ce GDT est $x < 10$, que x est une variable de l'environnement, dont l'invariant est $0 \leq x \leq 20$, et que y est une variable interne de l'agent avec comme invariant $y > 0$. Nous savons alors qu'au moment où l'exécution du nœud A va commencer, l'invariant de l'environnement est vrai (puisque c'est un invariant !) et que le TC de l'agent est vérifié. Le contexte de A est donc (sans simplifier) :

$$C_A \equiv x < 10 \wedge 0 \leq x \wedge x \leq 20 \wedge y > 0$$

Comme le plan associé au nœud A consiste à exécuter immédiatement le nœud B, le contexte du nœud B est égal au contexte du nœud A.

Considérons maintenant le nœud C. Le schéma de la figure 2 représente la trace de l'évolution du système.

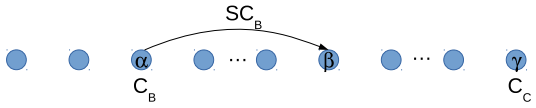


FIGURE 2 – Trace de l'évolution du système

Sur cette trace, nous avons distingué 3 états :

— l'état α est l'état dans lequel le système se trouve lorsque l'agent commence à s'exécuter (et qu'il va donc tenter de résoudre le but A) ;

— l'état β est l'état dans lequel le système se trouve lorsque l'exécution du but B se termine ;

— l'état γ est l'état dans lequel le système se trouve lorsque l'exécution du but C commence.

Comme on peut le voir, les états α et β ne sont pas forcément consécutifs, car le but B peut lui-même être décomposé en plusieurs sous-buts, et d'autres agents peuvent aussi s'exécuter entre temps. De même, les états β et γ ne sont pas consécutifs car d'autres agents peuvent agir au sein du système entre temps. Nous savons cependant que :

— le contexte de B est vérifié dans l'état α ;

— les invariants de l'agent et de l'environnement sont vérifiés dans les états β et γ ;

— la condition de satisfaction du but B est vérifiée entre les états α et β. Autrement dit, la valeur de x en β notée x_β, est égale à celle de x en α, notée x_α augmentée de 1 et la valeur de y en β est égale à celle de y_α augmentée de 1 ;

— la valeur de y en γ est égale à celle de y en β car, y étant une variable interne à l'agent, elle n'a pas pu être modifiée entre les états β et γ (un agent ne peut modifier que ses propres variables ou celle de l'environnement, et l'agent que nous considérons n'a pu effectuer aucune action entre β et γ).

Le contexte de C pourrait donc alors ressembler à la formule suivante :

$$\begin{cases} x_\alpha < 10 \wedge 0 \leq x_\alpha \wedge x_\alpha \leq 20 \wedge y_\alpha > 0 \\ 0 \leq x_\beta \wedge x_\beta \leq 20 \wedge y_\beta > 0 \\ 0 \leq x_\gamma \wedge x_\gamma \leq 20 \wedge y_\gamma > 0 \\ x_\beta = x_\alpha + 1 \wedge y_\beta = y_\alpha + 1 \\ y_\gamma = y_\beta \end{cases}$$

Par la suite, nous appelerons les états α, β et γ respectivement -2, -1 et 0. Le contexte de C sera donc le suivant :

$$C_C \equiv \begin{cases} x_{-2} < 10 \wedge 0 \leq x_{-2} \wedge x_{-2} \leq 20 \wedge y_{-2} > 0 \\ 0 \leq x_{-1} \wedge x_{-1} \leq 20 \wedge y_{-1} > 0 \\ 0 \leq x_0 \wedge x_0 \leq 20 \wedge y_0 > 0 \\ x_{-1} = x_{-2} + 1 \wedge y_{-1} = y_{-2} + 1 \\ y_0 = y_{-1} \end{cases}$$

3.3.3 Schéma de preuve

La méthode GDT4MAS définit également plusieurs *schémas de preuve*. Ces schémas de preuve sont des formules qui permettent de générer des *obligations de preuve*. Par exemple, le schéma de preuve qui permet de vérifier que la décomposition d'un but A en 2 buts B et C via l'opérateur SeqAnd est valide est :

$$C_C \wedge [SC_C]^{0\to 1} \to [SC_A]^{-2\to 1}$$

où $[P]^{a\to b}$ représente le prédicat obtenu en remplaçant dans le prédicat P toutes les occurrences des variables v non indicées par v_a et toutes les occurrences des variables v' par v_b. Ainsi, pour l'exemple de la figure 1, on obtient l'obligation de preuve suivante :

$$C_C \wedge (x_1 = x_0 + 3 \wedge y_1 = y_0 + 2) \\ \to (x_1 = x_{-2} + 4 \wedge y_1 = y_{-2} + 3)$$

La préservation des invariants est également vérifiée grâce à des schémas de preuve à appliquer pour chaque action du GDT.

3.3.4 PVS

PVS (Prototype Verification System) [13] est un environnement de preuve reposant essentiellement sur un démonstrateur de théorèmes permettant de faire des preuves sur des spécifications exprimées dans une logique classique d'ordre supérieur typée. Pour ce faire, le démonstrateur peut s'appuyer sur un ensemble de théories prédéfinies notamment sur la théorie des ensembles et l'arithmétique. Le démonstrateur est interactif c'est-à-dire que l'utilisateur peut intervenir dans le processus de preuve. En ce qui nous concerne nous souhaitons que l'intervention de l'utilisateur soit la plus limitée possible. C'est pourquoi les obligations de preuve sont générées sous une forme permettant, dans la plupart des cas, d'utiliser des stratégies de preuve automatiques. Celle que nous utilisons dans PVS est une stratégie de preuve très générale appelée $grind$ qui utilise la simplification propositionnelle, la simplification arithmétique (en raisonnant notamment sur les égalités), la skolémisation ainsi que la simplification disjonctive.

Le processus de preuve de PVS utilise le calcul des séquents. Ainsi chaque théorème à prouver est transformé en un séquent initial avec un antécédent vide et un conséquent contenant le théorème à prouver. PVS va ainsi construire un arbre de preuve dans lequel chaque séquent va être transformé en un ou plusieurs séquents (qui seront des nœuds fils) en utilisant différentes règles de déduction. La preuve est considérée comme réussie si PVS réussit à construire un arbre de preuve dont les séquents se situant aux feuilles sont tous considérés comme vrais.

3.4 Plateforme d'exécution

Afin de pouvoir plus facilement mener des expérimentations sur le modèle GDT4MAS, nous développons une plateforme permettant d'une part d'exécuter une spécification GDT4MAS et d'autre part de générer automatiquement les obligations de preuve associées dans le langage de spécification de PVS. Cette plateforme est implantée essentiellement en Java. Les spécifications sont exprimées en XML et peuvent être exécutées soit en mode aléatoire (les agents activés sont sélectionnés aléatoirement à chaque pas de temps), soit en mode *trace* si on souhaite tester un entrelacement particulier d'activation des agents. Pendant l'exécution, on peut visualiser graphiquement l'évolution des variables des agents ou d'un sous-ensemble d'entre elles ainsi que l'activation des agents. La plateforme génère également un log textuel permettant de suivre pas à pas l'exécution dans tous ses détails (agent activé, but en cours de résolution, exécution d'actions etc.).

4 Exploiter les échecs de preuve

La problématique abordée par cet article est la suivante. On suppose que l'on part d'une spécification GDT4MAS d'un système multi-agents. En utilisant le système de preuve associé, on peut générer un ensemble d'obligations de preuve permettant, une fois prouvées, de garantir la correction de la spécification de chaque type d'agent impliqué dans le système. La question que l'on se pose peut être formulée de la manière suivante : en supposant que l'on parte d'une spécification GDT4MAS erronée d'un SMA, et constatant la survenue d'un ou plusieurs échecs de preuve au cours du processus de preuve, est-il possible en exploitant, manuellement dans un premier temps, les informations fournies par le prouveur lors des échecs de preuve, d'identifier l'(les) erreur(s) dans la spécification ? Ces erreurs peuvent être très variées. Voici quelques exemples :

— utilisation d'un opérateur de décomposition ne permettant pas la résolution du but parent ;

— spécification d'une SC incomplète par rapport à ce qui est attendu de la résolution de ce but dans la suite du comportement de l'agent ;

— spécification d'un *triggering context* inadéquat ou incomplet ;
— spécification d'un invariant de type d'agent ou d'environnement inadéquat ou incomplet.

À long terme, l'objectif est de distinguer des catégories d'erreurs pour lesquelles les types d'échecs de preuve sont similaires et peuvent être exploités afin d'identifier l'erreur ayant généré ces échecs. C'est dans ce cadre que nous nous sommes intéressés à une étude de cas particulière de type producteur/consommateur dans laquelle nous avons introduit 2 types d'erreurs pour lesquels nous avons analysé les échecs de preuve engendrés afin de déterminer s'il était possible d'identifier les erreurs introduites au départ dans la spécification. Cette étude est présentée en détails dans la partie suivante.

4.1 Etude de cas

4.1.1 Description du SMA utilisé

L'étude de cas qui a été utilisée repose sur un système de type producteur/consommateur. Dans la version de base, le système est donc composé de 2 agents, chacun étant instance d'un type d'agent différent : le type Producteur et le type Consommateur. Cependant, comme nous allons le voir, l'analyse faite dans cette étude d'un point de vue preuve et présentée dans cet article reste la même avec plus de producteurs et/ou de consommateurs. En revanche, comme nous le verrons, il n'en est pas de même en terme d'exécution. L'environnement comporte, lui, une variable appelée $stockE$ de type entier. Cette variable représente le nombre de ressources d'un certain type produites dans l'environnement par le producteur.

Pour produire ces ressources, le producteur utilise des ressources internes d'un autre type modélisées par une variable interne $stockPro$ dont la valeur représente le nombre de ces ressources dont le producteur dispose. Pour produire une ressource dans l'environnement, le producteur consomme une ressource de son stock. Ce processus est représenté par le GDT du type Producteur (voir figure 3) comportant 3 nœuds : le but B modélise la consommation de la ressource interne et le but C représente la production d'une ressource dans l'environnement.

De son coté, le consommateur produit des ressources d'un troisième type. Le nombre de ces ressources est stocké dans une variable interne du type Consommateur appelée $stockCons$. Pour produire ces ressources, le consommateur doit utiliser 2 ressources dans l'environnement (ressources produites par le producteur).

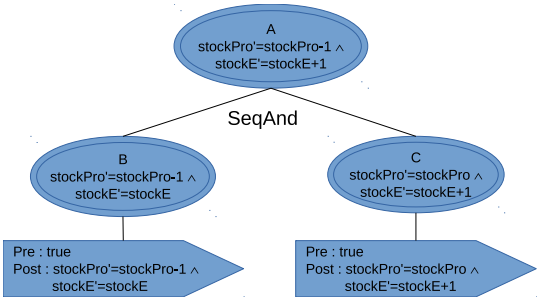


FIGURE 3 – GDT du type Producteur

Ce processus est formalisé par le GDT du type Consommateur (voir figure 4) : le but B modélise la consommation de 2 ressources dans l'environnement et le but C modélise la production d'une nouvelle ressource de type C.

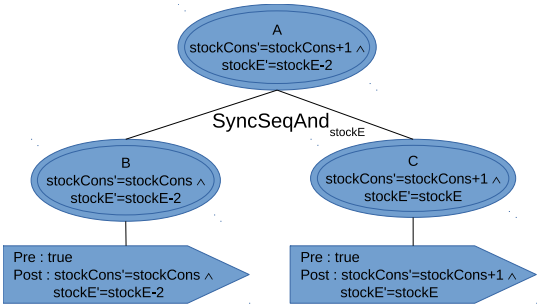


FIGURE 4 – GDT du type Consommateur

En plus du processus précédemment décrit, s'ajoute une contrainte supplémentaire : l'environnement permet de stocker au plus 2 ressources simultanément. Cela signifie que si l'environnement contient déjà 2 ressources alors il faudra qu'au minimum une de ces 2 ressources soit consommée avant que le producteur puisse à nouveau produire une ressource de type E. Cette contrainte est modélisée par le *triggering context* du type Producteur : $stockPro > 0 \wedge stockE < 2$. Avec ce *triggering context*, le GDT du producteur ne peut être activé que si, d'une part, il a encore des ressources internes en réserve et si, d'autre part, il y a moins de 2 ressources dans l'environnement de sorte qu'il pourra stocker la nouvelle ressource produite. De plus l'environnement est doté d'un invariant : $stockE > -1 \wedge stockE < 3$ qui exprime l'intervalle de variation possible de la variable $stockE$ compte tenu de la contrainte énoncée précédemment. Le type d'agent Producteur se voit également doté d'un invariant $stockPro > -1$ qui indique simplement que le nombre de ressources internes du producteur est positif ou

nul. Le *triggering context* du type Consommateur est plus simple : $stockeE > 1$. Autrement dit, le consommateur ne pourra commencer son processus de production, décrit par son GDT, que s'il y a au moins 2 ressources disponibles dans l'environnement. Enfin le type Consommateur est doté d'un invariant $stockCons > -1$ similaire à celui du type Producteur.

4.1.2 Analyse des échecs de preuve

Si l'on génère les obligations de preuve à partir de la spécification décrite précédemment, on obtient une théorie PVS comprenant 18 théorèmes, chaque théorème correspondant à une obligation de preuve devant être prouvée afin que les spécifications des 2 types d'agents soient considérées comme correctes. Il n'est pas possible de détailler ici l'ensemble de ces obligations mais rappelons qu'elles permettent de vérifier :

— que chaque décomposition de but en sous-buts est correcte (la résolution des sous-buts permettra bien de résoudre le but parent),

— pour chaque but feuille que :

  — le contexte d'exécution du but permet bien de satisfaire la pré-condition de l'action associée au but

  — la post-condition de l'action associée au but permet de satisfaire la SC du but.

  — l'invariant de l'environnement et l'invariant du type d'agent auquel appartient le GDT courant sont bien préservés par l'exécution de l'action associée au but feuille

Lorsque l'on essaie de faire la preuve de la théorie ainsi obtenue avec PVS, on obtient un échec de preuve pour 2 des théorèmes de la théorie. Dans ce cas, PVS fournit le dernier séquent de la branche de preuve en cause qu'il ne parvient pas à prouver. Les 2 théorèmes concernés sont :

— `SCtypeProducteurA` : obligation de preuve établissant que la SC du but A du type Producteur est bien satisfaite par l'exécution de la décomposition de ce but.

— `PostInvtypeProducteurC` : obligation de preuve permettant de vérifier que l'invariant de l'environnement et l'invariant du type Producteur sont bien préservés par l'exécution du but C.

Avant d'aller plus loin, voici les notations de variables utilisées dans la spécification PVS, correspondant aux différentes valeurs d'une même variable dans des états différents comme cela a été présenté dans la partie 3.3.2 :

— v_{-2} (v_2) : état de v avant l'exécution de B

— v_{-1} (v_1) : état de v après l'exécution de B

— v_0 (v0) : état de v avant l'exécution de C

— v_1 (v1) : état de v après l'exécution de C

Cela signifie par exemple que dans le théorème permettant d'établir que la SC de A sera vérifiée après l'exécution de la décomposition du but A, $stockE'$ dans la SC de A est représentée par $stockE1$ et $stockE$ est représentée par $stockE\_2$. Il faut souligner que dans le modèle d'exécution retenu, l'état du système avant l'exécution de A est considéré comme étant le même que celui avant l'exécution de B.

Premier échec de preuve Voici maintenant le théorème `SCtypeProducteurA` tel qu'il a été généré par la plateforme :

```
SCtypeProducteurA: THEOREM
((true)&(stockPro_2 > 0)&(stockE_2 < 2)&
(stockPro_2 > -1)&(stockE_2 > -1)&(stockE_2 < 3)&
(stockPro_1 = stockPro_2 - 1)&(stockE_1 = stockE_2)&
(stockPro_1 = stockPro0)&(stockPro_1 > -1)&
(stockE_1 > -1)&(stockE_1 < 3)&(stockPro0 > -1)&
(stockE0 > -1)&(stockE0 < 3)&(stockE1 = stockE0 + 1)&
(stockPro1 = stockPro0)&(stockPro1 > -1)&
(stockE1 > -1)&(stockE1 < 3))
=>
(stockPro1 = stockPro_2 - 1)&(stockE1 = stockE_2 + 1)
```

On peut remarquer que la partie droite de l'implication correspond bien à la SC de A projetée sur les variables `stockE1` (correspondant à `stockE'`), `stockE_2` (correspondant à `stockE`), `stockPro1` (correspondant à `stockPro`) et `stockPro_2` (correspondant à `stockPro`).

Voici comparativement le séquent produit par PVS suite à l'échec de preuve de ce théorème :

```
{-1}  (stockPro_2 > 0)
{-2}  (stockE_2 < 2)
{-3}  (stockPro_2 > -1)
{-4}  (stockE_2 < 3)
{-5}  (stockPro_1 = stockPro0)
{-6}  (stockE_1 = stockE_2)
{-7}  (stockPro_2 - 1 = stockPro0)
{-8}  (stockE_2 > -1)
{-9}  (stockE0 > -1)
{-10} (stockE0 < 3)
{-11} (stockE1 = 1 + stockE0)
{-12} (stockPro1 = stockPro0)
{-13} (stockPro0 > -1)
{-14} (1 + stockE0 > -1)
{-15} (1 + stockE0 < 3)
  |-------
{1}   stockE0 = stockE_2
```

La première chose dont on peut s'apercevoir est que le prédicat sur la variable $stockPro$ qui apparaissait dans la partie droite de l'implication du théorème initial n'apparait plus dans le conséquent du séquent. Cela signifie que cette partie a pu être prouvée et que le problème concerne donc uniquement la partie de la SC portant sur la variable $stockE$.

Le prouveur ne parvient donc pas à prouver $stockE0 = stockE\_2$ avec les hypothèses fournies dans la partie gauche du théorème.

Or dans les hypothèses des théorèmes générés par notre système de preuve, on ne peut avoir que des relations entre des variables dans des états consécutifs ou dans le même état (dans les invariants par exemple). Cela signifie que si la preuve du prédicat précédent ne peut pas être faite, c'est qu'il manque dans les hypothèses soit $stockE\_2 = stockE\_1$ soit $stockE\_1 = stockE0$, soit les deux. Or on observe que le second prédicat se trouve dans les antécédents du séquent ($\{-6\}$). Il manque donc uniquement le prédicat $stockE\_1 = stockE0$ exprimant la préservation de la valeur de la variable $stockE$ entre l'exécution de B et C. Or cette préservation ne peut être obtenue qu'en utilisant un opérateur synchronisé (ici `SyncSeqAnd`) qui va permettre de bloquer la variable $stockE$ tant que l'agent producteur n'a pas commencé à exécuter le but C. En effet, la sémantique du modèle ne fait aucune hypothèse sur ce qui se passe dans cette période pour les variables d'environnement car, dès que l'agent producteur a terminé d'exécuter le but B, il peut perdre la main et n'importe quel agent peut donc s'exécuter et modifier la variable $stockE$ avant que l'agent producteur ne commence à exécuter le but C. Dans le GDT du type Producteur il faut donc remplacer le `SeqAnd` par un `SyncSeqAnd` verrouillant la variable $stockE$.

Si l'on prend un peu plus de recul sur la sémantique de cet échec de preuve, on peut se demander pourquoi pour prouver la satisfaction de la SC de A après l'exécution de sa décomposition, il est indispensable que la valeur de la variable $stockE$ soit préservée entre l'exécution des buts B et C par le producteur. Supposons que ce ne soit pas le cas et qu'avant l'exécution du but A, $stockE$ vaille 2. Si le consommateur est activé entre l'exécution de B et C par le producteur, celui-ci va diminuer la valeur de $stockE$ de 2 (la mettant à 0) pour effectuer sa propre production. Puis le producteur va exécuter le but C qui augmentera la valeur $stockE$ de 1 (la mettant donc à 1). Le producteur ayant terminé d'exécuter la décomposition de son but A, sa SC devrait donc être satisfaite, ce qui n'est pas le cas ($stockE = 2$, $stockE' = 1$, donc on n'a pas $stockE' = stockE + 1$). C'est cette erreur que révèle l'échec de preuve. En revanche, si on utilise un opérateur synchronisé permettant de bloquer la variable $stockE$ entre l'exécution des buts B et C par le producteur, le consommateur ne pourra pas, comme dans l'exemple précédent, modifier la variable $stockE$ car elle sera bloquée et il devra attendre qu'elle ne le soit plus avant de pouvoir la modifier.

Second échec de preuve Voici maintenant le théorème `PostinvtypeProducteurC` tel qu'il a été généré par la plateforme :

```
((true)&(stockPro_2 > 0)&(stockE_2 < 2)&
(stockPro_2 > -1)&(stockE_2 > -1)&(stockE_2 < 3)&
(stockPro_1 = stockPro_2 - 1)&(stockE_1 = stockE_2)&
(stockPro_1 = stockPro0)&(stockPro_1 > -1)&
(stockE_1 > -1)&(stockE_1 < 3)&(stockPro0 > -1)&
(stockE0 > -1)&(stockE0 < 3)&(stockE1 = stockE0 + 1)&
(stockPro1 = stockPro0))
=>
(stockPro1 > -1)&((stockE1 > -1)&(stockE1 < 3))
```

Ce théorème a pour but de montrer que l'invariant de l'agent est bien préservé après l'exécution du but C d'où la formule $stockPro1 > -1$ dans la partie droite de l'implication. Il doit aussi permettre de montrer que l'invariant de l'environnement est bien préservé après l'exécution de C d'où la formule $(stockE1 > -1) \wedge (stockE1 < 3)$ à droite de l'implication.

Voici le séquent obtenu suite à l'échec de preuve du théorème précédent par PVS :

```
{-1}  (stockPro_2 > 0)
{-2}  (stockE_2 < 2)
{-3}  (stockPro_2 > -1)
{-4}  (stockE_2 < 3)
{-5}  (stockPro_1 = stockPro0)
{-6}  (stockE_1 = stockE_2)
{-7}  (stockPro_2 - 1 = stockPro0)
{-8}  (stockPro0 > -1)
{-9}  (stockE_2 > -1)
{-10} (stockE0 > -1)
{-11} (stockE0 < 3)
{-12} (stockE1 = 1 + stockE0)
{-13} (stockPro1 = stockPro0)
  |-------
{1}   (1 + stockE0 < 3)
```

Par comparaison avec le théorème initial, on s'aperçoit que le prouveur ne parvient pas à faire la preuve d'une partie de l'invariant de l'environnement à savoir le fait que la valeur de $stockE$ doit toujours être inférieure à 3. En effet $1 + stockE0 < 3$ correspond à la dernière formule de la partie droite de l'implication du théorème dans laquelle $stockE1$ a été remplacé par $1 + stockE0$ en utilisant l'antécédent $\{-12\}$). Ce prédicat ne peut pas être prouvé car la seule hypothèse dont on dispose sur $stockE0$ est $stockE0 < 3\{-11\}$. Il n'est donc pas possible de montrer que $1 + stockE0 < 3$ soit $stockE0 < 2$. Si on analyse les antécédents du séquent, on peut s'apercevoir que les hypothèses $\{-2\}$ et $\{-6\}$ permettent de déduire que $stockE\_1 < 2$. Pour

montrer $stockE0 < 2$, il manque donc l'hypothèse $stockE\_1 = stockE0$. C'est exactement la même hypothèse qui manquait déjà pour résoudre le premier échec de preuve. Cela signifie donc que cet échec de preuve est aussi lié à la non utilisation d'un opérateur synchronisé entre B et C permettant de bloquer la variable $stockE$.

Cet échec de preuve a donc permis de détecter la même erreur de conception que le premier échec de preuve. Il ne correspond en revanche pas au même bug. En effet, le fait que l'on ne puisse pas prouver $stockE0 < 2$ montre que l'on n'a pas la garantie que la variable $stockE$ avant l'exécution de C sera toujours inférieure à 2. Cela signifie donc qu'il existe forcément des situations où après l'exécution de C la variable $stockE$ sera supérieure à 2 ce qui est interdit par l'invariant de l'environnement. Ceci correspond donc à un nouveau bug possible dans l'exécution du système. Ce qui est particulièrement intéressant avec cet échec de preuve, c'est que le problème détecté ne peut pas se produire tant que l'on a un seul producteur dans le système. En revanche, si l'on considère qu'il y a plusieurs producteurs dans le SMA, le problème peut survenir. Supposons qu'il existe 2 producteurs p1 et p2, que la variable $stockPro$ de chaque producteur est initialisée à 5, que la variable $stockE$ est initialisée à 0, que la variable $stockCons$ du consommateur est initialisée à 0 et que l'exécution du SMA se fait avec un entrelacement contenant la séquence suivante :

```
...
p1 (stockPro de p1 passe à 4)
p1 (stockE passe à 1)
p1 (stockPro de p1 passe à 3)
p2 (stockPro de p2 passe à 4)
p2 (stockE passe à 2)
p1 (stockE passe à 3)
...
```

La trace d'exécution ci-dessus conduit bien au bug mis en évidence par l'échec de preuve précédent. Cette trace a été exécutée par notre plateforme et on a pu effectivement constater, grâce à la visualisation de l'évolution des différentes variables du système (figure 5), la confirmation de cet état fait puisque la variable $stockE$ passe bien par la valeur 3 pendant l'exécution.

Notons que si l'on utilise cette fois un opérateur synchronisé (`SyncSeqAnd`) dans le GDT du type Producteur comme le suggèrent les 2 échecs de preuve étudiés, dans la trace précédente, lorsque p2 essaie de modifier la variable $stockE$, il ne pourra pas le faire car la variable sera bloquée puisqu'à ce moment p1 est en attente de pouvoir exécuter son but C.

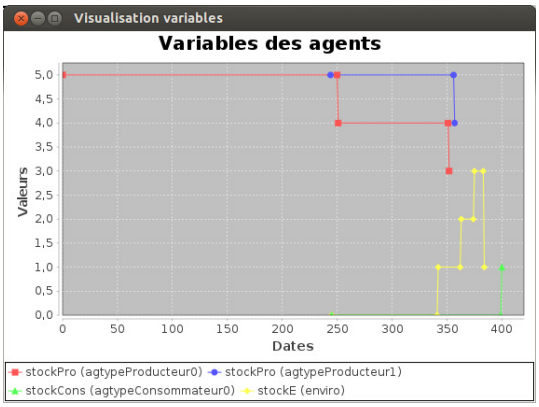


FIGURE 5 – Évolution des variables du SMA

Bilan de l'analyse des échecs de preuve À l'issue de l'analyse de ces premiers échecs de preuve, on peut tout d'abord remarquer que la structure même des obligations de preuve générées par notre système de preuve permet de savoir rapidement à quelle partie du comportement du système se rattachent les problèmes détectés puisqu'il suffit pour cela de regarder le nom du théorème dont la preuve a échoué. Ensuite, on a pu montrer que les 2 échecs de preuve étudiés permettent de détecter efficacement un problème de conception dans la spécification du type Producteur. Le premier échec de preuve a permis d'isoler l'hypothèse manquante pour faire la preuve, ce qui a permis de déterminer comment résoudre le problème : en utilisant un opérateur synchronisé permettant de bloquer la variable $stockE$ par le producteur pendant l'exécution de la décomposition du but A. Ce problème est en fait lié à un problème de synchronisation entre les différents agents (producteur/consommateur dans le premier cas, producteur/producteur dans le deuxième cas), caractérisé grâce aux échecs de preuve.

De plus, chacun de ces échecs a permis de mettre en évidence deux bugs différents (non vérification de la SC de A dans certains cas, non respect du bornage de $stockE$ dans d'autres cas) engendrés par cette même erreur de conception qui n'auraient pas été forcément faciles à déceler sans cela. Notamment, le bug lié au deuxième échec de preuve n'apparaît que si l'on a au moins 2 producteurs et que si l'on a un entrelacement particulier à l'exécution. Il est donc particulièrement peu évident à exhiber et de nombreuses exécutions seraient nécessaires pour le mettre en évidence. L'échec de preuve associé permet donc de gagner un temps précieux dans l'identification de ce bug.

Précisons que si l'on modifie la spécification du type d'agents Producteur en utilisant l'opérateur `SyncseqAnd` au lieu de `SeqAnd` pour décomposer le but A, la preuve se fait cette fois complètement, sans échec de preuve.

Cette petite étude de cas montre donc que l'on peut exploiter des échecs de preuve afin à la fois d'expliquer des comportements anormaux à l'exécution (bugs associés à chaque échec de preuve) et de déterminer ce qui les cause (l'absence de blocage de la variable d'environnement pendant l'exécution du producteur).

5 Conclusion et perspectives

Dans cet article, nous avons présenté une première étude prometteuse quant à l'exploitation des échecs de preuve pour l'aide à la mise au point de SMA. Nous avons notamment montré qu'une telle technique permet de mettre directement en évidence des bugs qui n'apparaitront que dans certaines exécutions, en fonction de l'entrelacement entre les actions des agents, et correspondant donc à des cas difficiles à mettre en évidence et à reproduire en utilisant des techniques de débuggage classiques. Bien sûr, d'autres types d'échecs de preuve seront à considérer pour valider de façon plus générale la technique. Nous comptons aussi réfléchir à l'exploitation semi-automatique des échecs de preuve, des motifs caractéristiques semblant se dessiner. À terme, nous espérons pouvoir établir une taxonomie des échecs de preuve pouvant se produire, en identifiant les causes potentielles et les moyens de correction adaptés.

Références

[1] O. Coelho, U. Kulesza, A. von Staa, and C. Lucena. Unit testing in multi-agent systems using mock agents and aspects.

[2] M. Dastani and J.-J. Ch Meyer. *Specification and Verification of Multi-agent Systems*, chapter Correctness of Multi-Agent Programs : A Hybrid Approach. Springer, 2010.

[3] L. A. Dennis and P. Nogueira. What can be learned from failed proofs of non-theorems. Technical report, Oxford University Computer Laboratory, 2005.

[4] A. Drogoul, N. Ferrand, and J.-P. Müller. Emergence : l'articulation du local au global. *ARAGO*, 29 :105–135, 2004.

[5] K. Suzanne Barber Dung N. Lam. Automated Interpretation of Agent Behaviour. In *AOIS*, pages 1–15, 2005.

[6] M. Kaufmann and J.S. Moore. Proof Search Debugging Tools in ACL2. http ://www.cs.utexas.edu/users/moore/-publications/acl2-papers.html, 2008.

[7] B. Mermet and G. Simon. GDT4MAS : an extension of the GDT model to specify and to verify MultiAgent Systems. In Decker *et al.*, editor, *Proc. of AAMAS 2009*, pages 505–512, 2009.

[8] B. Mermet and G. Simon. A new proof system to verify gdt agents. In *IDC*, pages 181–187, 2013.

[9] D. Meron and B. Mermet. A Tool Architecture to Verify Properties of Multiagent System at Runtime. In *PROMAS*, pages 201–216, 2006.

[10] S. Miles, M. Winikoff, S. Cranefield, C.D. Nguyen, A. Perini, P. Tonella, M. Harman, and M. Luck. Why testing autonomous agents is hard and what can be done about it. AOSE Technical Forum 2010 Working Paper.

[11] C. D. Nguyen, S. Miles, A. Perini, P. Tonella, M. Harman, and M. Luck. Evolutionary testing of autonomous software agents. *JAAMAS*, 25(2) :260–283, 2012.

[12] C.D. Nguyen, A. Perini, C. Bernon, J. Pavón, and J. Thangarajah. Testing in Multi-Agent Systems. In *AOSE*, pages 180–190, 2009.

[13] S. Owre, N. Shankar, and J. Rushby. Pvs : A prototype verification system. In *CADE 11*, 1992.

[14] E. Serrano, J.J. Gómez-Sanz, J.A. Botía, and J. Pavón. ntelligent data analysis applied to debug complex software systems. *Neurocomputing*, 72(13-15) :2785–2795, 2009.

[15] A.M. Tiryaki, S. Öztuna, O. Dikenelli, and R.C. Erdur. SUNIT : A Unit Testing Framework for Test Driven Development of Multi-Agent Systems. In *Agent Oriented Software Engineering (AOSE)*, pages 156–173, 2006.

[16] G. Vigueras and J.A. Botía. Tracking Causality by Visualization of Multi-Agent Interactions Using Causality Graphs. In *PROMAS*, pages 190–204, 2007.

[17] Zhiyong Zhang, John Thangarajah, and Lin Padgham. Model based testing for agent systems. In *AAMAS'09*, pages 1333–1334, 2009.

Affectation distribuée d'individus à des activités avec des préférences additivement séparables

Maxime Morge
maxime.morge@univ-lille1.fr

Antoine Nongaillard
antoine.nongaillard@univ-lille1.fr

Univ. Lille, CNRS, Centrale Lille, UMR 9189 - CRIStAL - Centre de Recherche en Informatique Signal et Automatique de Lille, F-59000 Lille, France

Résumé

Nous souhaitons proposer un réseau social numérique afin que les utilisateurs forment des groupes pour pratiquer ensemble des activités. Dans cet article, nous introduisons un modèle formel de formation de coalitions correspondant à ce cas d'usage. Nous nous restreignons à des préférences additivement séparables pour proposer un algorithme distribué. Nous démontrons que le résultat est Pareto-optimal. Nos expérimentations montre que la solution atteinte par notre algorithme est meilleure que celle obtenue via les techniques classiques de recherche locale et que sa distribution permet d'accélérer son exécution.

Mots-clés : *Résolution distribuée de problème, Négociation, Comportement d'agents, Problème d'appariement*

Abstract

We aim at providing a social network such that users form groups to practice together some activities. In this paper, we introduce a formal framework for coalition formation which is suitable for our usecase. We restrict ourselves to additively separable preferences in order to propose a distributed matching algorithm. We demonstrate that its outcome is a Pareto-optimum. Our experiments shows we reach a better outcome than the classical local search techniques and that the distribution of our algorithm speeds up its runtime.

Keywords: *Distributed problem solving, Negotiation, Agent behavior, Matching problem*

1 Introduction

Les Systèmes Multi-Agents (SMA) constituent un paradigme de premier ordre pour l'analyse, la conception et l'implémentation de systèmes composés d'entités autonomes en interaction. Afin de concevoir des environnements socio-techniques médiateurs ou simulés, les SMA permettent de modéliser des boucles de rétroaction entre acteurs hétérogènes dont les prises de décision locales font émerger des phénomènes globaux. Un des défis auquel la communauté SMA fait façe consiste à faciliter l'élicitation des préférences des utilisateurs.

Nos travaux s'inscrivent dans le projet « Partir Ensemble » (PartENS). Ce projet vise à comprendre et modéliser les dynamiques de rétroaction se produisant dans un collectif en interaction à la fois au sein d'un réseau social virtuel et d'un réseau social réel. Le cas d'usage sur lequel nous nous concentrons ici concerne un collectif de séniors âgés de 60 à 70 ans qui entrent en relation afin d'effectuer des activités ponctuelles. Ce terrain d'expérimentation implique plusieurs milliers d'individus inscrits au CCAS (Caisse Centrale d'Activités Sociales du Personnel) de la ville de Lille. L'objectif est ici de maximiser les activités organisées afin d'améliorer la cohésion sociale et lutter contre l'isolement chez certains séniors. Nous souhaitons proposer un réseau social numérique afin que les utilisateurs forment des groupes pour pratiquer ensemble des activités. Ce système a pour objectif de suggérer à chaque utilisateur des individus avec lesquels pratiquer ces activités.

Dans cet article, nous introduisons un modèle formel de formation de coalitions correspondant à notre cas d'usage. Un ensemble d'individus doivent être appariés pour pratiquer une des activités qui leur ait proposé en fonction des préférences des individus pour leurs alter egos et pour les activités. Nous nous restreignons à des préférences additivement séparables afin de proposer un algorithme d'affectation. Nous démontrons que le résultat de notre algorithme est valide et Pareto-optimal. En adoptant le modèle d'acteur [2], nous sommes en mesure de distribuer cet algorithme. Nos expérimentations montre que la solution atteinte par notre algorithme est de meilleur qualité que celles obtenues avec des techniques classiques de recherche locale. De plus, sa distribution permet d'accélérer son exécution (jusqu'à 3,5 fois).

La section 2 compare notre approche aux tra-

vaux scientifiques connexes. Nous introduisons notre problème d'affectation dans la section 3. Nous proposons un algorithme d'affectation dans la section 4. La distribution de cet algorithme est présentée sous la forme de comportements d'agents dans la section 5. La section 6 exhibe nos résultats expérimentaux. Finalement, nous synthétisons nos travaux puis nous dressons quelques perspectives (cf Section 7).

2 Travaux connexes

La théorie du choix social vise à construire et à analyser des processus pour la décision collective où un ensemble d'agents sélectionnent ou classent conjointement un sous-ensemble d'alternatives parmi celles disponibles. Contrairement à l'Économie, l'Informatique se préoccupe des questions algorithmiques afin de rendre opérationnelles ces méthodes. Nous nous focalisons ici sur un problème d'appariement en particulier.

Dans le problème de formation de coalition hédonique introduit par [4], chaque joueur est caractérisé par une relation de préférences sur l'ensemble des coalitions auxquelles il peut appartenir. Notre problème est une spécialisation de ce problème générale. Nous pouvons le représenter comme un jeu hédonique en associant à chaque activité un joueur qui préfère ne pas être surchargé et en munissant chacun des autres joueurs des préférences agrègeant celle d'un individu sur les activités et sur les alter egos. Nous considérons ici que ces deux préférences sont indépendantes, puis nous les agrégeons dans une fonction d'utilité. Sous cette hypothèse, notre configuration a des propriétés structurelles utiles qui se distinguent de celle d'un jeu hédonique générique. Le fait que les activités soient des points focaux [11] et la représentation succincte des préférences nous permet de proposer un algorithmique pour atteindre de "bon" appariements.

Le problème de sélection d'activité de groupe a été proposé dans [3]. Dans un tel problème, chaque agent participe à au plus une activité et ses préférences portent uniquement sur les activités et elles dépendent du nombre de participants à l'activité. C'est une généralisation d'un jeu hédonique anonyme. Même si ce problème a été étendu dans [7] pour prendre en compte les relations entre individus, ces relations sont encodées par un réseau social, i.e. un graphe non orienté où les nœuds correspondent aux individus et les arêtes représentent des liens de communication entre eux. À l'inverse, les individus sont munis, dans notre problème, d'une relation de préférence sur leur alter egos.

Le problème des hôpitaux-internes a été introduit dans [6]. Ce problème peut être considéré comme une spécialisation du problème de formation de coalition hédonique où un ensemble de résidents doivent être affectés à des hôpitaux étant donnés les préférences des résidents vis-à-vis des hôpitaux et les préférences des hôpitaux pour les résidents. Le problème HR a fait l'objet de nombreuses extensions [8]. À notre connaissance, aucune n'est adaptée à notre cas d'usage.

Comment un agent évalue ses penchants vis-à-vis des alternatives ? Nous optons ici pour des fonctions d'utilité, i.e. des préférences cardinales. Par souci de simplification, nous supposons que l'évaluation des activités et celle des groupes sont comparables. Bien que leur expressivité soit limitée, nos préférences sont linéaires par rapport au nombre d'individus et d'activités.

Quelle est la « meilleure » solution à un problème de choix collectif ? Dans la littérature, on trouve principalement deux types de règles qui dérivent le choix collectif de la satisfaction individuelle des agents : les premières se basent sur les propriétés désirables de la solution (e.g. la stabilité) alors que les secondes se basent sur l'agrégation de la satisfaction individuelle des agents (e.g. le bien-être utilitaire). Dans cet article, nous adoptons la seconde approche car aucun concept de la première approche ne convient.

Comment atteindre un appariement maximisant le bien-être utilitaire ? Comme affirmé dans cet article, ce problème peut être NP-difficile. C'est la raison pour laquelle, nous pouvons considéré les algorithmes de Problème d'Optimisation sous Contraintes Distribué (DCOP) ou les méthodes de recherche locale (LST). Il a été montré dans [5] que les algorithmes de DCOP ne passent pas nécessairement à l'échelle pour les problèmes d'appariement. Nous montrons ici que les techniques de LST ne conviennent pas car la fonction à optimiser possède de nombreux optima locaux. C'est la raison pour laquelle nous avons adopté une méthode de résolution multi-agents et notamment une modélisation multi-niveaux comme préconisé par [9] où chaque agent « activité » représente un groupe d'individus.

3 Problème IA

Notre objectif ici est de proposer un modèle formel de formation de coalitions adapté à notre cas d'usage. Nous introduisons ici le problème des individus/activités (IA). Dans une instance de problème IA, les individus privilégient les activités qui leur plaisent avec les partenaires qu'ils apprécient.

Définition 1 (Problème IA)**.** *Un **problème individus/activités (IA)** de taille (m, n), avec $m \geq 1$ et $n \geq 1$, est un couple $IA = \langle I, A \rangle$ avec m individus et n activités, où :*

- *$A = \{a_1, \ldots, a_n\}$ est un ensemble de n activités. Chaque activité a_j peut accueillir au plus c_j membres ;*
- *$I = \{1, \ldots, m\}$ est un ensemble de m individus. À chaque individu i correspond,*
  1. *une préférence sur les activités, i.e. une relation réflexive, complète et transitive $\trianglerighteq_i$ sur les activités $A \cup \{\theta\}$, notamment l'activité nulle (notée θ). La relation de préférence stricte correspondante est notée $\triangleright_i$,*
  2. *une préférence purement hédonique, i.e. une relation de préférence réflexive, transitive et complète $\succsim_i$ sur l'ensemble des groupes auxquels il appartient $G(i) = \{G \subseteq I; i \in G\}$. La relation de préférence stricte correspondante est dénotée $\succ_i$.*

L'activité nulle correspond à ne rien faire. Les préférences sur les groupes sont des préférences purement hédoniques car la satisfaction d'un individu dépend uniquement des autres membres de son groupe et non pas des autres groupes constitués par les individus restants.

Nous souhaitons former des coalitions autour des activités.

Définition 2 (Coalition)**.** *Soit $IA = \langle I, A \rangle$ un problème IA. Une **coalition** est un couple $C = \langle a, G \rangle$ où $a \in A \cup \{\theta\}$ et $G \subseteq I$. L'activité de la coalition C, éventuellement l'activité nulle, est a_C avec une capacité [1] c_C et le groupe G_C. Une coalition non-vide C est telle que $G_C \neq \emptyset$ et C est pour i si $i \in G_C$.*

Nous espérons que le nombre d'individus est considérablement plus important que le nombre d'activités ($m >> n$).

1. l'activité nulle a une capacité infinie.

Définition 3 (Appariement)**.** *Un **appariement** M pour le problème $IA = \langle I, A \rangle$ est représenté par les fonctions $a_M : I \to A \cup \{\theta\}$ et $g_M : I \to \mathcal{P}(I)$ telles que :*

$$\forall i \in I,\ a_M(i) \in A \cup \{\theta\} \quad (1)$$

$$\forall i \in I,\ i \in g_M(i) \subseteq I \quad (2)$$

$$\forall i \in I,\ a_M(i) = \theta \Rightarrow g_M(i) = \{i\} \quad (3)$$

$$\forall i \in I\, \forall j \in g_M(i),\ a_M(j) = a_M(i) \quad (4)$$

$$\forall i, j \in I,\ i \neq j \wedge a_M(i) = a_M(j) \neq \theta \Rightarrow g_M(i) = g_M(j) \quad (5)$$

L'affectation d'un individu est une activité, éventuellement l'activité nulle (cf équation 1). Chaque individu est associée à un groupe auquel il appartient (cf équation 2). Tous les individus qui sont affectés à l'activité nulle sont seuls (cf équation 3). Tous les individus associés les uns aux autres ont la même activité (cf équation 4) et réciproquement tous les individus qui sont affectés à la même activité, à l'exception de l'activité nulle, sont associés les uns aux autres (cf équation 5).

Pour simplifier les notations, nous introduisons la fonction de poste d'un appariement M qui retourne l'ensemble des individus affectés à chaque activité :

$$p_M : A \cup \{\theta\} \to \mathcal{P}(i)$$
$$p_M(a) = \{i \in I; a_M(i) = a\} \quad (6)$$

Les postes d'une activité peuvent être vide. Si $a_M(i) = \theta$, on dit que i n'est pas affecté. Une activité $a \in A$ est : i) surchargée si $|p_M(a)| > c_a$; ii) pleine si $|p_M(a)| = c_a$; iii) sous-chargée sinon. Un appariement est dit **valide** si aucune activité n'est surchargée.

Un appariement est une structure de coalition.

Proposition 1 (Partition)**.** *Soit M un appariement pour le problème $IA = \langle I, A \rangle$.*

$$\forall a \in A \cup \{\theta\},\ \langle a, p_M(a) \rangle \text{ est une coalition} \quad (7)$$

$$\bigcup_{a \in A \cup \{\theta\}} p_M(a) = I \wedge \quad (8)$$
$$\forall a_i, a_j \in A \cup \{\theta\}\ p_M(a_i) \cap p_M(a_j) = \emptyset$$

Preuve 1 (Partition)**.** *Des définitions 2 et 3.*

$C_M(i)$ est la coalition dans M qui contient i.

Chaque individu évalue les coalitions et donc les appariements en fonction du groupe auquel il appartient et de son activité.

Définition 4 (Rationalité)**.** *Soit $IA = \langle I, A\rangle$ un problème IA.*

— *Une coalition C pour i est **individuellement rationelle** pour i ssi :*

$$(a_C \trianglerighteq_i \theta) \wedge G_C \succsim_i \{i\} \quad (9)$$

— *Un appariement M est **individuellement rationel** ssi :*

$$\forall i \in I,\ (a_M(i) \trianglerighteq_i \theta) \wedge g_M(i) \succsim_i \{i\} \quad (10)$$

— *Soient C et C' deux coalitions qui sont individuellement rationelles pour i.*

— *L'individu i **préfère** C à C' (dénoté $C \succsim_i C'$) ssi :*

$$a_C \trianglerighteq_i a_{C'} \wedge G_C \succsim_i G_{C'} \quad (11)$$

— *L'individu i **préfère strictement** C à C' (noté $C \succ_i C'$) ssi :*

$$C \succsim_i C' \wedge (a_C \triangleright_i a_{C'} \vee G_C \succ_i G_{C'}) \quad (12)$$

— *Soient M et M' deux appariements valides pour IA. L'individu i **préfère** M à M' (noté $M \succsim_i M$) ssi :*

$$C_M(i) \succsim_i C_{M'}(i) \quad (13)$$

La relation de préférence stricte sur les appariements est notée $\succ_i$. Un individu choisit une coalition et donc un appariement tel que son activité est préférée à l'activité nulle et il préfère ses partenaires plutôt que de rester seul (cf équations 9 et 10). Il préfère une coalition à une autre s'il préfère l'activité et le groupe de la première (cf équation 11). Un individu préfère un appariement à un autre s'il préfère sa coalition dans le premier (cf équation 13). La relation de préférence sur les appariements valides est réflexive, transitive et éventuellement partielle.

Une première propriété désirable pour l'évaluation d'un appariement est la stabilité du cœur.

Définition 5 (Stabilité du cœur)**.** *Soit M un appariement pour le problème $IA = \langle I, A\rangle$. Une coalition non-vide C **bloque** l'appariement M ssi :*

1. *l'activité n'est pas surchargée : $|G_C| \leq c_C$*
2. *tous les individus de la coalition préfère celle-ci à leur affectation dans M :*

$$\forall i \in G_C,\ C \succsim_i C_M(i) \quad (14)$$

3. *au moins un individu de la coalition préfère strictement celle-ci à son affectation selon M :*

$$\exists i \in G_C\ C \succ_i C_M(i) \quad (15)$$

*L'appariement M est **stable de cœur** s'il est valide et qu'il n'existe pas de coalition bloquante :*

$$\forall C \subseteq (A \cup \{\theta\}) \times (2^I \setminus \emptyset) \quad (16)$$
ce n'est pas le cas que C bloque M

Les individus dans une coalition bloquante souhaite se désolidariser et former leur propre coalition, ce qui rend le partitionnement instable.

Un appariement est Nash stable si aucun individu n'a intérêt à dévier unilatéralement de sa coalition vers une autre (éventuellement vide).

Définition 6 (Nash stabilité)**.** *Soit M un appariement pour le problème $IA = \langle I, A\rangle$. L'appariement M est **Nash stable** s'il est valide, rationel et :*

$$\forall i \in I\, \forall a \in A,\ a \neq a_M(i) => \\ p_M(a) = c_a \vee C_M(i) \succsim_i \langle a, p_M(a) \cup \{i\}\rangle \quad (17)$$

Un appariement Nash stable est à l'abri des mouvements individuels car les activités sont pleines ou la coalition de chaque individu est au moins aussi bonne que les autres.

Comme l'illustre l'exemple suivant, même si la rationalité individuelle d'un appariement est une condition nécessaire pour être soit stable de cœur soit Nash stable, aucune de ces stabilités n'implique l'autre. De plus, un problème IA n'a pas nécessairement d'appariement stable de cœur ou Nash stable.

Exemple 1 (Stabilité)**.** *Considéron le problème IA avec 3 individus (1, 2 et 3) et une activité a telle que $a \trianglerighteq_i \theta$ avec $i \in \{1, 2, 3\}$.*
Nous supposons que la capacité de a est 2 et que les préférences sociales sont circulaires :

— $\{1, 2\} \succ_1 \{1\} \succ_1 \{1, 3\}$;
— $\{2, 3\} \succ_2 \{2\} \succ_2 \{1, 2\}$;
— $\{1, 3\} \succ_3 \{3\} \succ_3 \{2, 3\}$.

Cette instance n'a pas d'appariement stable de cœur car l'appariement M_1 (avec $p_{M_1}(a) = \{1, 2\}$) est bloqué par la coalition $\langle a, \{2, 3\}\rangle$, l'appariement M_2 (avec $p_{M_2}(a) = \{2, 3\}$) est bloqué par $\langle a, \{1, 3\}\rangle$ et M_3 (avec $p_{M_3}(a) = \{1, 3\}$) est bloqué par $\langle a, \{1, 2\}\rangle$. De plus, il n'y a pas d'appariement Nash stable. En particulier, M_1, M_2 et M_3 ne sont pas individuellement rationnels.
Supposons maintenant que la capacité de a est 3 et que l'individu 3 est "indésirable", i.e. les coalitions avec 3 ne sont pas individuellement rationnelles pour les autres :

— $\{1, 2\} \succ_1 \{1\} \succ_1 \{1, 2, 3\} \succ_1 \{1, 3\}$;
— $\{1, 2\} \succ_2 \{2\} \succ_2 \{1, 2, 3\} \succ_2 \{2, 3\}$;

— $\{1,2,3\} \succ_3 \{2,3\} \succ_3 \{1,3\} \succ_3 \{3\}$;
L'appariement M_1 tel que $p_{M_1}(a) = \{1,2\}$ est stable de cœur mais n'est pas Nash stable. L'appariement M_2 tel que $p_{M_2}(a) = \{3\}$ est Nash stable mais n'est pas stable de cœur.

Une autre propriété désirable pour évalue un appariement est la Pareto-optimalité.

Définition 7 (Pareto-optimal)**.** *Soient M et M' deux appariements valides pour le problème $IA = \langle I, A\rangle$. M'* ***domine au sens de Pareto*** *M ssi :*

$$\forall i \in I, C_{M'}(i) \succsim_i C_M(i) \tag{18}$$
$$\exists i \in I, C_{M'}(i) \succ_i C_M(i) \tag{19}$$

Un appariement est ***Pareto-optimal*** *s'il n'est pas dominé au sens de Pareto.*

Un appariement est Pareto-optimal s'il n'existe pas d'alternative pour laquelle tous les agents sont dans une position meilleure ou équivalente.

La stabilité de cœur est une condition suffisante mais pas nécessaire pour la Pareto-optimalité.

Proposition 2 (Pareto-optimum)**.** *Tous les appariements stable de cœur sont Pareto-optimaux.*

Preuve 2 (Pareto-optimum)**.** *Nous prouvons par contradiction qu'un appariement stable de cœur est Pareto-optimal. Nous supposons que M est un appariement stable de cœur qui n'est pas Pareto-optimal. Donc, il existe un appariement M' qui domine M au sens de Pareto. Considérons la coalition $C_{M'}(i)$ avec i qui satisfait l'équation 19. On vérifie :*

1. *M' est valide ;*
2. *l'équation 14 par l'équation 18 ;*
3. *l'équation 15 par l'équation 19.*

En conséquence, par la définition 5, on conclut que $C_{M'}(i)$ bloque M qui est en contradiction avec notre hypothèse.

Dans notre exemple précédent, quand la capacité de a est 2 et que les préférences sociales sont circulaires, tous les appariements où l'activité est pleine sont Pareto-optimaux. Quand la capacité est 3 et que l'individu 3 est indésirable, l'appariement où l'activité est pleine est également Pareto-optimal.

Même si la stabilité est une propriété souhaitable, il n'existe pas nécessairement une telle solution. En revanche, la Pareto-optimalité semble ne pas être discriminante. Une autre façon d'évaluer la qualité d'un appariement réside dans la notion de bien-être social. Pour cela, nous supposons que les individus ont des préférences cardinales. De plus, dans un problème IA, chaque individu évalue ses préférences vis-à-vis des 2^{m-1} groupes. La représentation de telles préférences est exponentielle en espace. En revanche, la représentation de préférences additivement séparables est linéaire par rapport au nombre d'individus.

Définition 8 (Additivement Séparable IA)**.** *Soit $IA = \langle I, A\rangle$ un problème IA de taille (m, n). Le problème est* ***additivement séparable*** *(ASIA) si chaque individu $i \in I$ est muni :*

1. *d'une fonction de valuation $v_i : A \cup \{\theta\} \to [-1; 1]$ représentant ses préférences sur les activités, éventuellement nulle ;*
2. *d'une fonction de valuation $w_i : I \setminus \{i\} \to [-1; 1]$ représentant ses préférences sur les partenaires potentiels.*

La fonction d'utilité pour un individu i est la fonction $u_i : G(i) \times A \cup \{\theta\} \to [-1; 1]$ définie telle que :

$$\forall g \in G(i)\ \forall a \in A \cup \{\theta\} \tag{20}$$
$$u_i(g, a) = \frac{[\frac{1}{m-1}\Sigma_{j\in g, j\neq i} w_i(j)] + v_i(a)}{2} \tag{21}$$

Nous supposons que les préférences sur les individus et les préférences sur les activités sont comparables. En particulier, l'utilité pour un individu d'être seul dépend uniquement de son évaluation de l'activité. En outre, nous supposons que plus l'individu a de partenaires, plus il est satisfait car nous visons à améliorer la cohésion sociale et éviter l'isolement.

Dans la suite, nous ne considérons pas que les utilités sont définies pour correspondre aux préférences comme dans [1] mais nous supposons un accès direct aux utilités des individus car cela sera le cas dans notre application pratique.

Nous adoptons ici l'approche utilitaire inspirée par Bentham. En d'autres termes, notre objectif est de maximiser la somme des utilités individuelles.

Définition 9 (Bien-être)**.** *Soit $IA = \langle I, A\rangle$ un problème ASIA de taille (m, n). Le* ***bien-être utilitaire*** *d'un appariements M est défini tel que :*

$$U_I(M) = \frac{1}{m}\sum_{i\in I} u_i(g_M(i), a_M(i)) \tag{22}$$

Plus le bien-être est important, meilleur est l'appariement.

Exemple 2 (problème ASIA)**.** *Considérons l'exemple précédent où la capacité de l'activité est* 3 *et l'individu* 3 *est indésirable. On définit les fonctions de valuations telles que* $v_1(a) = v_2(a) = v_3(a) = 0$*,* $w_1(2) = w_2(1) = \frac{1}{2}$*,* $w_1(3) = w_2(3) = -1$*,* $w_3(1) = \frac{1}{2}$ *et* $w_3(2) = 1$*. En conséquence, les préférences sur les groupes sont conformes à la formulation précédente. L'appariement où les individus* 1 *et* 2 *sont affectés à l'activité maximise le bien-être.*

4 Résolution

Afin de maximiser le bien-être utilitaire, nous pouvons considérer la programmation quadratique, i.e. une méthode d'optimisation avec un modèle mathématique représentable par une fonction quadratique. Une instance de problème ASIA peut être modélisé par $n \times m$ variables $x_{ia} \in \{0, 1\}$ telles que $x_{ia} = 1$ si l'individu i est affecté à l'activité a et $x_{ia} = 0$ sinon, m contraintes $\Sigma_{a \in A} x_{ia} \leq 1$ représentant l'exclusion mutuelle de l'affectation d'un même individu aux activités et n contraintes $\Sigma_{i \in I} x_{ia} \leq c_a$ garantissant la validité de l'appariement. En considérant les fonctions de valuations $(w_i(j))_{i \in I}$ et $(v_i(a))_{i \in JI}$, la fonction objectif correspondant au bien-être utilitaire, qui est à maximiser, est :

$$\sum_{a \in A} \sum_{i \in I} x_{ia}(v_i(a) + \frac{1}{m-1} \sum_{j \neq i} w_i(j) \times x_{ja})$$

Quand on écrit sous une forme standard ce problème, i.e. en minimisant une fonction objectif de la forme $\frac{1}{2} x Q x^T + c^T x$ avec une matrice symétrique Q, on peut remarquer que cette dernière n'est pas nécessairement définie positive et donc le problème peut être NP-difficile [10].

C'est la raison pour laquelle nous proposons ici un algorithme (cf algorithme 1) qui permet de calculer un "bon" appariement. Initialement, tous les individus sont seuls, affectés à l'activité nulle et libres. Tour à tour, chaque individu libre i considère l'activité qui l'attire et qu'il préfère a. Si aucun autre individu n'est affecté à cette activité, i est affecté. Sinon, l'algorithme tente d'améliorer le bien-être de ce groupe, éventuellement en renvoyant les individus dont la présence contribue le moins au bien-être du groupe. Si l'affectation de i n'améliore pas le bien-être du groupe, alors i doit concéder et donc considérer l'activité attractive suivante. Les individus qui sont remplacés par i doivent concéder. Si la capacité de a n'est pas atteinte alors le groupe peut croître (ligne 21). Un agent qui est rejeté par toutes les activités qu'il considère comme attrayantes reste seul et il est définitevement inactif. Un algorithme d'approximation consiste à exclure qu'un seul individu à chaque étape (ligne 20).

On peut remarquer que notre algorithme (éventuellement d'approximation) retourne toujours un appariement valide.

Proposition 3 (Terminaison)**.** *Notre algorithme (d'approximation) appliqué à* $ASIA$ *se termine et l'appariement retourné est valide.*

Preuve 3 (Terminaison)**.** *Soit* $IA = \langle I, A \rangle$ *un problème ASIA. Nous considérons l'invariant de boucle* $\Sigma_{i \in I} |concessions(i)| + |Free|$*. Cet invariant est positif ou nul. Il décroît strictement à l'issue de chacune des boucles car :*

1. *un individu affecté est retiré de* $Free$*;*
2. *un individu qui n'est pas affecté concède jusqu'à être éventuellement affecté à l'activité nulle ;*
3. *un individu, qui est désaffecté, concèdent et au moins un autre individu est affecté (donc retiré de* $Free$*) ;*

L'appariement obtenu est valide car les activités ne sont jamais surchargées.

Le resultat de notre algorithme exact est Pareto-optimal.

Proposition 4 (Pareto-optimalité)**.** *Notre algorithme appliqué à un problème* $ASIA$ *retourne un appariement Pareto-optimal.*

Preuve 4 (Pareto-optimalité)**.** *Soit* $IA = \langle I, A \rangle$ *un problème* $ASIA$*. Nous montrons par contradiction que le résultat de notre algorithme est un appariement Pareto-optimal* M*. Nous supposons que* M *est dominé au sens de Pareto par un appariement valide* M'*. Par l'équation 19, il existe un individu* i *tel que* $C_{M'}(i) \succ_i C_M(i)$*. Par l'équation 12,* $C_{M'}(i) \succsim_i C_M(i)$ *et :*

- *soit* $g_{M'}(i) \succ_i g_M(i)$ *et* $a_{M'}(i) = a_M(i)$*. Comme* $C_{M'}(i) \succsim_j C_M(i)$ *et par l'équation 11,* $g_{M'}(i) \succsim_j g_M(i)$ *pour tous les partenaires de* i*. Par l'équation 20,* $\Sigma_{j \in g_{M'}(i)} u_j(g_{M'}(i), a_{M'}(i)) > \Sigma_{j \in g_M(i)} u_j(g_M(i), a_M(i))$*. Ceci est une contradiction avec notre algorithme (ligne 27) qui calcule les postes qui maximisent le bien-être du groupe ;*
- *soit* $a_{M'}(i) \rhd_i a_M(i)$ *et donc* $a_{M'}(i)$ *précède* $a_M(i)$ *dans* $concessions(i)$*. Selon l'algorithme,* i *a été rejeté ou désaffecté*

Algorithme 1 : Calcul d'un appariement pour un problème ASIA

. **Entrées** : $IA = \langle I, A \rangle$
Sorties : un appariement M
$Free = I$;
pour chaque $i \in I$ **faire**
  $concessions(i) = A.SortWith(v_i(\_) > v_i(\_) > 0)$;
  $a_M(i) = \theta$;
  $g_M(i) = \{i\}$;
tant que $Free \neq \emptyset$ **faire**
  pour chaque $i \in Free$ **faire**
    si $concessions(i) = \emptyset$ **alors** $Free \setminus = \{i\}$;
    sinon
      $a = concessions(i).head$; `//a est l'activité préférée`
      $g = p_M(a)$;
      $g' = g \cup \{i\}$;
      si $g = \emptyset$ **alors**
        `/* les postes de` a `sont vides et donc` i `et affecté */`
        $a_M(i) = a$;
        $g_M(i) = \{i\}$;
        $Free \setminus = \{i\}$;
      sinon
        $u_{max} = -\infty$;
        $bg = \emptyset$;
        $SG = \{sg \subsetneq g'; sg \neq \emptyset\}$;
        `/* éventuellement` $SG = \{sg \subsetneq g'; |sg| = |g'| - 1\}$ `*/`
        si $c_a > |g|$ **alors** $SG \cup = g'$;
        `/*` g `peut croître */`
        pour chaque $sg \in SG$ **faire**
          $u = \Sigma_{k \in sg} u_k(sg, a)$;
          si $u > u_{max}$ **alors**
            $u_{max} = u$;
            $bg = sg$;
        `/*` bg `est le meilleur groupe */`
        pour chaque $j \in bg$ **faire** $g_M(j) = bg$;
        pour chaque $j \in g \setminus bg$ **faire**
          `/*` j `est désaffecté */`
          $a_M(j) = \theta$;
          $g_M(j) = \{j\}$;
          $Free \cup = \{j\}$;
          $concessions(j) = concessions(j).tail$;
        si $i \in bg$ **alors**
          `/*` i `est affecté */`
          $a_M(i) = a$;
          $g_M(i) = bg$;
          $Free \setminus = \{i\}$;
        else
          `/*` i `est rejeté */`
          $concessions(i) = concessions(i).tail$;
retourner M

par $a_{M'}(i)$. En conséquence, il existe un individu $j \in C_{M'}(i)$ tel que $C_M(j) \succ_j C_{M'}(j)$. Ceci est une contradiction avec l'équation 18.

Exemple 3 (Algorithme)**.** *Considérons l'exemple 2. Notre algorithme opère les affectations/désaffectations suivantes sur la liste d'individus libres* $(3, 2, 1)$ *:*

1. 3 *est affecté car les postes de* a *sont initialement vide ;*
2. 2 *est affecfé car la capacité de* 3 *n'est pas atteinte et* $\{2, 3\}$ *est le meilleur sous-groupe de* $\{2, 3\}$ *;*
3. 1 *remplace* 3 *car* $\{1, 3\}$ *est le meilleur sous-groupe de* $\{1, 2, 3\}$ *;*
4. 3 *concède et il est définitevement affecté à l'activité vide.*

L'appariement obtenu est Pareto-optimal et maximise le bien-être utilitaire.

On peut noter que dans le cas général notre algorithme ne maximise pas nécessairement le bien-être utilitaire.

5 Comportement d'agents

Nous considérons ici le modèle de programmation concurrente par passage de messages asynchrones, appelé modèle d'acteur, proposé dans [2]. Selon cette perspective, les primitives sont les agents et les événements. Un agent représente un programme indépendant qui s'exécute sur son propre processeur. Un événement consiste en la création d'un agent ou l'émission/la réception d'un message. On peut noter qu'un tel système est distribué car le délai de transmission des messages est arbitraire mais non négligeable. On suppose que les canaux de communication sous-jacents sont fiables (un message est délivré une et une seule fois) et que les messages peuvent arriver dans un ordre différent de celui de l'émission.

Afin de proposer un solveur distribué à partir de ce modèle, nous distinguons trois types d'agents :

1. l'agent « solveur » qui crée les autres agents et enregistre les affectations ;
2. les agents « individu » tous munis du même comportement mais avec des préférences qui les distinguent ;
3. les agents « activité » tous munis du même comportement mais qui gèrent des coalitions et des capacités différentes.

Le comportement de l'agent « solveur » consiste à : i) créer les agents ; ii) lancer la résolution ; iii) enregistrer les affectations et les désaffectations ; iv) renvoyer l'appariement quand tous les individus sont affectés.

Le comportement de l'agent « individu » consiste à créer sa liste de concessions puis à se proposer à l'activité préférée qui l'attire. Quand l'agent est affecté ou désaffecté, il en informe l'agent « solveur ». Lors d'une désaffectation, l'agent « activité » attend une confirmation avant d'affecter un nouvel individu afin que l'appariement ne soit pas retourné prématurément par l'agent « solveur ». Si un agent « individu » devient libre, il concède, i.e. il se propose à l'activité suivante qui l'attire jusqu'à être affecté à l'activité nulle.

Le comportement de l'agent « activité » est décrit par l'automate déterministe représenté dans la figure 1. Quand une proposition est reçue, elle est acceptée si le groupe courant est vide (dans l'état `Available`). Sinon, l'agent traîte les propositions une à une (dans l'état `Raising`) en identifiant le sous-groupe qui maximise le bien-être utilitaire. Si la capacité est atteinte, les sous-groupes de taille c_a sont considérés. Dans le cas contraire, le nombre de postes peut croître. Si le proposant n'est pas dans le nouveau groupe, alors la proposition est rejetée. Sinon la proposition est accepté quand le membre écarté a confirmé la désaffectation (dans l'état `Firing`). Quand une proposition a été traitée, l'agent « activité » est prêt à évaluer les propositions suivantes, éventuellement celles stockées.

6 Évaluation empirique

Nos expérimentations ont pour objectifs d'évaluer la qualité de la solution atteinte par notre algorithme et l'accélération due à sa distribution.

Nous avons implémenté notre prototype avec le langage de programmation Scala [2] et la boîte à outils Akka [3]. Cette dernière, en s'appuyant sur le modèle d'acteur [2], nous permet de réduire la distance entre les spécifications du SMA et son implémentation. Afin d'envisager un grand nombre d'individus, nous considérons notre algorithme d'approximation.

Nous avons implémenté un algorithme de recherche locale pour le comparer à notre algorithme. Cet algorithme de montée en gradient qui, à partir d'un appariement valide aléatoire cherche à améliorer le bien-être utilitaire, est itératif. Deux appariements sont voisins s'ils sont identiques à l'exception d'un individu affecté à l'activité nulle ou ayant changé d'activité. Si la nouvelle activité est pleine, alors tous les échanges d'individus sont envisagés.

Nous considérons des problèmes $ASIA$ constitués de n activités et m individus. Pour une même instance de problème, toutes les activités ont le même capacité (m/n). Pour chaque jeu de paramètres (n et m), nous avons généré (pseudo)-aléatoirement 100 instances.

Dans un premier temps, nous avons comparé le bien-être utilitaire de l'appariement obtenu par notre algorithme avec celui obtenu par recherche locale. La figure 2 présente les bien-utilitaires moyens obtenus pour chaque jeu de paramètres (avec $2 \leq n \leq 10$ et $2 \times n \leq m \leq 10 \times n$). Le bien-être obtenu par notre algorithme surpasse sans conteste celui la recherche locale. En effet, le bien-être utilitaire est pour le problème $ASIA$ une fonction qui possède de nombreux optima locaux.

Dans un second temps, nous avons comparé les temps d'exécution des versions distribuée et centralisée de notre algorithme d'approximation. La figure 2 présente les temps d'exécution moyen obtenus pour chaque jeu de paramètres (avec $2 \leq n \leq 10$ et $2 \times n \leq m \leq 400$). Alors que l'algorithme centralisé est plus rapide quand le nombre d'individus est faible (~ 40), son temps d'exécution croît rapidement avec le nombre d'individus (18 ms pour 100 individus et 10 activités) alors que le temps d'exécution de la version distribuée est moindre (9 ms dans ce dernier cas). De plus, le temps d'exécution de l'algorithme de montée en gradient est très important (1660 ms pour 100 individus et 10 activités). On peut s'attendre à un temps d'exécution encore plus important si on adopte une méthode de recherche locale du type recuit simulé sans pour autant garantir d'obtenir un optimum global.

En résumé, la solution atteinte par notre algorithme semble de bonne qualité. De plus, ce dernier est distribuable ce qui permet d'accélérer (jusqu'à 3,5 fois) son exécution.

7 Conclusions

Nous avons introduit ici le problème générique individus/activités où des individus doivent être affectés à des activités en privilégiant les activités qui leur plaisent avec les partenaires qu'ils apprécient. Bien que souhaitable, la stabilité d'une solution n'est pas garantie. À l'inverse, la

2. http ://www.scala-lang.org/
3. http ://akka.io

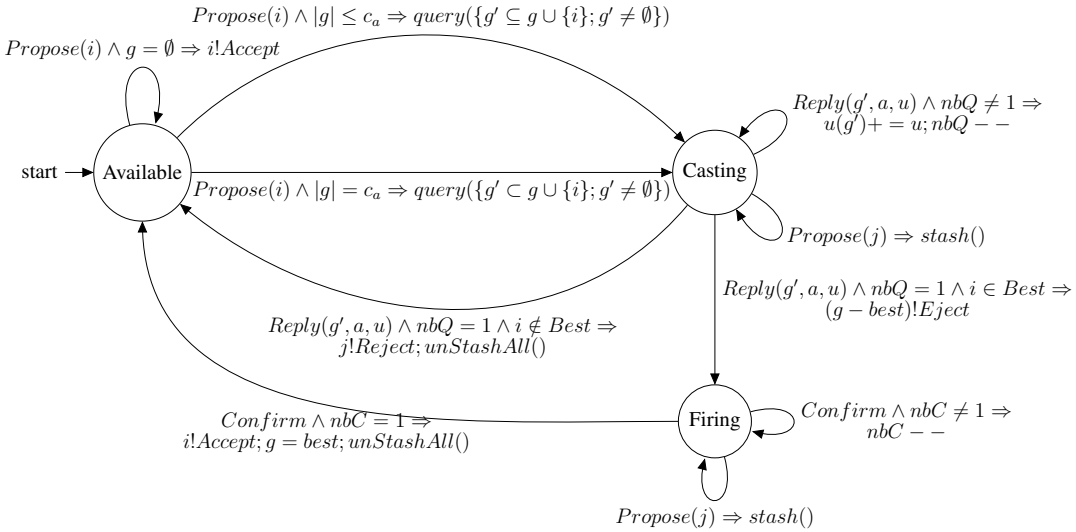


FIGURE 1 – Comportement d'un agent « activité »

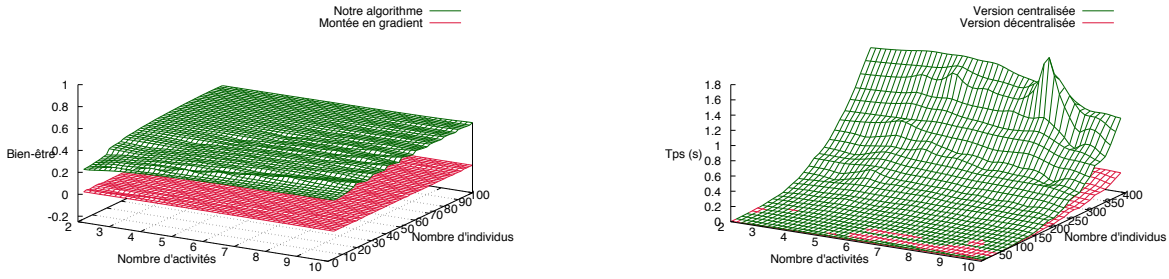


FIGURE 2 – Bien-être utilitaire (à gauche) et temps d'exécution (à droite)

Pareto-optimalité n'est pas discriminante. C'est la raison pour laquelle nous avons adopté des préférences additivement séparables afin d'évaluer la qualité d'une solution à l'aide du bien-être utilitaire des individus. De plus, la représentation des préférences additivement séparable est linéaire par rapport au nombre d'individus et d'activités. Maximiser le bien-être utilitaire peut être NP-difficile. C'est la raison pour laquelle nous avons proposé un heuristique où les individus se proposent aux activités qu'ils préfèrent quitte à concéder. Les individus qui contribuent le moins au bien-être du groupe son désaffectés et concèdent. Nous avons montré que cet algorithme renvoie systématiquement un appariement valide et Pareto-optimal. En adoptant le modèle d'acteur, nous avons distribué cet algorithme. La difficulté réside dans : i) la détection de l'arrêt (comme les messages peuvent arriver dans un ordre différent de l'envoi, un agent « activité » doit attendre la confirmation d'une désaffection avant d'affecter un nouvel individu) ; ii) la synchronisation (comme l'acceptation d'un individu dépend du groupe, un agent « activité » doit traiter les propositions les unes après les autres). Nos expérimentations montrent que le bien-être utilitaire obtenu par notre algorithme surpasse celui atteint par recherche locale et que la distribution permet d'accélérer notre algorithme (jusqu'à 3,5 fois).

Dans de futurs travaux, nous souhaitons évaluer notre moteur d'appariement avec des données réelles dans le cadre du projet PartENS. Afin de proposer des appariements qui soient socialement plus juste, nous envisageons de modifier le critère local de décision de notre algorithme pour maximiser le bien-être égalitaire.

Remerciements

Ce travail s'inscrit dans le projet PartENS soutenu par le programme chercheur-citoyen de la région Nord Pas de Calais. Nous remercions le comité de programme des JFSMA qui, par ses remarques, nous a permis d'améliorer cet article.

Références

[1] Boutilier, C., Caragiannis, I., Haber, S., Lu, T., Procaccia, A. D., and Sheffet, O. (2015). Optimal social choice functions : A utilitarian view. *Artificial Intelligence*, 227 :190–213.

[2] Clinger, W. D. (1981). *Foundations of actor semantics*. PhD thesis, Massachusetts Institute of Technology.

[3] Darmann, A., Elkind, E., Kurz, S., Lang, J., Schauer, J., and Woeginger, G. (2012). Group activity selection problem. In *Proc. of the 8th International Conference on Internet and Network Economics*, pages 156–169, Liverpool, UK. Springer Berlin Heidelberg.

[4] Dreze, J. and Greenberg, J. (1980). Hedonic coalitions : Optimality and stability. *Econometrica*, 48 :987–1003.

[5] Everaere, P., Morge, M., and Picard, G. (2012). Casanova : un comportement d'agent respectant la privacité pour des mariages stables et équitables. *RIA*, 26(5) :471–494.

[6] Gale, D. and Shapley, L. S. (1962). College admissions and the stability of marriage. *The American Mathematical Monthly*, 69 :9–14.

[7] Igarashi, A., Peters, D., and Elkind, E. (2017). Group activity selection on social networks. In *Proc. of AAAI*, pages 565–571.

[8] Manlove, D. F. (2014). *Algorithmics of Matching Under Preferences*. World Scientific.

[9] Nongaillard, A. and Picault, S. (2016). Modélisation multi-niveaux des problèmes d'affectation et d'appariement. In *JFSMA'16*, pages 75–84, Rouen, France. Cépaduès.

[10] Sahni, S. (1974). Computationally related problems. *SIAM Journal on Computing*, 3(4) :262–279.

[11] Schelling, T. C. (1980). *The strategy of conflict*. Harvard university press.

Négociation « one-to-many » adaptative pour améliorer l'acceptabilité des services d'un fournisseur SaaS

A. Najjar[a,b] amro.najjar@emse.fr
O. Boissier[a,b] olivier.boissier@emse.fr
G. Picard[a,b] gauthier.picard@emse.fr

[a]Institut Henri Fayol, MINES Saint-Etienne, France

[b]Laboratoire Hubert Curien UMR CNRS 5516, France

Résumé

Le taux d'acceptabilité et la satisfaction utilisateur sont devenus des facteurs clés pour éviter le désabonnement des clients et assurer le succès de tout fournisseur de logiciel en tant que service *(ou SaaS). Néanmoins, le fournisseur doit également minimiser les coûts de location de services* cloud*. Pour faire face à ces objectifs contradictoires, la plupart des travaux considèrent la gestion de ressources de manière unilatérale par le fournisseur. Ainsi, les préférences utilisateur et leur acceptabilité subjective sont ignorées. Des études récentes dans le domaine de la qualité d'expérience (QoE) recommandent aux fournisseurs d'utiliser des métriques comme les quantiles pour jauger plus précisément l'acceptabilité des services. Dans cet article, nous proposons un mécanisme de négociation « one-to-many » adaptatif pour améliorer l'acceptabilité des services d'un fournisseur SaaS. Se basant sur une estimation par quantiles de l'acceptabilité des services et sur l'apprentissage du modèle de négociation de l'utilisateur, ce mécanisme ajuste le processus de négociation du fournisseur afin de garantir un taux d'acceptabilité désiré tout en respectant des contraintes budgétaires. Ce mécanisme est mis en œuvre et ses résultats sont analysés au regard d'approches comparables.*

Mots-clés : *négociation, adaptation, taux d'acceptabilité, SaaS, cloud computing*

1 Introduction

Le taux d'attrition (la proportion de clients perdus sur une période donnée) est un des indicateurs les plus négatifs affectant les fournisseurs de « logiciel en tant que service » (SaaS) . Une étude récente de la dynamique des clients a montré que la majorité des consommateurs aux Etats-Unis ont changé de fournisseurs de service car ils ne répondaient pas à leurs attentes [1]. Par ailleurs, une augmentation significative de la popularité des applications multimédia est prédite [8]. Ainsi, le marché de demain se formera autour d'une demande intensive et fluctuante, avec de fortes attentes client. Afin de palier cette rapide évolution, les fournisseurs de services applicatifs (ASP) migrent de plus en plus vers le *cloud* pour gérer leurs ressources de manière flexible et ainsi minimiser leurs coûts opérationnels. Ainsi, les ASP et les fournisseurs SaaS doivent minimiser le taux d'attrition tout en respectant leurs contraintes budgétaires. Dans le cadre du *cloud computing*, ce problème, objet de nombreuses études, est connu comme la gestion de l'élasticité ou l'*auto-scaling* [4, 15]. Cependant, la plupart de ces travaux adoptent une approche centralisée où l'ASP prend unilatéralement les décisions d'allocation de ressource. Par conséquent, les préférences utilisateurs sont négligées, et il est souvent présumé que leur seuil d'acceptabilité tolère le meilleur service offert.

Les notions de taux d'acceptabilité et de taux d'attrition sont fortement liées à la perception subjective de l'utilisateur sur la qualité de service ou la qualité d'expérience (QoE) [12]. Bien que la plupart des études sur la QoE soient menées sur le plan conceptuel, la QoE est également un bon moyen de fournir une mesure pratique permettant de quantifier la satisfaction des utilisateurs et l'acceptation du service [31]. En particulier, la gestion de la QoE est apparue comme un processus visant à maximiser la QoE tout en optimisant les ressources utilisées [25]. Cependant, la plupart des travaux existants souffrent des deux limitations suivantes. Tout d'abord, la littérature sur la gestion de la QoE repose largement sur le *score d'opinion moyenne* (ou MOS) pour évaluer la satisfaction et l'acceptabilité du service [12]. Ce score a été utilisé à la fois pour les applications NGN (Next Generation Networks) et le *cloud computing*. Toutefois, comme il s'agit d'une moyenne

d'opinions des utilisateurs, MOS masque des informations importantes sur la diversité des utilisateurs et leurs préférences personnelles, alors que cette différence d'opinions peut avoir un impact significatif pour le fournisseur [6]. Pour cette raison, d'autres mesures ont été proposées pour permettre au prestataire de comprendre la satisfaction de l'utilisateur final et d'estimer le taux d'attrition des clients. Par exemple, des quantiles et des percentiles ont été proposés comme des outils de mesure permettant au fournisseur de s'assurer que, par exemple, 95 % de ses utilisateurs trouvent le service acceptable ou mieux [7]. La seconde limitation est que, dans la majorité de ces travaux, la gestion de la QoE reste un processus unilatéral réalisé par le fournisseur sans intégrer les préférences de l'utilisateur final dans la boucle [25]. Par conséquent, malgré l'objectif déclaré d'établir une estimation subjective et centrée sur l'utilisateur de la qualité du service, les travaux existants sont principalement axés sur les fournisseurs.

Par définition, l'approche *agent* est liée à une perspective individuelle [32]. Les agents sont ainsi pertinents pour représenter la subjectivité des opinions des utilisateurs et l'acceptation d'un service donné [17]. En outre, les principes du comportement de négociation discutés dans [19] constituent un outil utile pour représenter les attentes des utilisateurs finaux. Ces derniers sont des déterminants clés de la satisfaction des utilisateurs et de l'acceptabilité des services [23, 33, 14]. De plus, les systèmes multi-agents fournissent une plate-forme distribuée capable d'entreprendre des tâches nécessitant une coopération telle que la gestion des ressources [28]. La négociation multi-agent « one-to-many » est une solution intéressante pour intégrer l'utilisateur final et son acceptabilité subjective dans la gestion de l'élasticité [9]. Néanmoins, la majorité des travaux existants dans la littérature de négociation « one-to-many » traite d'un scénario où le vendeur cherche à trouver un accord *atomique* conclu avec l'un des multiples acheteurs simultanés. En outre, la plupart de ces travaux supposent un ensemble *fermé* de participants dans lequel les adversaires de l'agent unique sont connus à l'avance avant le début du processus de négociation. De plus, dans la plupart des travaux existants, les offres sont envoyées et reçues de manière synchrone. Ces hypothèses ne tiennent pas dans l'écosystème *cloud* ouvert d'aujourd'hui, où des centaines d'utilisateurs peuvent entrer ou quitter le système chaque minute et où les sessions de négociation ne sont pas synchronisées.

Dans cet article, nous développons de nouveaux mécanismes de négociation et de coordination multilatéraux adaptatifs et ouverts permettant au fournisseur d'atteindre un taux d'acceptabilité de service ciblé tout en satisfaisant ses contraintes budgétaires. Les utilisateurs peuvent décider d'accepter ou de rejeter le service proposé en fonction de leurs attentes et d'une estimation subjective de la qualité du service [14]. En fonction de ses mesures de la portion d'utilisateurs qui considèrent que le service est inacceptable, le fournisseur ajuste sa stratégie de négociation afin de ramener le taux d'acceptabilité à ses objectifs prédéfinis. Ainsi, à l'aide du mécanisme proposé, le fournisseur SaaS peut *(i)* s'adapter à la nature dynamique et ouverte de l'écosystème *cloud*, où les utilisateurs peuvent entrer dans le système, entrer le système ou le quitter à volonté, et *(ii)* intégrer les préférences de l'utilisateur dans le processus de décision pour parvenir à des accords mutuellement acceptés en assurant un taux d'acceptabilité prédéfini précis et en atteignant ses objectifs commerciaux. Les mécanismes de négociation et de coordination proposés sont développés dans l'architecture EMan [13].

Le reste de l'article est structuré comme suit. La section 2 introduit brièvement l'architecture EMan dans laquelle vient s'intégrer notre proposition. La section 3 détaille la stratégie de négociation adaptative et l'approche d'apprentissage et de modélisation de l'adversaire proposée. La section 4 détaille le cadre expérimental et discute des résultats obtenus. La section 5 discute des travaux connexes, avant de conclure avec quelques perspectives en section 6.

2 L'architecture EMan

Le mécanisme proposé dans cet article est implémenté dans l'architecture EMan [16, 14, 13]. C'est une architecture multi-agent pour la gestion de l'élasticité de plate-formes SaaS. EMan suit le même schéma architectural partagé par la plupart des solutions « one-to-many » existantes dans la littérature. Dans la version initiale de EMan [14, 13], les utilisateurs sont assistés par des agents autonomes dont le but est de maximiser la QoE de leurs utilisateurs respectifs. Nous montrerons dans la section suivante comment nous y incluons le mécanisme d'adaptation que nous proposons.

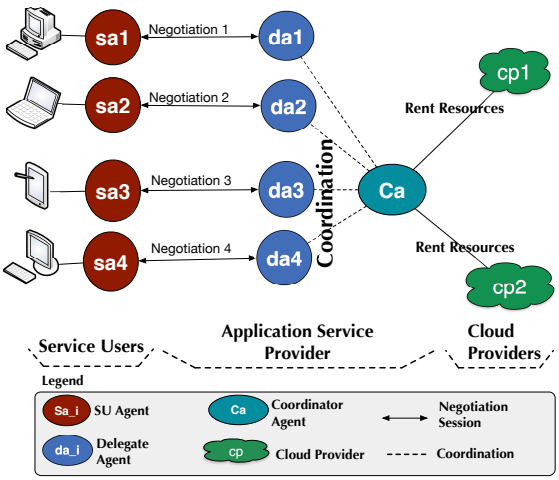


FIGURE 1 – L'architecture EMan déployée dans un écosystème *cloud*

2.1 Agents

L'architecture EMan représentée dans la figure 1 modélise la négociation qui a lieu entre un fournisseur SaaS et ses utilisateurs finaux ou utilisateurs de services (SU). La négociation entre les fournisseurs SaaS et les fournisseurs de *cloud computing* n'est pas décrite dans cet article. L'architecture EMan contient trois types d'agents : les agents utilisateur de service (notés sa_i), les agents délégués (notés da_i) et un seul coordinateur (noté ca). Ces deux derniers types d'agents représentent le fournisseur.

Agents utilisateurs (SA ou sa_i). Un agent utilisateur de service participe au processus de négociation pour le compte d'un utilisateur de service. Un sa_i a une fonction d'utilité M_{sa_i} qui encode ses préférences. M_{sa_i} est utilisée à chaque cycle t pour évaluer l'utilité des offres $o^t_{da_i}$ provenant du délégué correspondant (voir section 2.2). Si le service comporte J attributs, alors M_{sa_i} est défini comme suit :

$$M_{sa_i}(o^t_{da_i}) = \sum_{j=1}^{j=J} w_{sa_i,j} \cdot \mu_{sa_i,at_j}(o^t_{da_i}[at_j]) \quad (1)$$

où w_{sa_i,at_j} est le poids associé à l'attribut at_j pour spécifier l'importance que cet utilisateur donne à cet attribut (ces pondérations doivent satisfaire $\sum_{j=1}^{J} w_{sa_i,at_j} = 1$), $o^t_{da_i}[at_j]$ est la valeur dans l'offre $o^t_{da_i}$ du $j^{\text{ème}}$ attribut (i.e. at_j) et μ_{sa_i,at_j} est la fonction d'utilité de cet attribut. Cette fonction est définie comme suit :

$$\mu_{sa_i,at_j}(o^t_{da_i}[at_j]) = \frac{rv_{sa_i,at_j} - o^t_{da_i}[at_j]}{rv_{sa_i,at_j} - pv_{sa_i,at_j}} \quad (2)$$

où $\mu_{sa_i,at_j}(o^t_{da_i}[at_j]) \in [0,1]$ et pv_{sa_i,at_j} et rv_{sa_i,at_j} sont respectivement la valeur préférée (i.e. la meilleure) et la valeur de réserve (i.e. la pire) de sa_i pour cet attribut [1]. Notons que, comme l'ont confirmé des études empiriques, l'utilisateur est capable de choisir les valeurs de $pvat_j$, rv_{sa_i,at_j} et $w_{sa_i,j}$ [23, 14, 33]. Par conséquent, sa_i peut les obtenir de l'utilisateur. Ces valeurs ne sont pas divulguées au fournisseur.

Pour prendre sa décision d'acceptation ou de rejet, un sa_i s'appuie sur sa fonction d'utilité et sur son *taux d'aspiration* (AR) courant. $AR^t_{sa_i} \in [0,1]$ indique la quantité d'utilité que sa_i s'attend à obtenir dans ce cycle de négociation t. Pour parvenir à des accords, sa_i fait des concessions en réduisant son AR. Tous les sa_i suivent une stratégie de concession temporelle (ou TBC) [5]. Cette hypothèse est assez courante dans la littérature, notamment dans les travaux dont l'objectif est de construire un modèle du comportement de l'adversaire [3]. Par conséquent, $\Delta AR^t_{sa_i}$, la concession faite par sa_i pour le cycle t, dépend du temps restant avant l'échéance. Il est calculé comme suit [27] :

$$\Delta AR^t_{sa_i} = AR^{t-1}_{sa_i} \cdot \left(\frac{t}{T_{sa_i}}\right)^{\lambda_{sa_i}} \quad (3)$$

où T_{sa_i} est la date limite de négociation, et λ_{sa_i} est un paramètre qui contrôle le degré de convexité de la courbe de concession de sa_i. λ_{sa_i} détermine le comportement de sa_i (conciliant, linéaire ou conservateur) [27].

Agents délégués (DA ou da_i). Une fois qu'un nouvel agent utilisateur sa_i entre dans le système, le coordinateur crée un nouvel agent délégué da_i et initie une session de négociation bilatérale avec sa_i. Comme sa_i, da_i a une fonction d'utilité $M_{da_i} \in [0,1]$. Cependant, l'utilité d'une offre $o^t_{sa_i}$ du point de vue du délégué da_i est déterminée par le coût requis pour servir l'offre. RC est une valeur clé qui détermine le comportement de négociation d'un délégué. Il représente le coût de réserve (maximum) que

1. Comme discuté dans [14, 13], μ_{sa_i,at_j} peut également être une fonction logarithmique dérivée de la loi Weber-Fechner Law [30] et de l'hypothèse logarithmique [21]. Cependant, les résultats du mécanisme adaptatif de cet article sont valides pour μ_{sa_i,at_j} linéaire (Equation 2) ou logarithmique.

le délégué dépense sur les utilisateurs. Cette valeur, initialisée par le coordinateur lorsque le da_i est généré, est très importante puisqu'elle permet au fournisseur d'imposer sa contrainte budgétaire, i.e. le coût moyen dépensé pour un utilisateur ne doit pas dépasser RC. Si l'offre n'est pas acceptée par un da_i, il utilise sa stratégie de négociation pour générer une contre-offre. Pour conclure des ententes avec sa_i, le da_i utilise une stratégie de concession fondée sur le temps qui réduit son taux d'aspiration (AR). La concession est calculée comme suit :

$$\Delta AR^t_{da_i} = \frac{1}{T_{da_i}} \tag{4}$$

où T_{da_i} est l'échéance du délégué, initialisée par le ca en fonction de ses connaissances métier. Lorsque da_i atteint T_{da_i}, il ne quitte pas le processus de négociation, il cessera simplement de faire des concessions.

2.2 Négociation bilatérale entre SA et DA

Dans l'architecture EMan, les SA et DA n'ont pas accès aux préférences et aux stratégies de négociation des autres agents. Par conséquent, leur comportement de négociation suit la *fonction de décision de négociation* [5] et est déterminé par la fonction d'utilité et la stratégie de négociation. Quant au coordinateur, il est capable de manipuler les stratégies de négociation des délégués si le mode d'adaptation est actif.

Le service négocié est caractérisé par un ensemble de J attributs, $s = \langle at_1, at_2, \cdots, at_J \rangle$. L'objet o^t_i est l'offre ou contre-offre échangée pendant le processus de négociation à l'étape t, entre sa_i et da_i. Il attribue une valeur v^t_k à chacun des attributs décrivant le service négocié. La négociation entre un da_i et un sa_i est basée sur le *protocole d'offre alternative*. Lorsqu'un sa_i reçoit une offre, il peut (*i*) accepter l'offre, (*ii*) proposer une contre-offre, ou (*iii*) quitter le processus de négociation. Une fois qu'une offre est acceptée par un agent, elle ne peut pas être reniée [13].

Afin de tenir compte de la nature ouverte et dynamique de l'écosystème *cloud*, les sessions de négociation dans l'architecture EMan sont non synchrones, i.e. certaines sessions seront déjà terminées alors que d'autres sessions seront encore actives ou n'auront pas encore démarré. Pour plus d'informations sur le protocole de négociation et la stratégie d'acceptation, veuillez vous référer à [13, 14].

Cette section offrait un aperçu de l'architecture EMan. Nous savons que les protocoles de négociation et les stratégies présentés ci-dessus peuvent être exploitables. Toutefois, cette question dépasse le cadre de cet article. Nous considérons que cette question est d'une importance moindre pour la raison suivante : la négociation dans l'écosystème *cloud* a une forte composante *intégrative* puisque les ressources offertes par le *cloud* semblent être illimitées [29]. Par conséquent, au lieu d'exploiter les stratégies de négociation des utilisateurs et de les forcer à faire de lourdes concessions, le fournisseur est plus susceptible de rechercher des règlements gagnant-gagnant permettant de maximiser le taux d'acceptabilité tout en respectant ses contraintes budgétaires.

3 Stratégie de négociation « one-to-many » adaptative

Cette section détaille notre algorithme adaptatif. Afin d'estimer le taux d'acceptation à tout moment t, le fournisseur s'appuie sur un algorithme d'estimation de quantile présenté dans la section 3.1. Si le taux d'acceptation ciblé est violé, le coordinateur active le mode adaptatif. Avec ce mode actif, un délégué da_i doit analyser le comportement de son « adversaire » [2] sa_i, apprendre un modèle de sa stratégie de négociation afin d'estimer son délai de négociation $\tilde{T_{sa_i}}$ et envoyer cette estimation au coordinateur (section 3.2). Ce dernier choisit les sessions de haute priorité (en fonction de leurs délais) et ajuste leurs stratégies de négociation tout en respectant les contraintes budgétaires (section 3.3).

3.1 Déclenchement de l'adaptation

Les quantiles et les percentiles sont des valeurs qui partagent un ensemble fini de valeurs en q sous-ensembles de tailles (presque) égales. La littérature sur la QoE et la satisfaction de l'utilisateur recommande aux fournisseurs de s'appuyer sur les quantiles et les percentiles comme mesures plus précises (par rapport à MOS) pour évaluer l'acceptabilité du service [7].

Chaque fois que la session i est terminée, le coordinateur est informé par da_i des résultats de cette session. En utilisant ces données, le coordinateur exécute un algorithme d'estimation de

2. Notons les guillemets autours de la notion d'adversaire, terme utilisé en négociation, bien qu'ici le délégué n'ait pas forcément de relation conflictuelle avec l'utilisateur.

quantile pour détecter le taux d'acceptabilité du service courant.

Soit Q la fonction d'estimation quantile ou percentile. Soit R le jeu de données contenant les résultats des sessions terminées. R peut contenir soit 0, pour les sessions échouées, soit 1 pour les sessions réussies. Si le coordinateur cherche à s'assurer que β pourcents des utilisateurs qui ont fait appel au service à ce jour ont eu une session de négociation réussie (acceptant la qualité de service proposée), le coordinateur doit vérifier que le $(100-\beta+1)^{\text{ème}}$ percentile est égal à 1 :

$$Q(R, 100-\beta+1) = 1 \qquad (5)$$

Tant que cette condition est maintenue, le coordinateur n'a pas besoin d'intervenir dans le processus de négociation. Dès que la condition de l'équation 5 est violée, le coordinateur déclenche le mécanisme d'adaptation en commandant à tous les délégués actifs d'activer leur mode adaptatif. De plus, lorsque le coordinateur engendre de nouveaux délégués, ils entreront également dans ce mode actif.

Notez que le coordinateur continue d'évaluer l'équation 5 même après l'activation du mode adaptatif.

La section suivante détaille le rôle des délégués après l'activation du mode adaptatif.

3.2 Apprentissage du comportement de l'adversaire

Estimation de la concession de l'adversaire. Même lorsque le mode adaptatif n'est pas actif, lorsqu'un délégué da_i reçoit $o^t_{sa_i}$ au cycle t du sa_i correspondant, il estime la concession faite par sa_i en comparant $o^t_{sa_i}$ avec $o^{t-1}_{sa_i}$ l'offre précédente faite par sa_i. Puisque da_i n'a pas accès aux préférences de sa_i ou à la fonction M_{sa_i}, il ne peut pas calculer la concession réelle de sa_i. Au lieu de cela, il s'appuie sur sa propre fonction d'utilité pour estimer la concession faite par sa_i en supposant qu'une concession faite par sa_i est synonyme d'un gain d'utilité pour da_i. Ainsi, l'estimation de la concession faite par sa_i au cycle de négociation t, est définie comme suit :

$$c^t_{sa_i} = M_{da_i}(o^t_{sa_i}) - M_{da_i}(o^{t-1}_{sa_i}) \qquad (6)$$

Pour apprendre le comportement de concession de sa_i, da_i effectue le suivi des concessions de sa_i pendant la session de négociation. La figure 2 compare $\Delta AR^t_{sa_i}$, les concessions réelles faites par sa_i (en bleu), avec $c^t_{sa_i}$, l'estimation

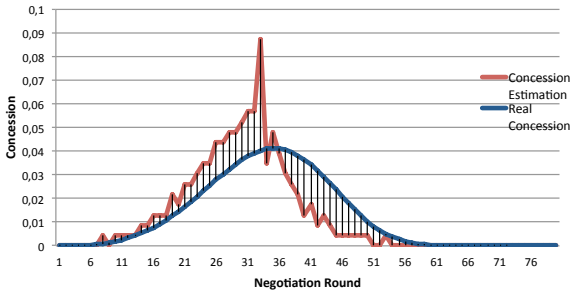


FIGURE 2 – $\Delta AR^t_{sa_i}$, la vraie concession faite par sa_i (en bleu) comparée à $c^t_{sa_i}$, l'estimation faite par da_i (en rouge).

par da_i de la même concession (en rouge). Dans cet exemple, $T_{sa_i} = 80$ cycles et $\lambda_{sa_i} = 3.0$. Comme le montre la figure, bien que la courbe rouge soit bruitée (car elle est estimée à l'aide de la fonction d'utilité de da_i, puis normalisée), elle semble fournir une estimation adéquate de la concession de sa_i puisque l'objectif principal est d'apprendre T_{sa_i}, la date limite de négociation de sa_i.

Apprentissage du modèle de l'adversaire.

Si le mode d'adaptation est actif, les données de concessions recueillies dans le paragraphe précédent devraient être utilisées pour établir un modèle de comportement de concession de sa_i et inférer T_{sa_i}. Ceci peut être réalisé, comme cela a été montré dans la littérature [3], en utilisant une régression non linéaire supposant que tous les utilisateurs suivent une concession basée sur le temps définie par l'équation 3. Cependant, à la différence des travaux existants prédominants dans la littérature, où apprendre le modèle de négociation de l'adversaire est fait et corrigé à chaque cycle une fois qu'une nouvelle concession est faite, dans les arrangements de négociation « one-to-many » abordée dans cet article, une telle approche n'est pas pratique pour des raisons d'évolutivité. En effet, des centaines ou des milliers d'utilisateurs peuvent négocier avec le fournisseur simultanément. En outre, si le fournisseur exécute les algorithmes d'apprentissage des délégués sur les ressources louées sur le *cloud*, cela peut augmenter considérablement les coûts.

Pour cette raison, dans la solution proposée, un délégué peut exécuter l'algorithme de régression non linéaire une seule fois. Il s'agit donc maintenant de déterminer quand un délégué doit le faire. D'une part, si un délégué lance ce pro-

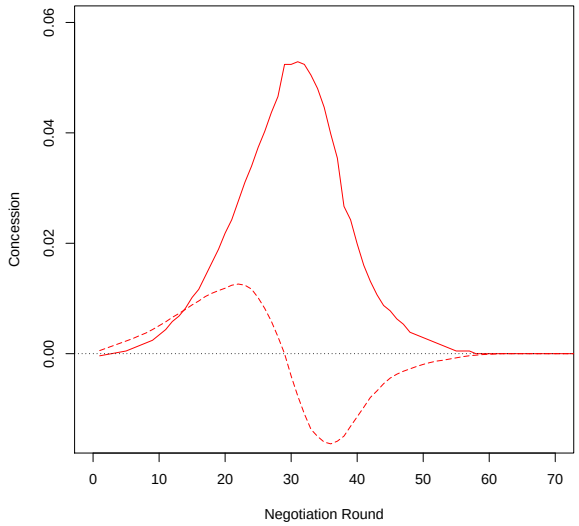


FIGURE 3 – Une version lissée de $c^t_{sa_i}$ (ligne pleine) et sa dérivée (ligne pointillée).

cessus trop tôt dans la session de négociation, le processus de régression n'aura que quelques points d'entrée. D'autre part, si le délégué attend trop longtemps, l'utilisateur correspondant risque d'atteindre son échéance et de quitter le processus de négociation.

Tous les sa_i suivent une stratégie de concession fondée sur le temps dans laquelle le taux de concessions de sa_i ralentit quand sa_i a déjà fait la plupart des concessions qu'il pouvait faire. Par conséquent, quel que soit le type de sa_i (conservateur ou linéaire) et de son échéance réelle T_{sa_i}, lorsque le taux de changement (ou la dérivée) de l'estimation par da_i de la concession de sa_i diminue significativement en valeurs négatives, cela signifie que sa_i a fait la plupart de ses concessions et que, pour le reste de la session de négociation avant que sa_i atteigne T_{sa_i}, sa_i cessera de faire des concessions considérables. Ainsi, si da_i lance sa régression non linéaire à ce stade, il aura le plus de points de données utiles.

La figure 3 superpose $c^t_{sa_i}$ avec sa première dérivée. Le sa_i de cette figure est exactement le même que celui de la figure 2, mais nous avons utilisé un filtre Savitzky-Golay pour lisser la courbe et filtrer le bruit [24]. Pour calculer le taux de variation des concessions, nous calculons la dérivée et nous lissons le résultat en utilisant une variante a du même filtre.

L'algorithme de régression non linéaire prend comme variables d'entrée le cycle de négociation (t) et la concession estimée ($c^t_{sa_i}$) et émet des valeurs estimées des paramètres T_{sa_i} et λ_{sa_i}, notée $\tilde{T_{sa_i}}$ et $\tilde{\lambda_{sa_i}}$ respectivement. Alors, d'après $\tilde{T_{sa_i}}$, da_i calcule $\tilde{r_i} = \tilde{T_{sa_i}} - t$, le nombre estimé de cycles [3] restant dans la session i. Tant que la session i n'est pas terminée, da_i met à jour $\tilde{r_i}$ à chaque cycle et l'envoie à plusieurs reprises au coordinateur.

3.3 Adaptation de la négociation

Comme nous l'avons expliqué dans la section précédente, le coordinateur reçoit $\tilde{r_i}$ de chaque session de négociation i dont le délégué da_i considère que sa_i s'approche de la fin de son échéancier et mérite d'être priorisé. Ces estimations $\tilde{r_i}$ sont stockées dans une *liste des priorités* qui est triée en continu par ordre croissant : une session dont la date limite est estimée plus tôt sera en haut de la liste.

Notons que puisque les sessions de négociation sont non synchronisées, le coordinateur doit répéter le tri chaque fois qu'il reçoit une nouvelle estimation d'un délégué da_i. En outre, chaque fois qu'une session est terminée avec succès ou non, le coordinateur supprime son enregistrement de la liste des priorités. Ainsi, dans la liste des priorités, les sessions sont triées de la plus haute à la plus basse des priorités. Maintenant le coordinateur doit décider combien de ces sessions prioritaires doivent être *priorisées*. Pour ce faire, le coordinateur entreprend ce processus comme suit :

1. Estimer le taux d'acceptabilité courant (en utilisant la fonction d'estimation de quantile comme décrit dans la section 3.1). Ensuite, calculer la différence entre le taux d'acceptabilité souhaité et sa valeur courante actuelle.
2. Estimer p le nombre de sessions réussies requises pour rétablir le taux désiré et choisir les premières p sessions de la liste de priorités et les *prioriser* en ajustant leurs stratégies de négociation pour les encourager à accepter le service.

Lorsque la stratégie de négociation d'un délégué da_i est ajustée, da_i utilise une stratégie de concession fondée sur le temps définie dans la section 2.1, affectées des modifications suivantes. Tout d'abord, le délai de négociation de da_i devient $\tilde{r_i}$ au lieu de T_{da_i}. Deuxièmement,

3. Les termes *cycle* et *round* sont utilisés indifféremment ici.

le coût de réserve de da_i est augmenté de la valeur notée $RcPrio$. $RcPrio$ est calculé en divisant l'excédent disponible dans *Surplus* parmi toutes les sessions priorisées. *Surplus* est une variable dans laquelle le coordinateur accumule le surplus obtenu lors de sessions de négociation réussies. Plus précisément, lorsqu'une session est déclarée réussie, le coordinateur met à jour sa variable *Surplus* comme suit :

$$\textit{Surplus} \leftarrow \textit{Surplus} + surplus_i \quad (7)$$

où $surplus_i$ est l'excédent obtenu de la session de négociation réussie i. $surplus_i$ est calculé comme suit :

$$surplus_i = RC - Cost(\Pi(\hat{o}_i)) \quad (8)$$

Où RC est le coût de réserve des délégués, $\hat{o}_i$ est l'offre acceptée dans la session i et $Cost(\Pi(\hat{o}_i))$ est le coût à payer par le fournisseur pour satisfaire à cette offre. Ainsi, la différence entre le coût réel et le coût (maximum) de la réserve est considérée comme excédentaire. Notez que les délégués ne peuvent pas obtenir plus de $RcPrio_{max} = RC/2$, considéré comme la valeur maximale pour les sessions priorisées. Troisièmement, le coût préféré de da_i est changé pour le coût de la dernière offre qu'il a faite. $AR^t_{da_i}$ est remis à 1.

Cette section a présenté la modélisation de l'adversaire et les processus d'adaptation de la négociation. La section suivante présente l'évaluation expérimentale de notre approche

4 Évaluation expérimentale

Pour évaluer le mécanisme adaptatif, nous l'implémentons dans l'architecture EMan [14, 13]. Cette dernière est une architecture multi-agent pour la gestion de l'élasticité SaaS mise en œuvre à l'aide de Repast Simphony [18], un environnement de simulation multi-agent. L'évaluation est organisée en trois expériences. La première expérience (section 4.2) vise à évaluer l'algorithme d'adaptation. La deuxième expérience (section 4.3) examine l'impact de la charge de travail appliquée au fournisseur (c'est-à-dire le nombre d'utilisateurs entrant dans le système par minute) sur le taux d'acceptabilité. La troisième expérience (section 4.4) évalue la surcharge des mécanismes de négociation et coordination.

4.1 Paramètres expérimentaux

Le nombre total d'utilisateurs entrant dans le système est noté $|SU|$. Dans les expériences suivantes, $|SU| = 10000$ utilisateurs. Les utilisateurs entrent dans la simulation suivant un processus aléatoire de Poisson dont la valeur moyenne est A par minute. Le service dans le scénario expérimental implique deux attributs : l'un est le délai de livraison du service tandis que l'autre représente la qualité du service. Les profils utilisateur (réserve, valeurs préférées et pondérations pour chaque attribut) sont générés au hasard. Les délais de négociation des sa_i sont générés au hasard : $T_{sa_i} \in [40 : 120]$, et $\lambda_{sa_i} \in [1 : 8]$ (i.e. les utilisateurs peuvent être linéaires ou conservateurs). Le coût des services acceptables par les utilisateurs varie de 0.1 à 0.9. RC, le coût de réserve du délégué (le coût maximal alloué à un utilisateur non prioritaire) est fixé à 0.6. Ce paramètre représente les contraintes budgétaires du fournisseur. Par conséquent, sans le mode d'adaptation, environ un quart des utilisateurs n'accepteront pas le service puisque RC ne peut pas satisfaire leur service le moins acceptable, $RcPrio_{max} = RC/2$. Par conséquent, lorsqu'un utilisateur sa_i est prioritaire, le RC du délégué respectif da_i est au plus augmenté de $RC/2$. $Goal$ est le pourcentage d'utilisateurs que le fournisseur cherche à satisfaire. Son impact est évalué dans la section suivante. Notez que les résultats décrits ci-dessous restent valides même si les valeurs des paramètres ci-dessus (par exemple, RC, $RcPrio_{max}$) sont modifiées. De plus, nous avons obtenu des résultats similaires lorsque la fonction d'utilité d'attribut d'utilisateur est une fonction logarithmique dérivée de l'hypothèse logarithmique [21, 14].

4.2 Évaluation du mécanisme d'adaptation

La figure 4 montre les résultats de cette expérience. La courbe bleue représente le taux d'acceptation lorsque $Goal = 95\%$. Comme le montre la figure, au début de la simulation, la valeur du taux d'acceptation a oscillé avant de se stabiliser sur la valeur d'objectif ($Goal = 95\%$). Les courbes rouge et marron, qui tracent le taux d'acceptation lorsque $Goal = 90\%$ et $Goal = 85\%$, montrent des résultats similaires. Pour comparer les résultats du mécanisme d'adaptation, la courbe noire trace le taux d'acceptation lorsque le mécanisme est désactivé.

Comme on peut le constater à partir de ces résultats, le mécanisme d'adaptation a réussi à rétablir le taux d'acceptation de l'objectif prédéfini $Goal$ d'une manière précise. Ceci est expliqué par le fait que l'algorithme de prédiction discuté dans la section 3.2 a réussi à prédire le délai de négociation utilisateur T_{sa_i} avec une précision

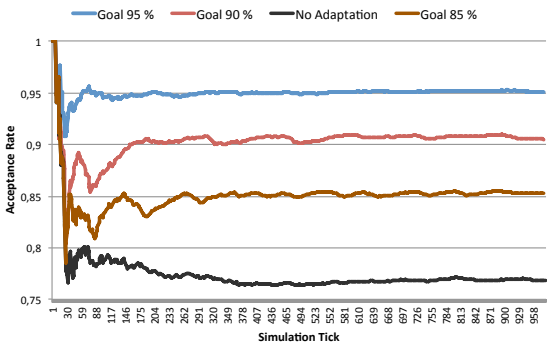


FIGURE 4 – Taux d'acceptation avec $A = 160$.

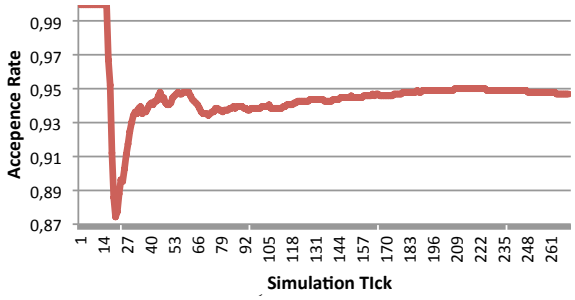


FIGURE 5 – Taux d'acceptation ($Goal$=95% et $A = 1280$).

TABLE 1 – Impact de la charge de travail A sur le taux d'acceptation

A	20	40	80	160	320	640	1280	2500
Avec adaptation	95%	95%	95%	95%	95%	94.9%	94.7%	93.3%
Sans adaptation	77%	77%	77%	77%	77%	77%	77%	77%

acceptable. Le taux d'erreur moyen entre la prédiction $\tilde{T_{sa_i}}$ et la valeur réelle T_{sa_i} est d'environ 6 cycles, ce qui signifie qu'un délégué da_i surestime/sous-estime T_{sa_i} par trois cycles de négociation moyenne.

Intuitivement, le taux d'acceptabilité de service atteint par l'algorithme de prédiction est livré avec plus de coûts investis par utilisateur. Cependant, cette augmentation ne viole pas la contrainte budgétaire du fournisseur : le coût moyen par utilisateur était de 0.55, 0.52, 0.5 et 0.44 pour $Goal = 95\%$, $Goal = 90\%$, $Goal = 85\%$ et non-adaptatif, respectivement. Ainsi, la contrainte budgétaire du fournisseur (i.e. le coût moyen par utilisateur ne dépasse pas $RC = 0.6$) a été satisfaite parce que le fournisseur s'appuie sur le *Surplus* pour servir les utilisateurs priorisés.

4.3 Impact de la charge A

Cette expérience étudie l'impact de A, la charge de travail appliquée au système, sur le taux d'acceptation afin d'évaluer l'élasticité du mécanisme d'adaptation.

La table 1 montre le taux d'acceptation, à la fin de la simulation, lorsque $Goal = 95\%$ [4] et A augmentant de 20 à 2500 utilisateurs par minute. Comme le montre le tableau, le mécanisme s'est avéré très élastique. Lorsque $A \in [20 : 320]$ le taux d'acceptation ne change pas à mesure que A augmente. Toutefois, lorsque A augmente à des valeurs plus élevées (640 à 2500), le taux d'acceptation témoigne d'une légère diminution. Cependant, même avec un très haut A, les résultats obtenus par le mécanisme d'adaptation restent très proches de l'objectif. Cette diminution est expliquée comme suit : l'algorithme d'adaptation est réactif et le coordinateur ne peut évaluer le taux d'acceptation qu'à chaque fois qu'une session se termine avec succès ou sans succès. Ainsi, lorsque A est trop élevé, un nombre significatif de sessions peut échouer dans le même cycle de simulation et l'algorithme d'adaptation exécuté par le coordinateur souffre d'un léger retard et devient un peu court pour répondre à l'objectif prédéfini ($Goal = 95\%$), comme on peut le voir dans le tableau. Pourtant, ce résultat est à relativiser pour la raison suivante : $|SU|$, le nombre total d'utilisateurs dans la simulation, influence le seuil d'élasticité du mécanisme d'adaptation. Par exemple, lorsque $|SU| = 10000$ et $A = 2500$, les utilisateurs entrent dans le système en 4 vagues massives représentant environ un quart du nombre total d'utilisateurs. Ainsi, le coordinateur n'a pas assez de temps pour rétablir le taux d'acceptation à ($Goal = 95\%$). La figure 5 montre comment le mécanisme d'adaptation réagit à une chute du taux d'acceptation lorsque $A = 1280$. Le coordinateur parvient à restaurer la valeur de l'objectif prédéfini ($Goal = 95\%$). Pourtant, une nouvelle chute se produit avec la nouvelle vague d'utilisateurs et la simulation est terminée. Notez que dans un scénario réaliste, lorsque $|SU|$ tend vers l'infini, le coordinateur sera toujours en mesure de restaurer le taux d'acceptation dans sa valeur d'objectif, même avec un A significatif.

4. Nous avons effectué la même expérience avec différentes valeurs de $Goal \in \{92\%, 90\%, \ldots\}$ et avons obtenu des résultats similaires.

4.4 Surcoût de la coordination et de la négociation

Pour évaluer les surcoûts induits par la coordination et les mécanismes de négociation, nous comptons le nombre de messages échangés pour entreprendre l'intervention du coordinateur et le nombre de messages échangés respectivement lors des sessions de négociation bilatérales.
Sans le mécanisme d'adaptation, le coordinateur est sollicité une seule fois par session pour obtenir le résultat de la session (succès ou échec). Avec le mécanisme adaptatif, le nombre de messages échangés entre les délégués et le coordinateur augmente d'environ 50%. Nous considérons que cette augmentation est insignifiante pour les raisons suivantes. Tout d'abord, même avec le mécanisme adaptatif, les tâches assumées par le coordinateur sont légères comme cela a été discuté dans les sections 3.1 et 3.3. Deuxièmement, étant donné qu'il est probable que le fournisseur SaaS gère les délégués et le coordinateur sur le même centre de données dans le *cloud*, le coût de communication entre eux est négligeable.

En ce qui concerne le nombre de messages de négociation, les résultats montrent une légère diminution ($\approx 7\%$) lorsque le mécanisme adaptatif est actif, car ce dernier aide à conclure des accords plus rapidement.

5 Travaux connexes

En dépit de la littérature relativement riche sur les négociations « one-to-many », l'objectif de la plupart de ces travaux, modélisant des négociations entre un acheteur et de nombreux concurrents vendeurs, est de trouver un accord unique (i.e. *atomique*) qui maximise l'utilité de l'acheteur tandis que d'autres sessions, moins bénéfiques, sont avortées [10, 20]. En outre, les mécanismes de négociation et de coordination existants (par exemple [20, 10]) sont principalement *fermés* et *synchrones*. Par conséquent, ces solutions ne peuvent pas tenir compte de la nature agile de l'écosystème où des milliers d'utilisateurs peuvent surgir sur le portail de services.
Les travaux récents dans le domaine de la composition du service utilisent la négociation « one-to-many » pour atteindre des accords *composite* avec plus d'un vendeur de services atomiques afin de construire un service composite [22, 11]. Dans [22], les auteurs proposent un mécanisme de négociation « one-to-many » qui redistribue le surplus obtenu lors des sessions de négociation réussies aux sessions de négociation en cours afin d'augmenter leurs chances de succès. Néanmoins, ces travaux supposent l'ensemble de vendeurs atomiques *fermé* : ils sont tous connus avant le début de la négociation.

La négociation « one-to-many » a également été utilisée dans le domaine du *cloud computing*. Les auteurs de [2] développent une approche basée sur la négociation pour gérer l'allocation des ressources dans ce cadre. Cependant, un fournisseur accepte une offre si et seulement s'il peut obtenir un gain immédiat en acceptant l'offre. Pour cette raison, le taux d'acceptabilité des utilisateurs n'est pas pris en compte et aucun mécanisme d'adaptation n'est proposé. [26] développent un mécanisme de négociation simultanée de SLA dans les systèmes basés sur le *cloud*. Pourtant, la principale contribution du travail semble être axée sur le protocole. Par conséquent, il ne fournit pas un mécanisme pour représenter l'acceptabilité des utilisateurs ni offre une solution pour ajuster les comportements de négociation des délégués pour tenir compte des objectifs du fournisseur.

6 Conclusions

Cet article a présenté un mécanisme de négociation « one-to-many » adaptatif conçu pour améliorer le taux d'acceptabilité d'un fournisseur SaaS tout en respectant ses contraintes budgétaires. L'approche proposée confère au fournisseur un contrôle précis du taux d'acceptabilité souhaité. En outre, comme cela a été montré expérimentalement, avec la solution proposée le fournisseur de SaaS est capable de faire face aux pics de charge de manière adéquate. Notre travail de recherche futur visera à donner au fournisseur un contrôle plus fin sur le niveau de satisfaction des utilisateurs qu'il cherche à atteindre. En particulier, le fournisseur doit être en mesure de s'assurer qu'un pourcentage prédéfini d'utilisateurs considère le service comme étant « *bon* » ou « *meilleur* » [7]. Plusieurs rapports de l'ITU (Union internationale des télécommunications) et de l'ETSI (Institut européen des normes de télécommunications) recommandent d'aller dans cette direction [7].

Références

[1] Accenture. Accenture 2013 global consumer pulse survey global & u.s. key findings. report, Dublin, Ireland, 2013.

[2] B. An, V. Lesser, D. Irwin, and M. Zink. Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems : volume 1-Volume 1*, pages 981–988, 2010.

[3] T. Baarslag, M. J. Hendrikx, K. V. Hindriks, and C. M. Jonker. Learning about the opponent in automated bilateral negotiation : a comprehensive survey of opponent modeling techniques. *Autonomous Agents and Multi-Agent Systems*, pages 1–50, 2015.

[4] E. F. Coutinho, F. R. de Carvalho Sousa, P. A. L. Rego, D. G. Gomes, and J. N. de Souza. Elasticity in cloud computing : a survey. *annals of telecommunications-annales des télécommunications*, 70(7-8) :289–309, 2015.

[5] P. Faratin, C. Sierra, and N. R. Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3) :159–182, 1998.

[6] T. Hobfeld, R. Schatz, and S. Egger. Sos : The mos is not enough ! In *Quality of Multimedia Experience (QoMEX), 2011 Third International Workshop on*, pages 131–136. IEEE, 2011.

[7] T. Hoßfeld, P. E. Heegaard, M. Varela, and S. Möller. Qoe beyond the mos : an in-depth look at qoe via better metrics and their relation to mos. *Quality and User Experience*, 1(1) :2, 2016.

[8] C. V. N. Index. Cisco vni forecast and methodology, 2015-2020. cisco white paper, june 1, 2016, 2016.

[9] A. R. Lomuscio, M. Wooldridge, and N. R. Jennings. A classification scheme for negotiation in electronic commerce. *Group Decision and Negotiation*, 12(1) :31–56, 2003.

[10] K. Mansour and R. Kowalczyk. A meta-strategy for coordinating of one-to-many negotiation over multiple issues. In *Foundations of Intelligent Systems*, pages 343–353. Springer, 2012.

[11] K. Mansour and R. Kowalczyk. On dynamic negotiation strategy for concurrent negotiation over distinct objects. In *Novel Insights in Agent-based Complex Automated Negotiation*, pages 109–124. Springer, 2014.

[12] S. Möller and A. Raake. *Quality of Experience*. Springer, 2014.

[13] A. Najjar. *Multi-Agent Negotiation for QoE-Aware Cloud Elasticity Management*. PhD thesis, École nationale supérieure des mines de Saint-Étienne, 2015.

[14] A. Najjar, C. Gravier, X. Serpaggi, and O. Boissier. Modeling user expectations satisfaction for saas applications using multi-agent negotiation. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 399–406, Oct 2016.

[15] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier. Survey of elasticity management solutions in cloud computing. In *Continued Rise of the Cloud*, pages 235–263. Springer, 2014.

[16] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier. Une négociation multi-partite pour une gestion élastique des applications hébergées dans un cloud. In *Journées Fracophones Systèmes Multi-Agent*, Oct. 2014.

[17] A. Najjar, X. Serpaggi, C. Gravier, and O. Boissier. Multi-agent systems for personalized qoe-management. In *Teletraffic Congress (ITC 28), 2016 28th International*, volume 3, pages 1–6. IEEE, 2016.

[18] M. J. North, T. R. Howe, N. T. Collier, and J. Vos. The repast simphony runtime system. In *Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, 2005.

[19] D. G. Pruitt. *Negotiation behavior*. Academic Press, 2013.

[20] I. Rahwan, R. Kowalczyk, and H. H. Pham. Intelligent agents for automated one-to-many e-commerce negotiation. In *Australian Computer Science Communications*, volume 24, pages 197–204. Australian Computer Society, Inc., 2002.

[21] P. Reichl, S. Egger, R. Schatz, and A. D'Alconzo. The logarithmic nature of qoe and the role of the weber-fechner law in qoe assessment. In *Communications (ICC), 2010 IEEE International Conference on*, pages 1–5. IEEE, 2010.

[22] J. Richter, M. Baruwal Chhetri, R. Kowalczyk, and Q. Bao Vo. Establishing composite slas through concurrent qos negotiation with surplus redistribution. *Concurrency and Computation : Practice and Experience*, 24(9) :938–955, 2012.

[23] A. Sackl and R. Schatz. Evaluating the impact of expectations on end-user quality perception. In *Proceedings of International Workshop Perceptual Quality of Systems (PQS)*, pages 122–128, 2013.

[24] A. Savitzky and M. J. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8) :1627–1639, 1964.

[25] R. Schatz, M. Fiedler, and L. Skorin-Kapov. Qoe-based network and application management. In *Quality of Experience*, pages 411–426. Springer, 2014.

[26] M. Siebenhaar, U. Lampe, D. Schuller, R. Steinmetz, et al. Concurrent negotiations in cloud-based systems. In *Economics of Grids, Clouds, Systems, and Services*, pages 17–31. Springer, 2012.

[27] S. Son and K. M. Sim. A price-and-time-slot-negotiation mechanism for cloud service reservations. *IEEE Transactions on Systems, Man, and Cybernetics*, 42(3) :713–728, 2012.

[28] D. Talia. Clouds meet agents : Toward intelligent cloud services. *IEEE Internet Computing*, (2) :78–81, 2012.

[29] L. L. Thompson, J. Wang, and B. C. Gunia. Negotiation. *Annual review of psychology*, 61, 2010.

[30] L. L. Thurstone. Psychophysical analysis. *The American journal of psychology*, 38(3) :368–389, 1927.

[31] I. Wechsung and K. De Moor. Quality of experience versus user experience. In *Quality of Experience*, pages 35–54. Springer, 2014.

[32] M. Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

[33] V. A. Zeithaml, L. L. Berry, and A. Parasuraman. The nature and determinants of customer expectations of service. *Journal of the academy of Marketing Science*, 21(1) :1–12, 1993.

Approches multiagents pour l'allocation de courses à une flotte de taxis autonomes

G. Picard gauthier.picard@emse.fr
F. Balbo flavien.balbo@emse.fr
O. Boissier olivier.boissier@emse.fr

Univ Lyon, MINES Saint-Étienne, CNRS, Laboratoire Hubert Curien UMR 5516, Saint-Étienne, France

Résumé

Ce travail étudie le problème d'allocation décentralisée de courses à une flotte de taxis autonomes. Classiquement, pour résoudre ce problème les demandes sont centralisées dans un portail où un dispatcheur alloue les courses aux taxis (idéalement de manière optimale). Ceci nécessite que les taxis aient accès en continu au portail (via un réseau cellulaire). Cependant, avoir accès à une telle infrastructure de communication globale coûte cher à la société de gestion de taxis. L'idée est ici d'utiliser une infrastructure véhicule-à-véhicule, peu coûteuse, pour coordonner les taxis sans infrastructure de communication globale. Notre approche est présentée et évaluée de manière empirique par simulation. Nous avons développé différentes stratégies multiagents, requérant différentes infrastructures de communication et mécanismes de coordination, et les analysons en terme de qualité de service, de satisfaction client, et de gain.

Mots-clés : *DCOP, allocation de ressources, taxis autonomes*

Abstract

This work is interested in decentrally solving a taxi allocation problem over a fleet of autonomous taxis. Classically, to solve this problem, requests are centralized into a portal where a dispatcher allocates requests to taxis (ideally, in an optimal manner). This requires taxis have continuous access to the portal. However, getting access to such global communication infrastructureis very expensive for taxi companies. The idea here is to use new affordable vehicle-to-vehicle communication technologies to coordinate taxis without global communication infrastructure. Our approach is presented and empirically evaluated via simulations. We have developed different scenarios with different communication infrastructure and coordination mechanisms, and we analyze, their resulting quality of service, user welfare and gain.

Keywords: *DCOP, resource allocation, autonomous taxis*

1 Introduction

Le développement de véhicules autonomes capables de communiquer en pair-à-pair et le succès des solutions de transport à la demande (e.g. Uber, Lyft, Heetch) sont les principales motivations de ce travail. En effet, la jonction de ces deux phénomènes amène à l'automatisation du processus de résolution du problème d'allocation dans le domaine du transport à la demande. Ainsi, dans le cadre d'un partenariat avec le constructeur Renault, nous étudions le paramétrage d'une flotte de taxis électriques pour répondre sans contrôle centralisé à des demandes de déplacement en milieu urbain et avec un coût de communication limité. En effet, les solutions commerciales de gestion de flottes de véhicules avec collecte de données en temps réel et exploitation, notamment dans le cloud, deviennent rapidement coûteuses selon le rythme de collecte et le volume de données à traiter. Par conséquent, développer une solution décentralisée reposant sur de la communication inter-véhiculaire (VANET) avec des performances équivalentes présente un intérêt économique en plus d'un intérêt scientifique.

La modélisation et le développement de systèmes décentralisés est au cœur des travaux multiagents. Le problème d'allocation de taxis peut donc bénéficier de ces recherches comme ce fut le cas pour [3, 5, 8, 11, 13]. De plus, si l'objectif est une solution optimale – à savoir une allocation optimale – les travaux issus du domaine de l'optimisation sous contraintes distribuée constituent une approche pertinente. Ici, nous proposons donc un modèle multiagent du problème d'allocation pour une flotte de taxis (taxi swarm allocation problem ou TSAP). Il s'agit d'une instanciation du modèle d'allocation de ressources en temps réel (online resource allocation model, ou OLRA) avec l'ajout de contraintes sur la communication (global ou P2P) [14]. Nous étudions les bénéfices et les inconvénients de ces différentes stratégies par rapport à une solution centralisée pour résoudre ce problème d'al-

location distribuée et dynamique de ressources. Notre objectif est de concevoir un processus multiagent d'allocation décentralisé capable de rivaliser avec une solution centralisée, tout en palliant l'absence de connaissance sur les demandes (aucune prévision sur l'échéance et la localisation des requêtes) et de communication globale.

Nous commençons par la présentation du problème d'allocation de taxi et transformons une partie de celui-ci en un programme linéaire en nombres entiers, comme fait classiquement dans une allocation de ressource, en section 2. Nous présentons la contribution principale de ce papier, le modèle multiagent, en section 3, et évaluons ce modèle (section 4) en utilisant le simulateur dédié que nous avons développé. Nous discutons les travaux liés et les approches classiques en section 5, avant de conclure sur quelques perspectives en section 6.

2 Problème d'allocation de Taxi (TSAP)

Dans le scénario considéré (cf. figure 1), les demandes sont générées de manière non déterministe [1] par des sources émettrices. Les taxis sont mobiles, distribués et communiquent via un réseau inter-véhiculaire ad-hoc (VANET, représenté en bleu et rouge) ou par un réseau cellulaire (4G, non représenté comme réseau global) pour répondre aux demandes. Les taxis peuvent échanger des informations sur les demandes qu'ils connaissent, leurs statut ou leurs décisions. L'objectif principal est de trouver une allocation de taxis aux demandes tout en minimisant les coûts opérationnels et en maximisant le bien être des utilisateurs.

2.1 Définition du problème

Le scénario considéré se déroule dans une ville donnée représentée par son plan.

Définition 1 *Une* carte *est un graphe étiqueté* $\langle \mathcal{V}, \mathcal{E}, \mathcal{D}, \mathcal{S} \rangle$*, où* $\mathcal{V}$ *est un ensemble de nœuds,* $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ *est un ensemble d'arcs,* $\mathcal{D} = \{d_{ij} \in \mathbb{R}_+ | i, j \in \mathcal{V}, \exists e_{ij} \in \mathcal{E}, i \neq j\}$ *est l'ensemble des étiquettes précisant les distances entre les nœuds connectés, et* $\mathcal{S} \subseteq \mathcal{V}$ *est un ensemble de sources.*

A partir de cette carte, plusieurs informations sont supposées être facilement disponibles,

1. L'hypothèse est que nous n'avons pas de modèle probabiliste de la distribution de ces demandes.

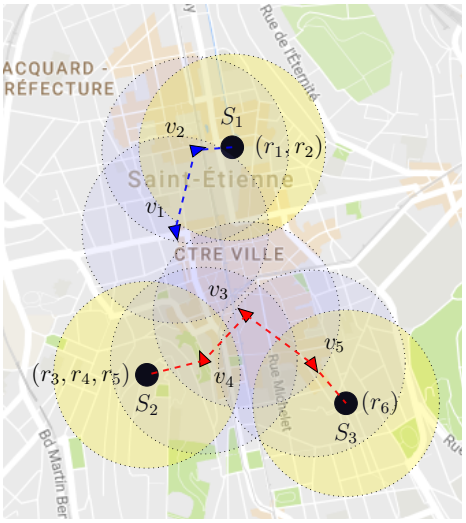


FIGURE 1 – Exemple d'une instance du problème d'allocation de taxi (TSAP) au temps t, où deux taxis (v_i) peuvent uniquement interagir (lignes en pointillés) lorsqu'ils sont dans la même sphère de communication (cercles en pointillés). Les demandes (r_j) sont émises aux sources (S_k).

telles que le chemin le plus court d'un point à un autre, la distance d'un chemin donné $dist(p_1, p_2)$, le temps pour atteindre une position $travel(p_1, p_2, t)$, etc. –informations qui sont communément disponibles dans systèmes de navigation en ligne et hors ligne. Sur cette carte, émises par des noeuds sources de $\mathcal{S}$, des demandes pour atteindre une destination sont générées par des voyageurs. Soit $\mathcal{R}$ l'ensemble de ces demandes.

Définition 2 *Une* demande $r \in \mathcal{R}$ *est définie par la source de son origine* $(origine : \mathcal{R} \to \mathcal{S})$ *et de sa destination* $(destination : \mathcal{R} \to \mathcal{S})$ *ainsi qu' une fenêtre temporelle de validité* $(tw : \mathcal{R} \to [T, T])$*.*

$origine(r)$ (resp. $destination(r)$) fournit le nœud source (resp. destination) de la demande r. Pour simplifier la présentation, les valeurs minimales et maximales de la fenêtre temporelle $tw(r)$ d'une demande r sont respectivement notées $tw_{min}(r)$, $tw_{max}(r)$. Cette fenêtre temporelle définit l'intervalle dans lequel la demande doit être satisfaite, comme dans le problème classique Demand-Responsive Transportation systems (DRTS) [6]. Pour simplifier (pas de contrainte sur la durée de la course), la fenêtre temporelle de dépôt n'est pas prise en compte. En fait, cette fenêtre peut être calculée par translation de la fenêtre temporelle de la prise du pas-

sager $tw(r)$ avec le temps pour passer du nœud de prise de passager à celui de dépôt. Une demande est dite *active* au temps t si t est dans la fenêtre de temps. Les demandes sont émises via l'infrastructure de communication qui peut être globale (e.g. réseau cellulaire) ou locale (e.g. diffusion par VANET) et ne sont pas gérées par les taxis.

Définition 3 *Un* taxi $v \in \mathcal{A}$ *est caractérisé par sa position* ($pos : \mathcal{A} \times T \to \mathcal{V}$)*, sa destination courante* ($dest : \mathcal{A} \times T \to \mathcal{V} \cup \{\emptyset\}$) *et un rayon de communication fixé* ($rng : \mathcal{A} \to \mathbb{R}_+$) *dépendant de son équipement de communication.*

$dest(v,t)$ est le nœud de destination du taxi v au temps t. Si $dest(v,t) = \emptyset$ alors le taxi n'a pas de destination précisée.

Soit $||p_1 - p_2||$ la distance euclidienne entre i et j. Deux composants (taxi ou source) sont connectés s'ils sont situés dans la plus petite sphère de communication centrée sur l'un des deux composants. Dans ce cas, ils peuvent communiquer de manière bidirectionnelle :

$$connected(i,j,t) = \begin{cases} 1, & \text{si } ||pos(i,t) - pos(j,t)|| \\ & \quad \leq min(rng(i), rng(j)) \\ 0, & \text{sinon} \end{cases} \quad (1)$$

Définition 4 *Un* ensemble connecté *est l'ensemble des sources et taxis connectés directement ou par transitivité à une source ou un taxi* i *:*

$$\begin{aligned} CSet(i,t) \quad &= \{j \in \mathcal{S} \cup \mathcal{A} \mid connected(i,j,t)\} \\ &\cup \{j \in \mathcal{S} \cup \mathcal{A} \mid \exists k \in CSet(i,t) \\ &\cup \mathcal{A} \cap CSet(j,t), k \neq i \wedge k \neq j\} \end{aligned}$$

Notons $CSet_{\mathcal{A}}(i,t) = CSet(i,t) \cap \mathcal{A}$ *la restriction de* $CSet(i,t)$ *aux taxis, et* $CSet_{\mathcal{A}}(t) = \{CSet_{\mathcal{A}}(i,t) \mid i \in \mathcal{A}\}$ *l'ensemble de ces ensembles connectés au temps* t*.*

En cas de rayon de communication infini, il n'y a qu'un seul ensemble connecté : l'ensemble de tous les taxis et sources.

Exemple 1 *Sur la figure 1, la carte est le graphe représentant la ville de Saint-Etienne. Trois sources* (S_1 *à* S_3*) émettent des demandes* (r_1 *à* r_6*). Cinq taxis* (v_1 *à* v_5*) sont déployés dans la ville pour répondre aux demandes. L'infrastructure de communication choisie est un VANET. Les taxis et sources ont donc un rayon de communication limité (cercle en pointillé). Par conséquent, deux ensembles connectés sont identifiés* ($CSet(v_1) = \{v_1, v_2, S_1\}$ *et* $CSet(v_3) = \{v_3, v_4, v_5, S_2, S_3\}$*).*

A partir des concepts et définitions introduits, le problème considéré dans ce papier est défini comme suit :

Problème 1 (TSAP) *Le* problème d'allocation de Taxi *(TSAP) consiste en une affectation continue des taxis (incluant les taxis transportant des passagers) à de nouvelles demandes, sur une infrastructure de communication donnée, tout en minimisant les coûts et en maximisant les services aux utilisateurs, pour une période donnée.*

Dans notre étude, les coûts correspondent à la *distance* totale parcourue par les taxis ; le service aux utilisateurs est évalué par le ratio entre le nombre des demandes servies et le nombre total de demandes et également le temps moyen d'attente. La période de temps considérée est la période de travail d'un taxi sur une journée (matin au soir).

Le premier problème à considérer dans le cadre de TSAP est le manque de connaissance sur les demandes à venir. C'est la raison pour laquelle le problème consiste en l'affection continue des taxis aux nouvelles demandes. De manière évidente, il n'est pas possible de définir un plan optimal pour les taxis sur la fenêtre temporelle. Ainsi, le système devra réagir et résoudre les sous problèmes à chaque arrivée d'une demande.

Problème 2 (TSAP(t)**)** *Un TSAP*(t) *consiste en une affectation des demandes actives à un ensemble de taxis de TSAP, au temps* t*.*

L'idée ici est de résoudre TSAP(t) à chaque arrivée de demande dans le système. Notons qu'à la fin d'une période de temps de TSAP, résoudre chaque TSAP(t) peut ne pas conduire à l'optimal théorique. De plus, les critères d'optimisation de TSAP ne peuvent pas être dérivés directement pour un TSAP(t), puisque ce dernier concerne un instant. Notons qu'il est possible de changer l'affectation d'un taxi libre à des demandes tant que ce taxi n'a pas chargé ses clients, comme dans [4, 7, 8].

Un autre problème à considérer en résolvant TSAP est que l'infrastructure de communication

peut ne pas assurer que toutes les demandes sont connues dans le système et que les taxis sont atteignables à tout moment. L'approche choisie dans ce papier est de proposer des mécanismes multiagents pour les taxis afin d'échanger les informations utiles et de se coordonner pour trouver des solutions aux TSAP(t), dans des configurations totalement décentralisées.

2.2 Modéliser TSAP(t) en programme linéaire en nombres entiers

Comme il est classique de procéder pour l'allocation de ressources, TSAP(t) peut être modélisé comme un problème de programmation linéaire en nombres entiers binaires (*PL en 0-1*). Soit v_{ij}^t une variable représentant la décision binaire d'un taxi i d'avoir pour prochaine destination l'origine de la demande j (ou aucune destination) au temps t.

Problème 3 (*PL en 0-1*-TSAP(t)) *Selon le formalisme PL en 0-1, TSAP(t) est défini ainsi :*

$$\min_{v_{ij}^t} \quad \sum_{v_{ij}^t} c_{ij}^t . v_{ij}^t \tag{2}$$

avec

$$\forall i \in \mathcal{A} \quad \textstyle\sum_{j \in \mathcal{R} \cup \{\emptyset\}} v_{ij}^t = 1 \tag{3}$$

$$\forall j \in \mathcal{R} \quad \textstyle\sum_{i \in \mathcal{A}} v_{ij}^t \leq 1 \tag{4}$$

Alors qu'intrinsèquement TSAP(t) est multi-objectif, nous le transformons en un problème d'optimisation mono-objectif (2), où le coût opérationnel et le critère de service sont tous les deux encapsulés dans la fonction de coût c_{ij}^t. La contrainte (3) assure que chaque taxi décide de son prochain nœud et la contrainte (4) assure qu'une demande ne peut pas être allouée à plus d'un taxi afin d'éviter que deux taxis ne répondent à la même demande. Nous pouvons noter que d'autres écritures non binaires sont possibles pour cette modélisation de TSAP(t) avec des solutions équivalentes mais une charge en mémoire et en temps de calcul différentes. Par exemple :

$$\min_{v_i^t} \quad \sum_{v_i^t} c_i^t(v_i^t) \tag{5}$$

$$\textbf{avec} \quad \textbf{AllDiff}(v_1^t, \ldots, v_{|\mathcal{A}|}^t) \tag{6}$$

où les variables prennent une valeur dans $\mathcal{R} \cup \{\emptyset\}$ et **AllDiff** vérifie que les décisions des taxis sont différentes (sans considérer le non choix d'une destination de l'origine d'une demande).

Classiquement, les systèmes de gestion de taxi résolvent ce problème soit à la main (par un opérateur humain, appelé dispatcheur) soit par un logiciel du marché, comme CPLEX. Bien que simple dans sa formulation, de tels problèmes appartiennent à la famille des problèmes NP-difficile. Les systèmes actuels centralisés sont capables de résoudre de tels problèmes en un temps raisonnable mais ne passe pas à l'échelle d'une ville avec des milliers de demandes et de taxis. Le problème doit être relâché en assignant par exemple des zones aux taxis ou interdisant de remettre en cause les allocations avec pour conséquence une allocation sous optimale des nouvelles demandes. Notre objectif est de concevoir un processus multiagent d'allocation décentralisé capable de rivaliser avec une solution centralisée, tout en palliant l'absence de connaissance sur les demandes et de communication globale.

3 Modèle multiagent

Dans cette section, nous optons pour une approche multiagent pour résoudre TSAP. Nous analysons différentes solutions de coordination qui se différencie selon la manière dont les décisions sont coordonnées et l'infrastructure de communication requise.

3.1 Comportement générique d'un taxi

Un taxi exécute en continu une boucle avec à chaque cycle les instructions suivantes :

1. lecture des messages entrants (concernant les autres taxis, les demandes, etc.) ;
2. mise à jour de ses croyances à propos des demandes et taxis ;
3. décision de la prochaine destination ;
4. déplacement vers la prochaine position en direction de la destination ;
5. émission de messages sur lui-même et transmission des messages concernant les demandes et autres taxis.

A chaque fois qu'un taxi disponible obtient de nouvelles informations à propos d'une demande (e.g. transmise par un autre taxi, ou reçu d'un portail centralisé), il met à jour ses croyances. Nous notons $KR(v_i, t)$ l'ensemble des demandes connues à l'instant t par v_i et $KT(v_i, t)$ l'ensemble des taxis connus par v_i à

l'instant t. Par extension, nous notons également $KR(C,t) = \bigcup_{v_i \in C} KR(v_i,t)$, l'ensemble des demandes connues par un ensemble de taxis C. Ainsi au sein d'un ensemble connecté, les taxis alignent leur connaissance. Avec ses croyances, un taxi doit décider de son déplacement (étape 3). Ce déplacement peut être : (a) en direction d'une source pour satisfaire une demande ; (b) en direction d'une autre position sur le réseau, pour par exemple anticiper l'arrivée de nouvelles demandes.

La décision est prise selon des critères concernant les demandes ($\kappa : \mathcal{A} \times \mathcal{R} \times T \rightarrow]0,1]$) qui assignent à chaque demande connue une utilité. Selon la nature et l'information utilisée dans ces critères, le comportement d'un agent peut être plus ou moins coopératif. Ces critères correspondent à la manière d'évaluer les coûts c_{ij}^t dans *PL en 0-1*-TSAP(t). La littérature sur le sujet montre principalement l'usage de critères fondés sur la distance ou le temps comme la demande la plus urgente ou la plus proche mais sans échange d'informations additionnelles entre les taxis [1, 3, 4, 8, 11, 13].

3.2 Processus d'allocation

Le processus d'allocation pour résoudre le *PL en 0-1*-TSAP(t) est à réaliser par les taxis eux mêmes ou éventuellement un dispatcheur. Selon la manière dont les décisions sont prises, le processus peut être plus ou moins coordonné. A présent, nous présentons les différentes alternatives de coordination que nous considérons :

(a) **Coordination centralisée (`c-alloc`)** : les agents reçoivent leurs ordres d'un dispatcheur (comme dans les approches classiques non multiagents) en utilisant une infrastructure globale de communication – i.e. à chaque temps t le dispatcheur collecte les informations pour calculer les coûts, puis résout directement *PL en 0-1*-TSAP(t) (voir section 2.2) et informe les taxis de la solution.

(b) **Coordination via un portail (`p-alloc`)** : Les taxis accèdent aux demandes via un portail et prennent eux mêmes leur décision mais se coordonnent en réservant les demandes sur ce même portail. Afin d'éviter les conflits, une demande ne peut être réservée qu'une seule fois et les taxis ne remettent pas en cause leur décision.

(c) **Coordination par un DCOP (`d-alloc`)** : L'environnement diffuse les demandes selon un modèle P2P et les agents prennent leur décision par eux-mêmes mais se coordonnent avec les agents du même ensemble connecté selon une approche DCOP afin d'éviter les conflits au sein des ensembles connectés mais pas entre eux (voir section 3.3).

La seule manière d'assurer une allocation optimale pour *PL en 0-1*-TSAP(t) est de centraliser la prise de décision avec une communication globale (cas a). Les alternatives peuvent conduire à une allocation sous-optimale et des conflits résultat d'une coordination incomplète et/ou d'informations manquantes.

3.3 Coordination par un DCOP

Dans cette section, nous nous focalisons sur l'implémentation d'une solution DCOP (**`d-alloc`**) pour résoudre *PL en 0-1*-TSAP(t)) à chaque fois qu'une nouvelle information est obtenue. Ce protocole est réalisé entre taxis interconnectés (ensembles connectés). Afin d'évaluer le coût pour atteindre l'origine d'une demande, les taxis partagent des informations sur les demandes et les autres taxis.

PL en 0-1-TSAP(t) est transformé en un DCOP (distributed constraint optimization problem) $\langle A, X, D, C \rangle$ [2]. Nous optons ici pour un encodage binaire de notre problème, l'obtention de bonnes performances ayant été prouvée pour des problèmes similaires [9, 10]. Les variables dans X sont toutes binaires et ainsi tous les domaines dans D sont réduits à $\{0,1\}$. Chaque taxi $v_i \in A$ a ses propres variables de décision (v_{ij}^t) correspondant à chaque demande $r_j \in KR(v_i,t)$ connues au temps t, plus la décision vide ($\emptyset$). L'ensemble de contraintes C contient les coûts de l'équation (2) et les contraintes (3) à (4). Comme indiqué précédemment, plusieurs fonctions de coût peuvent être considérées. Ici, les coûts unaires c_{ij}^t sont dérivés du critère de décision κ :

$$c_{ij}^t(v_{ij}) = \begin{cases} \dfrac{1}{\kappa(v_i, r_j, t)}, & \text{si } v_{ij}^t = 1 \\ 0, & \text{sinon} \end{cases} \quad (7)$$

La contrainte (3) impose que tous les agents fassent au moins une décision. Comme les agents peuvent décider de ne rien faire ($\emptyset$), cette contrainte est toujours vérifiée. La contrainte (8) est du type " Au moins un" qui est connue comme étant un *tractable higher order potential* (THOP), qui peut être évalué en temps linéaire [9] :

$$\mathbf{AMO}_j^t(v_{1j},\ldots,x_{nj}) = \begin{cases} 0, & \text{if } \sum_{i\in\{1\ldots n\}} v_{ij}^t \leq 1 \\ -\infty, & \text{otherwise} \end{cases} \quad (8)$$

Dans notre problème, pour chaque ensemble connecté il y a autant de contraintes **AMO** que de r_j demandes connus par les taxis de l'ensemble connecté considéré. Les taxis ne peuvent se coordonner qu'avec les taxis auxquels ils sont connectés, par conséquent tous les agents d'un même ensemble sont connectés aux mêmes contraintes **AMO**. La figure 2 illustre le graphe de contrainte d'un tel DCOP.

Au final, L'objectif de ce DCOP est le suivant :

$$\min_{v_{ij}^t} \sum_{v_{ij}^t} c_{ij}^t(v_{ij}^t) + \sum_{\substack{C\in CSet_{\mathcal{A}}(t) \\ r_j\in KR(C,t)}} \mathbf{AMO}_j^t(v_{1j}^t,\ldots,v_{|C|j})$$

Notons que deux ensembles connectés peuvent être informés d'une même demande selon leur positionnement respectifs vis-à-vis des sources. Dans un tel cas, chaque ensemble connecté doit satisfaire une contrainte **AtMostOne** différente pour la même demande. La conséquence est un potentiel conflit avec l'allocation de deux taxis différents pour la même demande. Sans communication global et donc la considération d'un seul ensemble connecté, de tels conflits ne peuvent être évités.

Il existe de nombreux algorithmes dans la littérature pour résoudre de tel DCOPs. L'encodage binaire que nous avons choisi a montré de bonnes performances sur des problèmes similaires en utilisant l'algorithme Binary MaxSum (BMS) [9]. Dans le cas d'un non encodage binaire les composants connectés du graphe de facteurs deviennent des arbres (une variable de décision par taxi, toutes connectés à une contrainte **AllDiff** pour chaque ensemble connecté). Ici, tout algorithme d'inférence comme DPOP ou le classique MaxSum donneront une solution optimale avec une charge de communication et un temps d'exécution faible Finalement, une autre alternative est de résoudre le problème d'allocation pour un ensemble connecté dans un agent unique qui collecte les coûts des agents auxquels il est connecté et calcule la solution optimale à *PL en 0-1*-TSAP(t) limité à son ensemble connecté puis informe de l'allocation optimale les autres taxis de l'ensemble connecté .

4 Évaluations

Dans cette section, nous évaluons les performances des trois stratégies coordonnées décrites en section 3.2, dans des configurations différentes (nombre de taxis, rayon de communication), en utilisant une simulation à temps discret.

4.1 Configuration des expérimentations

La carte de la ville de Saint-Étienne a été choisie pour la simulation (cf. Figure 3). Le réseau est composé de 503 arcs avec 4645 points. Le nombre de points par arc dépend de l'information enregistrée dans le fichier Open Street Map (OSM) décrivant la ville de Saint-Etienne. La distance totale du réseau est de 198 km et la distance moyenne entre deux points est de 43 mètres. La position des sources S_1, S_2 et S_3 est fixée. Ce n'est pas un des paramètres étudié. Les sources sont réparties sur la carte pour pouvoir comparer différents types de courses : longue ou courte distance. La distance entre les sources (calculées à partir de leur coordonnées longitude et latitude en kilomètres) est approximativement de 0.8, 1.5 et 2. La durée d'un trajet est respectivement 19, 36 et 48 cycles. Les taxis se déplacent d'un point à un autre sur le même arc à chaque cycle de simulation. La vitesse moyenne des taxis est de 30 km/h. Un cycle est équivalent à 5 seconds.

A chaque cycle de simulation, 0 à 2 demandes sont générées aléatoirement. Pour chacune d'elles, les sources origine et destination sont générées aléatoirement en suivant la loi espace-temps : les demandes sont concentrées sur la source S_1 de l'espace. Un cycle sur deux (choisi uniformément et de manière aléatoire) l'origine d'une nouvelle demande est S_1 et est quelconque sinon. La concentration des demandes varie également au cours du temps (tous les 100 cycles de simulation, un pic de demandes est généré). Le nombre de demandes augmente de manière aléatoire entre 0 et 10 et leur origine est fixée à S_1. La fenêtre temporelle des demandes est définie comme suit : tw_{min} est initiée avec la valeur du cycle courant et tw_{max} est initié avec la valeur minimale à laquelle est ajoutée une valeur aléatoire dans $[70, 140]$.

Comme mentionné précédemment, résoudre *PL en 0-1*-TSAP(t) en utilisant **`c-alloc`**, **`p-alloc`** ou **`d-alloc`** nécessite de définir les fonctions de coût (c_{ij}^t's) en s'appuyant sur le critère de décision (κ). Afin de pouvoir comparer les résultats, nous utilisons les mêmes critères pour

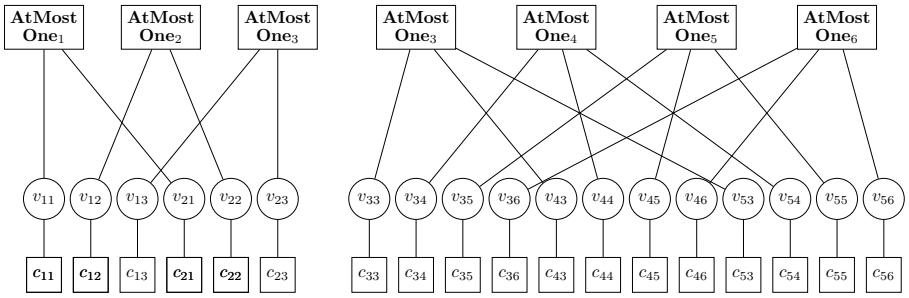


FIGURE 2 – Exemple de modèle DCOP pour le TSAP(t) de la figure 1, avec deux composants connectés(un pour chaque ensemble connecté), les deux étant concernés par la demande r_3.

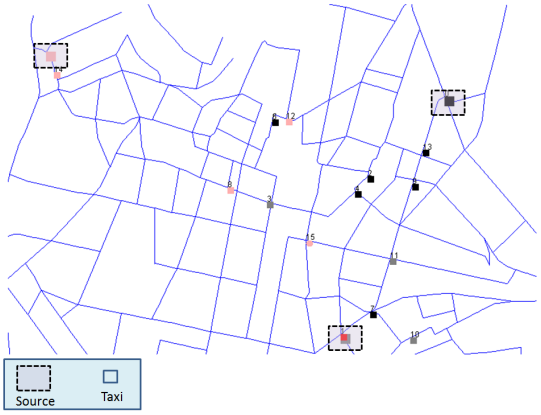


FIGURE 3 – Plan de Saint-Etienne utilisé dans les simulations

chaque stratégie : un critère coopératif fondé sur le temps et l'espace, noté $\kappa_{\mathbf{coop}}$,

$$\kappa_{\mathbf{coop}}(v_i, r_j, t) = \alpha.\kappa_{\mathbf{space}}(v_i, r_j, t) + (1-\alpha).\kappa_{\mathbf{time}}(v_i, r_j, t) \quad (9)$$

avec $\alpha \in [0, 1]$ (ici 0.5), $\kappa_{\mathbf{space}}$ critère préférant les demandes qui ont le moins de chance d'être servies du fait du peu de taxis proches et $\kappa_{\mathbf{time}}$ critère préférant les demandes qui ont une durée d'attente prédite plus élevée (voir Annexe A).

4.2 Résultats et Analyse

En utilisant les configurations décrites ci-dessus, nous exécutons plusieurs simulations pour une période de 3000 cycles de simulation (env. 4h temps simulé). Regardons en premier la qualité de service (**qos**) des différentes stratégies, représentée dans la Figure 4a, qui est le pourcentage de demandes satisfaites. Une demande est satisfaite si un taxi est à la source origine de la demande avant la fin de la fenêtre temporelle de la demande avec l'objectif de réaliser la course pour le client. Notre objectif est d'obtenir au moins 90%. Cet objectif est atteint par **c-alloc** et **p-alloc** avec une flotte composée d'au moins 15 taxis. **d-alloc** nécessite 16 taxis et un rayon de communication de 300 ou plus pour la même performance.

La figure 4b représente le gain résultant pour les différentes stratégies, i.e. la différence entre la distance moyenne d'une course (moyenne de la distance parcourue dans les cycles de simulation pour un taxi transportant des passagers) et la moyenne de la distance parcourue (la distance moyenne parcourue dans les cycles de simulation pour les taxis libres). Le plus haut est la valeur, meilleur est le résultat puisqu'un taxi rapporte plus à la compagnie. **c-alloc** et **p-alloc** atteignent exactement le même niveau alors que **d-alloc** reste en dessous quel que soit le rayon de communication. Cependant, à partir d'un rayon de 250 les résultats sont équivalents, ce qui signifie que ce rayon est suffisant pour une bonne coordination.

La figure 4c présente le temps moyen passé par les clients à attendre que leurs demandes soient prises en compte. Nous pouvons remarquer que les stratégies décentralisées avec un rayon plus grand que 250 obtiennent de meilleures performances que les stratégies centralisées ou basées sur un portail – elles atteignent l'équivalent mais avec une moins bonne qualité de service.

La figure 5 montre le temps nécessaire pour réaliser les simulations sur une seule machine. Le nombre de cycles est identique mais la durée des calculs modifie la durée des cycles. De manière évidente, **p-alloc** nécessite moins de calcul puisqu'une fois les demandes allouées, elles ne sont pas révisées. Ainsi la **vérification** des demandes est peu déclenchée alors que les autres approches vérifient le statut de la demande à

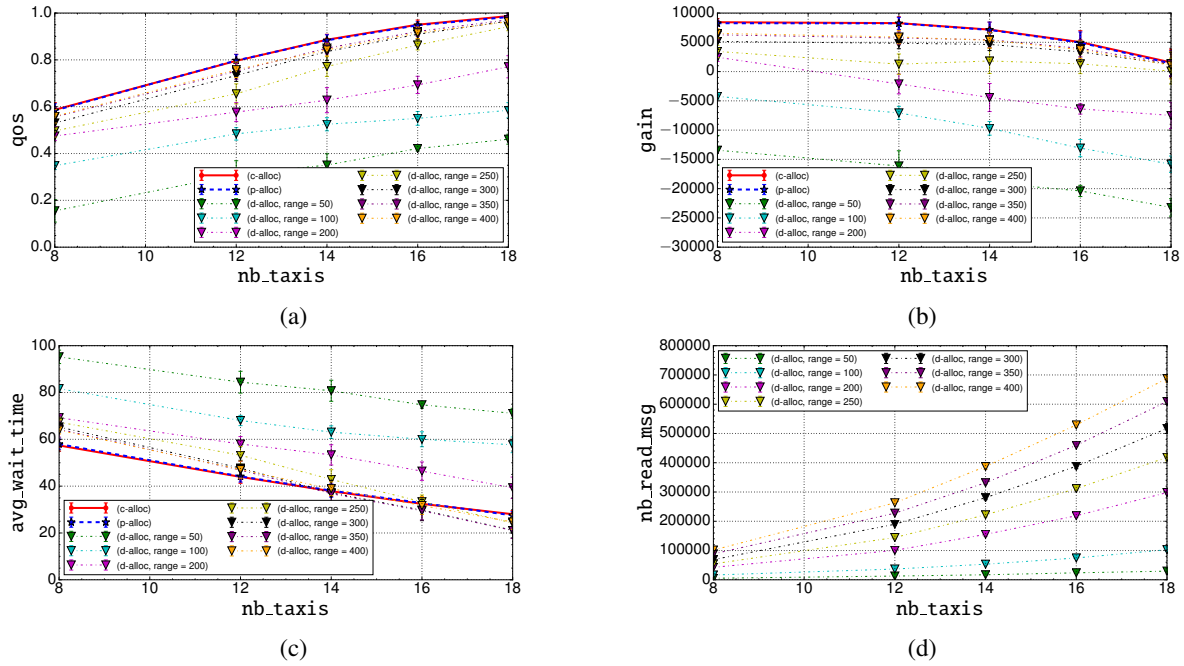


FIGURE 4 – Qualité de service (4a), gain (4b), temps moyen d'attente d'un client (4c) et nombre de messages lus par des taxis (4d) pour les différentes stratégies avec des rayons de commmnunication pour la stratégie décentralisée en utilisant des interactions P2P, pour un nombre croissant de taxis.

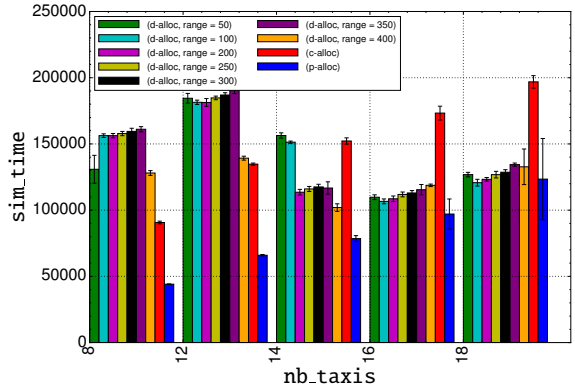


FIGURE 5 – Temps de simulation (en ms) pour les différentes stratégies, pour un nombre croissant de taxis.

chaque pas de simulation tant qu'elles ne sont pas fermées. Puisqu'il résout implicitement *PL en 0-1*-TSAP(t) à chaque cycle de simulation, **`c-alloc`** induit un coup de calcul élevé. Dans le cas décentralisé **`d-alloc`**, le temps de la résolution dépend du rayon et du nombre de taxis qui impactent le nombre d'ensembles connectés et donc le nombre de sous-problèmes à résoudre. Avec peu de taxis et un rayon faible, peu d'ensembles connectés co-existent au même moment, et donc il y a donc moins de calcul nécessaires pour résoudre les sous problèmes.

Pour résumer, une stratégie fondée sur l'utilisation d'un portail se comporte aussi bien qu'une stratégie centralisant le processus d'allocation. Ceci signifie que résoudre *PL en 0-1*-TSAP(t) n'est pas nécessaire pour atteindre de très bonne solutions à TSAP(t). Laisser les taxis prendre leurs décisions eux-même et les coordonner par un simple portail est suffisant. Cependant, alors que **`c-alloc`** et **`p-alloc`** nécessitent plus de messages (linéaire en nombre de taxis), les deux souffrent d'un goulot de centralisation, qui peut mener à l'échec. Au contraire, **`d-alloc`** nécessite moins de messages comme représenté sur la figure 4d, parce que l'infrastructure P2P nécessite beaucoup de relais de messages. En fait dans les réseaux VANET, les messages sont diffusés impliquant un nombre important de traitement de messages à chaque nœud. Cependant, puisque les agents se coordonnent localement, perdre une information majeure est moins probable et impacte moins le système dans sa globalité.

5 Travaux liés

Dans [11], les auteurs analysent les activité d'un dispatcheur et mettent en évidence l'impossibi-

lité de satisfaire à la fois les taxis (critère du taux d'utilisation des taxis) et les clients (critère du temps d'attente à la fois dans les périodes de pointe ou pas). Leur conclusion est que les solutions décentralisées multiagents peuvent améliorer la performance de ces solutions de transport. Notre proposition évalue en plus les bénéfices d'un processus d'allocation centralisé.

La centralisation du processus d'allocation avec un dispatcheur automatique est assez courante dans les approches multiagents [3,5,8,13]. Dans [8], un dispatcheur central met en œuvre trois stratégies d'allocation. Elles utilisent plus ou moins d'information temps réel sur l'ordonnancement des taxis résultat d'interactions avec des agents taxis qui ne prennent pas part au processus de décision. L'utilisation d'un dispatcheur central avec des agents taxis qui donnent l'information sur leur état courant est aussi proposée dans [5, 13]. Dans [3], une approche d'allocation fondée sur un marché est proposée. Cette solution décentralisée est traitée entre des agents clients et des agents fournisseurs du service de taxi. La décision d'allocation mise en œuvre par le fournisseur du service de taxi est cependant centralisée. Ces propositions évaluent leur propre stratégie d'allocation sans comparaison avec des alternatives décentralisées.

Dans [12], un protocole de coordination décentralisé entre des agents taxis disponibles est proposée. Un sous ensemble de demandes est affecté à un sous-ensemble (de taille équivalente) d'agents taxi disponibles. Les agent taxis échangent en P2P leur évaluation des demandes afin d'arriver à un consensus. Dans [4], les auteurs proposent un processus de négociation fondé sur une estimation du coût et du temps de transport, l'allocation est calculée par un agent dédié à chacune des demandes client. Ces solutions agents illustrent la faisabilité des allocations décentralisées dans un environnement dynamique. Cependant, il s'agit de solutions *ad-hoc* qui ne s'appuient pas sur des protocoles prouvés à la différence du domaine DCOP [2]. De plus ces solutions sont fondées sur une hypothèse de connexion totale entre les agents, ce qui implique des coûts importants.

Dans la limite de notre connaissance, la seule proposition qui compare allocation centralisée et décentralisée est [7] qui propose une alternative hybride. L'un des objectifs de ce travail est de diviser l'espace de recherche avec des agents nœuds stations qui ne peuvent interagir qu'avec les agents taxis dans leur voisinage direct. Ces derniers planifient des propositions pour les demandes client et les agents nœuds stations appliquent des politiques de filtrage. Les expérimentations montrent des meilleurs résultats qu'une solution pleinement décentralisée avec des communications complètes des demandes aux agents taxi. Une explication possible est que le partitionnement de l'espace de recherche améliore la qualité des résultats et réduit le temps de calcul.

La limitation de l'espace de recherche est implicite dans des solutions d'allocation décentralisées. Dans les solution centralisées, le réseau est décomposé en espace où l'allocation est réalisée et est étendue à l'espace le plus proche si aucune allocation n'est trouvée [1, 13]. Dans une solution décentralisée, la même approche est proposée par [7] avec les nœuds stations pour lesquels le nombre d'agents taxis est limité de manière arbitraire [4, 12]. Dans notre proposition, la limitation l'espace de recherche n'est pas définie *a priori* mais est le résultat de la connexion des véhicules aux sources où les demandes sont émises.

6 Conclusions

Dans ce travail, nous avons étudiés l'impact de la décentralisation dans le processus d'allocation de demandes à une flotte de taxis autonomes. Nous avons modélisé le problème d'allocation de taxis et proposé trois stratégies pour le résoudre : (a) coordination centralisé par un dispatcheur (avec communication globale), (b) coordination via un portail (avec communication globale et locale), (c) coordination par un DCOP (avec communication locale).

Comparée aux approches centralisées (dispatcheur ou portail), la coordination par un DCOP montre des résultats proches de l'optimal en termes de qualité de service et de meilleurs résultats concernant la satisfaction des clients (i.e. le temps d'attente). De nombreuses perspectives sont envisageables notamment concernant la robustesse à la perte de messages. En effet, nous envisageons que des mécanismes locaux seront plus adaptés à un environnement bruité puisque la communication en P2P implique de la redondance. Une autre perspective est l'étude d'autres critères d'évaluation. En fait, ce n'est pas le sujet principal de cet article mais le critère d'évaluation des demandes à un grand impact sur la qualité des résultats et doit être plus étudiés en prenant en compte à la fois le point de vue des clients et des taxis. Enfin, le couplage de ces approches dans un système hybride pourrait égale-

ment donner des solutions pertinentes.

Remerciements

Ces travaux ont été en partie financés par Renault Innovation.

Références

[1] Aamena Alshamsi, Sherief Abdallah, and Iyad Rahwan. Multiagent self-organization for a taxi dispatch system. In *8th international conference on autonomous agents and multiagent systems*, pages 21–28. Citeseer, 2009.

[2] J. Cerquides, A. Farinelli, P. Meseguer, and S. D. Ramchurn. A tutorial on optimization for multi-agent systems. *The Computer Journal*, 57(6) :799–824, 2014.

[3] Malcolm Egan and Michal Jakob. Market mechanism design for profitable on-demand transport services. *Transportation Research Part B : Methodological*, 89 :178–195, 2016.

[4] Andrey Glaschenko, Anton Ivaschenko, George Rzevski, and Petr Skobelev. Multi-agent real time scheduling system for taxi companies. In *8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2009), Budapest, Hungary*, pages 29–36, 2009.

[5] Josep Maria Salanova Grau and Miquel Angel Estrada Romeu. Agent based modelling for simulating taxi services. *Procedia Computer Science*, 52 :902–907, 2015.

[6] Mark ET Horn. Fleet scheduling and dispatching for demand-responsive passenger services. *Transportation Research Part C : Emerging Technologies*, 10(1) :35–63, 2002.

[7] Xu Jin and Luo Jie. A study of multi-agent based model for urban intelligent transport systems. *International Journal of Advancements in Computing Technology*, 4(6) :126–134, 2012.

[8] Michał Maciejewski and Kai Nagel. The influence of multi-agent cooperation on the efficiency of taxi dispatching. In *International Conference on Parallel Processing and Applied Mathematics*, pages 751–760. Springer, 2013.

[9] M. Pujol-Gonzalez, J. Cerquides, A. Farinelli, P. Meseguer, and J. A. Rodriguez-Aguilar. Efficient inter-team task allocation in robocup rescue. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 413–421, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[10] M. Pujol-Gonzalez, J. Cerquides, P. Meseguer, J. A. Rodríguez-Aguilar, and M. Tambe. Engineering the decentralized coordination of uavs with limited communication range. In *15th Conference of the Spanish Association for Artificial Intelligence, CAEPIA 2013*, pages 199–208. Springer, 2013.

[11] Darshan Santani, Rajesh Krishna Balan, and C Jason Woodard. Spatio-temporal efficiency in a taxi dispatch system. In *6th International Conference on Mobile Systems, Applications, and Services, MobiSys*, 2008.

[12] Kiam Tian Seow, Nam Hai Dang, and Der-Horng Lee. A collaborative multiagent taxi-dispatch system. *IEEE Transactions on Automation Science and Engineering*, 7(3) :607–616, 2010.

[13] Wen Shen and Cristina Lopes. Managing autonomous mobility on demand systems for better passenger experience. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 20–35. Springer, 2015.

[14] Mahdi Zargayouna, Flavien Balbo, and Khadim Ndiaye. Generic model for resource allocation in transportation. application to urban parking management. *Transportation Research Part C : Emerging Technologies*, 71 :538–554, 2016.

A Évaluation des requêtes

$$\kappa^{\mathbf{coop}}_{\mathbf{dist}}(v_i, r_j, t) = \frac{1}{closerFree(v_i, r_j, t) + closerRiding(v_i, r_j, t) + 1}$$

où, $closerFree$ décompte le nombre de taxis libresplus proches de r_j que v_i, et $closerRiding$ décompte le nombre de taxis en course plus proche de r_j que v_i, tout en considérant la distance à parcourir pour déposer leur client actuel à sa destination :

$$closerFree(v_i, r_j, t) = \sum_{v_k \in KT(v_i,t)} free(v_k, t).closer(pos(v_k), pos(v_i), origin(r_j), t)$$

$$closerRiding(v_i, r_j, t) = \sum_{v_k \in KT(v_i,t)} (1 - free(v_k, t)).$$
$$closer(dest(v_k), pos(r_j), t + travel(dest(v_k, t), pos(v_i, t), t))$$

où

$$closer(x, y, z, t) = \begin{cases} 1, & \text{si } x \neq y \text{ et} \\ & travel(pos(x,t), pos(z,t), t) \\ & \leq travel(pos(y,t), pos(z,t), t) \\ 0, & \text{sinon} \end{cases}$$

$$\kappa^{\mathbf{coop}}_{\mathbf{time}}(v_i, r_j, t) = \frac{free(v_i, t)}{\sum_{r_k \in KR(v_i,t)} worst(pos(v_i, t), r_k, r_j, t) + 1} - \frac{(1 - free(v_i, t))}{\sum_{r_k \in KR(v_i,t)} worst(pos(dest(r_j), t), r_k, r_j, t) + 1}$$

avec :

$$worst(x, r_1, r_2, t) = \begin{cases} 1, & \text{if } ((t + travel(x, pos(origin(r_1), t), t)) - tw_{min}(r_1)) \geq \\ & \quad ((t + travel(v, pos(origin(r_2), t), t)) - tw_{min}(r_2)) \\ 0, & \text{otherwise} \end{cases}$$

Les SMA multi-niveaux comme cadre de modélisation et de simulation en épidémiologie

Sébastien Picault[a, b] Yu-Lin Huang[a] Vianney Sicard[a] François Beaudeau[a] Pauline Ezanno[a]

[a]Bioepar, INRA, Oniris, La Chantrerie, 44307 Nantes, France
prenom.nom@oniris-nantes.fr

[b]Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL (équipe SMAC)

Résumé

Les modèles épidémiologiques, pour proposer des mesures de maîtrise efficaces, gagnent sans cesse en niveau de détail, et ce à des échelles d'observations multiples, de l'individu aux politiques publiques. Cet élan est freiné par la diversité des paradigmes de modélisation, l'absence de méthodologie logicielle pour l'implémentation de codes toujours plus complexes, et parfois l'existence d'hypothèses implicites. Nous proposons d'utiliser une démarche de modélisation multi-agents multi-niveaux pour intégrer les méthodes existantes au sein d'un cadre générique, forcer une séparation des aspects déclaratifs et procéduraux mais aussi entre domaines d'expertise, et réduire la part de code dévolue aux modélisateurs. Nous illustrons cette démarche par une application sur la maîtrise de la fièvre Q chez les bovins.

Mots-clés : *Modélisation multi-niveaux ; simulation multi-agents ; épidémiologie*

Abstract

To recommend efficient control measures, epidemiological models incorporate ever-finer details, from individual diversity to public policies, which involve several observation scales. Difficulties in this approach arise from the variety of modelling paradigms, an increased complexity of simulation programs not yet couterbalanced by software engineering methods, and sometimes the existence of implicit assumptions. We propose to use a multi-level agent-based modelling approach to integrate existing methods within a common interface, provide a separation between procedural and declarative concerns but also between expertise fields, and reduce the amount of code left to the designers' responsibility. We illustrate this approach through an application to Q fever control in cattle.

Keywords: *Multilevel modelling ; Multi-agent-based simulation ; Epidemiology*

1 Introduction

Les travaux que nous présentons s'inscrivent dans le cadre du projet MIMHES [1] qui vise à comprendre les mécanismes de propagation des maladies infectieuses des animaux de production afin de proposer et d'évaluer des stratégies de maîtrise. Dans ce projet, plusieurs maladies enzootiques ont été étudiées à plusieurs échelles d'observation. Divers modèles ont été développés pour chacun des pathogènes cibles, à l'échelle intra-troupeau ou inter-troupeaux, utilisant des modèles à compartiments ou centrés invidivus, et ce dans des langages de programmation variés (Scilab, C++, Python, R). Ils ont fait la preuve de leur pertinence pour rendre compte des observations de terrain et aidé à concevoir des recommandations en matière de stratégies de maîtrise de ces maladies. Néanmoins, ils manquent de généricité d'un point de vue d'ingénierie logicielle, de sorte que l'exploration de nouvelles hypothèses ou scénarios demande en général d'importants efforts de recodage. Par ailleurs, le choix du paradigme de modélisation (compartiments ou individus) contraint fortement l'architecture du simulateur, alors qu'il serait souvent profitable de pouvoir remplacer des modèles à compartiments par des modèles centrés individus ou vice-versa.

Cette situation est tout à fait représentative des difficultés qui se présentent en modélisation épidémiologique [24], et nous sert donc dans cet article de cas concret pour tester des solutions logicielles nouvelles. Nous pensons notamment que les simulations multi-agents sont un bon candidat pour fournir une interface homogène à plusieurs paradigmes de modélisation. Plus précisément, les concepts et techniques développés dans le cadre des simulations multi-agents multi-niveaux apparaissent parti-

1. « Multi-scale modelling, from animal Intra-Host to Metapopulation, of mechanisms of pathogen spread to Evaluate control Strategies » http://www6.inra.fr/mihmes

culièrement pertinentes pour les modèles épidémiologiques en raison de la nature intrinsèquement multi-niveaux des processus infectieux, des techniques d'élevage, et des échanges d'animaux.

L'article est organisé de la façon suivante. Dans la section 2, nous rappelons les paradigmes principaux utilisés en modélisation épidémiologique. Les principes de conception et le *framework* proposés en réponse aux problèmes cidessus sont présentés dans la section 3, où nous insistons notamment sur la nécessité d'une séparation forte des divers aspects des modèles. Enfin (section 4), nous illustrons notre approche par une application à la maîtrise de la fièvre Q dans les troupeaux bovins laitiers.

2 Aperçu des paradigmes de modélisation en épidémiologie

2.1 Équations et compartiments

Depuis les travaux fondateurs de Kermack et McKendrick [15], l'approche classique en modélisation épidémiologique consiste à partitionner la population en plusieurs *compartiments* représentant chacun le nombre d'invididus partageant un même état. Cette hypothèse forte permet de ne s'intéresser qu'aux flux d'individus passant d'un compartiment à un autre. Un tel modèle est ordinairement représenté au moyen d'un diagramme de flux, par exemple dans le fameux modèle « SIR » (figure 1) constitué de trois états de santé : *sensible* (S), où les individus peuvent s'infecter suite à des contacts, directs ou indirects, avec des individus infectieux ; *infectieux* (I), où les individus sont contagieux ; *rétabli* (R), où les individus ont perdu leur contagiosité et ne peuvent plus être réinfectés. Les trois variables S, I, R correspondant au nombre d'individus dans chaque compartiment sont contrôlées par un système d'équations différentielles ordinaires reposant directement sur les taux de transition entre compartiments. Elles peuvent être calculées soit de façon continue et déterministe, soit de manière discrète et stochastique, après avoir transformé les taux de transition en probabilités, et en appliquant alors un tirage aléatoire selon une loi multinomiale.

Les modèles compartimentaux, en plus de fournir un éclairage analytique sur le système, sont assez adaptables. Les taux d'entrée et de sortie peuvent représenter des dynamiques démographiques. Les compartiments peuvent être subdivisés en agrégats plus spécifiques, par exemple pour assurer un découpage spatial ou pour modéliser des classes d'âge. Il est également possible de faire coexister plusieurs espèces, comme c'est le cas pour les maladies à vecteurs. De plus, la contamination indirecte par l'environnement peut être explicitement représentée par un ou plusieurs compartiments dédiés [17].

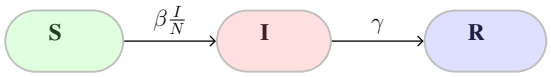


FIGURE 1 – Diagramme de flux pour le modèle classique SIR. Les nœuds sont des compartiments donnant le nombre d'individus dans chaque état de santé. Les arcs sont étiquetés par les taux de transition.

Cependant, ce paradigme est moins adapté pour rendre compte de la diversité rencontrée dans les paramètres biologiques (comme la sensibilité à l'infection), ou pour intégrer des aspects comportementaux, comme les mouvements saisonniers ou les mesures de prévention et de maîtrise. Lorsque plusieurs problématiques se superposent aux états de santé (classes d'âges, conduite de l'élevage, décision de vaccination...), les seules possibilités laissées par cette approche consistent soit à subdiviser encore et encore les compartiments (e.g. [17]), ce qui finit par ressembler beaucoup à un modèle centré individus, soit à complexifier le formalisme [23].

2.2 Les modèles centrés individus

En raison de ces limitations, un nombre croissant de travaux font appel aux modèles centrés individus (*Individual-Based Models*, IBM) qui permettent une prise en compte explicite de la diversité des *états individuels* (e.g. [9]), ou même aux modèles multi-agents (*Agent-Based Models*, ABM), offrant dans son principe une diversité des *comportements individuels* et des *interactions* entre individus (e.g. [2, 26]). Les IBM/ABM offrent l'avantage d'une extensibilité quasiment infinie, puisque l'introduction de nouvelles hypothèses se traduit principalement par l'ajout de nouvelles familles d'agents ou de nouveaux comportements. La dynamique du modèle peut être scrutée en détail, et les entités du domaine sont représentées de façon très directe par les entités computationnelles. Il est également plus simple de représenter des processus multifactoriels et de comprendre la nature des mécanismes causaux à l'œuvre dans les systèmes épidémiologiques [18].

En contrepartie, le coût computationnel est nettement plus élevé. Non seulement un IBM est en général plus lent qu'un modèle à compartiments, mais il faut également un plus grand nombre de répétitions, car le niveau de détail se paie de paramètres supplémentaires, qui conduisent donc à une analyse de sensibilité plus poussée. En outre, la facilité à changer des hypothèses mène naturellement à la tentation de multiplier le nombre de scénarios à évaluer afin de comparer des mesures de maîtrise ou des politiques publiques.

D'autre part, tous ces travaux s'intéressent à des cas pratiques et des pathogènes particuliers. Les outils ainsi construits sont pour la plupart *ad hoc* avec dans le meilleur des cas une modélisation objet qui s'abstrait quelque peu du cadre applicatif [27]. Quelques-uns, à l'opposé, s'appuient sur une plateforme de simulation multi-agents, comme GAMA [2] ou NetLogo [26], ce qui constitue un progrès dans la mesure où nombre d'algorithmes (notamment en matière d'ordonnancement des agents) ne sont plus à la charge du modélisateur. Toutefois ces plateformes ne sont en rien spécifiques à l'épidémiologie et ne constituent donc pas en elles-mêmes une solution systématique. Force est donc de constater qu'il n'existe à notre connaissance aucune tentative, entre ces deux extrêmes, pour faire des SMA un paradigme de modélisation épidémiologique à part entière, doté d'une méthodologie générale ainsi que d'algorithmes et d'architectures réutilisables.

2.3 Vers les simulations multi-niveaux

De nouveaux problèmes se posent lorsque l'on passe à l'échelle régionale ou nationale : les dynamiques spatiales ne peuvent plus être ignorées, et il devient nécessaire de coupler des modèles développés pour des échelles différentes [4].

Une approche classique pour manipuler des modèles épidémiologiques à une échelle régionale repose sur le concept écologique de métapopulation [12]. Une métapopulation est un système de populations locales interconnectées, vivant dans des zones plus ou moins isolées, chacune dotée de sa propre dynamique démographique, voire infectieuse. Les contacts entre populations locales reposent sur les relations de voisinage, les mouvements, ou le transport de pathogènes par des hôtes ou par le vent. Cette méthode réduit le coût computationnel qu'entraînerait la stricte application d'une approche IBM, aux dépens d'une modélisation plus grossière de la dynamique des sous-populations. Elle est très répandue en épidémiologie humaine, où elle permet de traiter des populations et des zones à grande échelle sur la base de structures de contact assez bien décrites (horaires de travail, moyens de transport notamment). De plus, les métapopulations permettent un traitement déterministe comme stochastique. En revanche, comparées à des simulations ABM équivalentes, ces modèles tendent à surestimer la propagation des infections [1, 14].

D'un autre côté, le coût computationnel de simulations multi-agents classiques à cette échelle est considérablement élevé. À moins de faire appel à la fois à des plateformes massivement parallèles, des hypothèses épidémiologiques très simples, et un haut niveau d'optimisation logicielle [22], manipuler des millions ou milliards d'agents est un problème dur, d'autant que de nombreuses répétitions sont nécessaires pour comparer des scénarios de façon significative.

Comme on peut en juger d'après cette diversité des paradigmes de modélisation, il n'existe clairement pas de solution ultime qui serait adaptée à n'importe quelle situation. Au contraire, un cadre de modélisation polyvalent doit être capable de fournir des moyens, méthodologiques aussi bien que logiciels, pour combiner toutes ces méthodes de façon appropriée à chaque contexte de mise en œuvre.

3 Une méthode et un framework de modélisation multi-niveaux

Nous partons de l'hypothèse que les modèles doivent être d'emblée *conçus comme multi-niveaux*, plutôt que de construire par exemple un modèle intra-troupeau et tenter de l'étendre à l'échelle régionale, ou au contraire de chercher à partir d'un modèle de métapopulation en rajoutant à chaque troupeau des caractéristiques plus fines. Ce parti-pris, loin d'être utopique, peut être réalisé en s'appuyant sur une forte modularité en particulier en ce qui concerne 1) la *structure* des modèles, pour laquelle une approche multi-agents multi-niveaux semble bien adaptée, et 2) une décomposition explicite de tous les processus à l'œuvre dans le système (dynamique infectieuse, gestion du troupeau, échanges commerciaux...). Dans cette section, nous présentons les principes de conception et l'architecture logicielle que nous proposons en réponse à ces problématiques.

3.1 Modularité structurelle : modélisation multi-niveaux

En épidémiologie animale, la possibilité de réifier des entités intermédiaires entre l'animal et la population (groupes d'âge, troupeaux...), ainsi que des zones spatiales correspondantes (enclos, fermes, pâturages...) constitue un point clef pour évaluer les mécanismes fins de mesures de maîtrise. Or, le nombre de travaux de recherche sur les simulations multi-agents multi-niveaux est en forte croissance depuis quelques années. Ces méthodes consistent à faire appel à des agents pour réifier des niveaux d'organisation, d'observation ou d'échelle au sein même de la simulation en cours. Néanmoins la plupart de ces contributions sont conçues pour des domaines d'applications spécifiques et s'appuient sur des méthodes *ad hoc*.

Aussi, nous avons choisi de nous appuyer sur un des rares méta-modèles multi-niveaux génériques existants (avec principalement [8, 10, 20]), à savoir PADAWAN [25], qui offre des caractéristiques adaptées aux besoins en épidémiologie computationnelle. En particulier, ce méta-modèle impose une *forte séparation entre les aspects déclaratifs et procéduraux*, à travers notamment l'indépendance entre les comportements (spécifiés comme règles d'interaction génériques) et les agents amenés à les réaliser. Les agents et les environnements peuvent être associés via deux relations : la *situation* (qui permet aux agents d'interagir au sein d'un ou plusieurs environnements) et *l'encapsulation* (un agent peut « contenir » un environnement pour héberger d'autres agents). L'utilisation conjointe de ces deux relations permet de définir la notion *d'hébergement* (a héberge b si b est situé dans un environnement encapsulé par a).

Ainsi, un SMA multi-niveaux est construit comme *combinaison d'une structure* (l'architecture et l'organisation des agents et des environnements emboîtés) *et d'une fonction* (une description explicite et intelligible des processus qui se déroulent dans le système). Comme les agents peuvent représenter n'importe quelle sorte d'entité, ils fournissent une interface homogène mais polymorphe pour intégrer de multiples paradigmes de modélisation.

3.2 Modularité fonctionnelle : gestion des connaissances

Nous estimons par ailleurs que les connaissances du domaine introduites dans un modèle (paramètres, hypothèses, processus, données...) doivent être segmentées en fonction de la diversité des points de vue, des objectifs et des expertises. Nous nous situons en cela dans le même esprit que les travaux de [6, 7] dans le cadre de modèles à compartiments, qui distinguent par exemple les problématiques liées au processus infectieux, à la gestion des espèces et à la distribution spatiale. Nous souhaitons aller plus loin dans cette voie en permettant l'introduction de mécanismes *a priori* quelconques, y compris extérieurs à l'épidémiologie *stricto sensu*, comme la conduite des élevages, les aspects économiques de la prise de décision par l'éleveur, le vétérinaire ou les pouvoirs publics, les effets physiologiques des traitements, etc. La première tâche dans la conception d'un modèle consiste donc à identifier l'ensemble de ces domaines et de leurs processus.

Pour permettre à des experts épidémiologistes d'être pleinement acteurs de la modélisation, il est en outre nécessaire que le cadre conceptuel qui leur est proposé soit *aussi peu intrusif que possible* par rapport aux outils utilisés habituellement, de façon à rendre plus précises les méthodes existantes sans demander de gros efforts d'adaptation. Nous proposons donc les principes suivants :

1. *Améliorer les formalismes existants pour réduire leurs ambiguïtés, sans les rendre abscons*. Par exemple, le protocole ODD proposé par [13] (principalement pour l'écologie) est un premier pas vers l'explicitation des connaissances des experts, mais c'est avant tout un cadre textuel qui reste ambigu [3]. Au contraire, l'utilisation de formalismes très riches issus soit de la physique multi-échelles, comme les *Bond Graphs* [11], soit de la biologie moléculaire, comme SBGN [16], serait inapproprié au regard des préoccupations épidémiologiques car d'une complexité disproportionnée.
2. *Fournir des guides méthodologiques pour apporter à des problèmes typiques une réponse pertinente et standardisée*, comme le font les *Design Patterns* en Génie Logiciel.
3. *Automatiser autant que possible les tâches centrales du processus de simulation*, de façon à n'imposer aux utilisateurs finaux le développement que de portions de code spécifiques et de petite taille.

Afin de maintenir une proximité avec les formalismes existants, nous proposons de transformer le diagramme de flux utilisé dans les

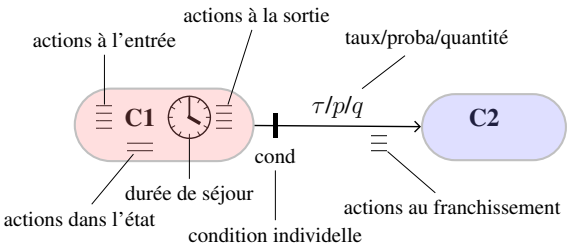


FIGURE 2 – Extension du diagramme de flux sous la forme d'une machine à états.

modèles à compartiments en une véritable machine à états finis. Pour ce faire, nous considérons que les nœuds et les arcs peuvent être dotés d'informations complémentaires (fig. 2). Les états peuvent recevoir des caractéristiques facultatives : 1) une *distribution des durées de séjour*, qui spécifie combien de temps un individu est susceptible de rester dans l'état courant, et qui peut être particulièrement utile pour décrire des effets démographiques ; 2) des *actions* exécutées à l'entrée, durant le séjour, ou à la sortie de l'état, par exemple pour gérer l'excrétion dans les états infectieux. En plus de leur étiquette qui représente soit un taux, soit une probabilité, soit un nombre absolu d'individus, on peut ajouter aux transitions : 1) *des conditions de franchissement*, pour déterminer quels agents sont autorisés à migrer de l'état source vers l'état de destination, et 2) des *actions* exécutées par les individus qui franchissent l'arc, i.e. après avoir quitté l'état source et avant d'entrer dans l'état de destination.

Ces ajouts correspondent en fait à des informations « métier » qui, étant absentes des modèles spécifiés par un diagramme de flux classique, sont d'ordinaire codées directement lors de l'implémentation. Déplacés vers l'amont dans la phase conception, ces éléments contribuent à une vision plus précise du modèle dans son ensemble, et peuvent être inclus dans un processus de génération de code. De plus, ce diagramme de machine à états peut être utilisé indifféremment pour décrire ce qui se passe dans un modèle à compartiments (déterministe ou stochastique) ou centré individus.

Toutes les informations concernant les processus d'un modèle, les machines à état correspondants avec leurs états, transitions, conditions, actions, durées et les paramètres de taux, probabilités ou quantités sont spécifiées au moyen d'un fichier de configuration au format YAML (choisi pour sa grande lisibilité), qui est traité pour générer l'architecture de simulation mais peut également servir à produire une documentation technique, des figures, inclure des commentaires sur les sources utilisées et les hypothèses sous-jacentes, etc.

3.3 Architecture du framework EMuLSion

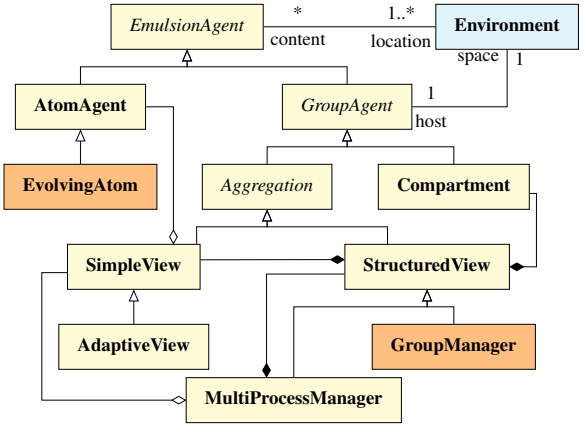


FIGURE 3 – Diagramme de classes de la hiérarchie des agents multi-niveaux dans le framework EMuLSion. Le comportement des agents de classes à fond orangé est contrôlé par une machine à états.

Ces principes de conception ont été implémentés et expérimentés à travers un framework réalisé en Python, appelé « EMuLSion [2] ». Dans ce framework, un modèle est composé de processus (séquences d'actions sur les agents), certains d'entre eux étant dirigés par des machines à états. Ils s'appuient sur un ensemble de paramètres clairement décrits comme les taux, probabilités, quantités, distributions, etc. donnés en valeurs, en intervalles, ou exprimés en fonction d'autres paramètres. Ces informations sont analysées au moyen d'une bibliothèque de calcul symbolique (Sympy) et la cohérence du modèle est vérifiée automatiquement.

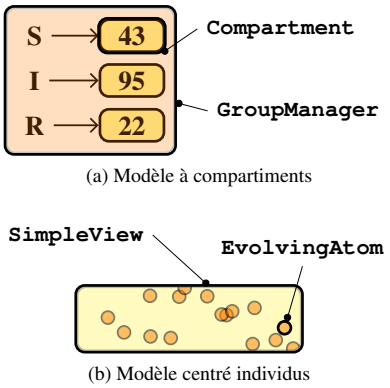
Les classes des agents utilisés dans EMuLSion sont présentées sur la figure 3. Tous les agents multi-niveaux sont situés dans au moins un environnement. Ils se répartissent en deux grandes catégories : les *atomes* et les *groupes*. La classe `AtomAgent` représente des « individus » (e.g. les animaux). Sa sous-classe `EvolvingAtom` correspond aux individus dont le comportement est régi par des machines à états. De leur côté, les groupes d'agents peuvent encapsuler un environnement local où d'autres agents peuvent se situer. Les groupes sont composés eux-mêmes de deux familles : les *compartiments* et les

2. pour *Epidemiological MUlti-Level SimulatION framework*

agrégats. La classe `Compartment` est utilisée lorsque les individus sont suffisamment homogènes pour être décrits par une simple quantité. Une `Aggregation` fournit au contraire une *vue* sur des individus ou d'autres groupes, i.e. une représentation où les agents sont rassemblés en fonction de variables identifiées au préalable comme pertinentes selon un point de vue donné, comme l'état de santé, le groupe d'âge, etc.

Parmi ces agrégats, on distingue d'abord les agents `SimpleView` qui hébergent des invidus (atomes) et gèrent l'ordonnancement de leurs comportements. Une `AdaptiveView` détecte en outre les individus dont les variables clefs ont une valeur différente des autres, et demandent à leur hôte de les placer dans un groupe adéquat. Ensuite, un agent `StructuredView` a pour fonction d'associer des agents `SimpleView` ou `Compartment` aux valeurs possibles de variables clefs. Par exemple, dans un modèle SIR, la variable `health_state` peut prendre trois valeurs (S, I, R), et se trouve donc associée à trois agents `Compartment` ou à trois agents `SimpleView`. Le `GroupManager` fait de même, mais dispose en outre d'une machine à états pour déterminer quels individus doivent changer de valeur pour ces variables clefs. Enfin, un agent `MultiProcessManager` peut gérer plusieurs processus dans une même simulation. Il s'appuie au minimum sur un agent `SimpleView` pour superviser tous les individus, et sur un agent (en fait, souvent plusieurs) `StructuredView` ou `GroupManager` pour gérer les différents processus du modèle.

On notera incidemment que cette architecture s'appuie délibérément sur les quatre *Design Patterns* pour les SMA multi-niveaux définis dans [19]. Le `Compartment` correspond à une agrégation destructive d'agents micro en un agent macro (pattern ZOOM). Les agents `SimpleView` et `StructuredView` ne font que représenter au niveau macro l'état des agents micro qu'ils hébergent (pattern VUE). À l'opposé, le `GroupManager`, en charge d'une machine à états propre à une problématique, dépouille les agents micro qu'il héberge de leur autonomie comportementale au regard de ce processus précis et se charge de les piloter (pattern `Marionnettiste`). Le dernier pattern (`Cohabitation`) correspond à une situation où un `MultiProcessManager` en héberge d'autres, par exemple une métapopulation et ses sous-populations : dans ce cas certain processus macroscopiques (échanges commerciaux) peuvent agir comme un forçage sur les agents microscopiques, sans pour autant priver ces derniers de leur autonomie.



(a) Modèle à compartiments

(b) Modèle centré individus

FIGURE 4 – (a) Structure d'un modèle à compartiments. Le GroupManager utilise une machine à états pour déterminer les flux entre compartiments. (b) Structure d'une simulation IBM. Les EvolvingAtoms possèdent leur propre machine à états et sont hébergés par un agent SimpleView.

3.4 Modes d'utilisation

Cette architecture qui repose sur les capacités de composition des agents permet en pratique de représenter les paradigmes de modélisation épidémiologique classiques. Les *modèles à compartiments* sont construits en utilisant des agents `Compartment` hébergés par un `GroupManager`, lui-même doté d'une machine à états qui détermine les états de santé et leurs transitions (fig. 4a). Les agents `Compartment` peuvent adopter un comportement déterministe ou stochastique à la demande. Les *modèles centrés individus* sont composés d'agents `EvolvingAtom`, chacun doté d'une (ou plusieurs) machine(s) à états, et hébergés par un agent `SimpleView` qui ordonnance leur activité (fig. 4b). Enfin, les *métapopulations* peuvent être réalisées au moyen d'un agent `StructuredView` ou `MultiProcessManager` contenant plusieurs autres agents construits selon l'une ou l'autre des approches précédentes, avec en plus des processus dédiés à la description des structures de contact.

Mais d'autres solutions peuvent être élaborées, en particulier en regroupant les individus en fonction de leur état, pour chacun des aspects modélisés. L'intérêt principal de tels regroupements est de fournir un accès direct à des individus jugés similaires selon un certain point de vue. En particulier, pour déter-

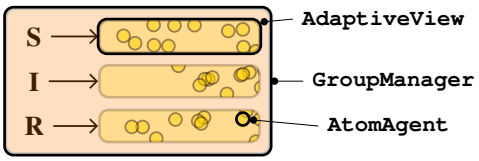


FIGURE 5 – Structure du regroupement d'individus au sein de vues adaptatives. Les atomes sont répartis en fonction de leurs états communs. Le GroupManager est doté d'une machine à états pour piloter les changements des atomes.

miner lesquels doivent changer d'état, il suffit dans une approche stochastique d'un seul tirage selon une loi multinomiale par groupe, au lieu d'une épreuve de Bernoulli par individu, d'où une efficacité accrue. Pour réaliser ces regroupements dynamiques, la première étape consiste à construire un agent `GroupManager` équipé d'une machine à états pour un aspect du modèle, tel que le processus infectieux qui impacte la variable `health_state` (fig. 5). Chaque valeur possible de `health_state` est associée à un agent `AdaptiveView`, contenant des `AtomAgent`s. Les atomes changent d'état de santé selon la machine à états du `GroupManager`, mais si un processus extérieur affecte leur variable `health_state`, par exemple un traitement qui transforme des « I » en « R », l'agent `AdaptiveView` en charge de l'état « I » détecte ces modifications et demande au `GroupManager` de déplacer ces atomes modifiés à la bonne place (chez l'agent `AdaptiveView` en charge de l'état « R »).

L'agent `MultiProcessManager` se charge de coordonner plusieurs structures analogues (cf. section suivante, fig. 7). Il reçoit la liste des processus affectant les individus ou les groupes, ainsi que les variables associées à chaque regroupement. Si un processus est lié à une machine à états, il fait automatiquement appel à un `GroupManager` (structuré comme indiqué ci-dessus), sinon il fait de même avec un simple `StructuredView`. Chaque individu est donc accessible à partir d'une structure globale (`SimpleView`) et de chacun des regroupements.

Ce framework a été testé en détail sur plusieurs variations de modèles théoriques du type SIR, pour vérifier que tous les cas d'usages décrits ci-dessus permettaient de reproduire des résultats équivalents. Dans la section qui suit, nous montrons une application à un cas réel, la fièvre Q.

4 Application à la fièvre Q chez les bovins

Afin de donner une illustration concrète de notre approche, nous allons montrer comment elle a été appliquée à une des maladies étudiées dans le projet MIHMES : la fièvre Q, une zoonose qui touche principalement les ruminants. Deux modèles centrés individus ont été développés auparavant pour cette maladie dans le cadre des bovins et servent de référence : l'un pour la propagation intra-troupeau [9] (en R), l'autre pour la propagation inter-troupeaux [21] (en Python).

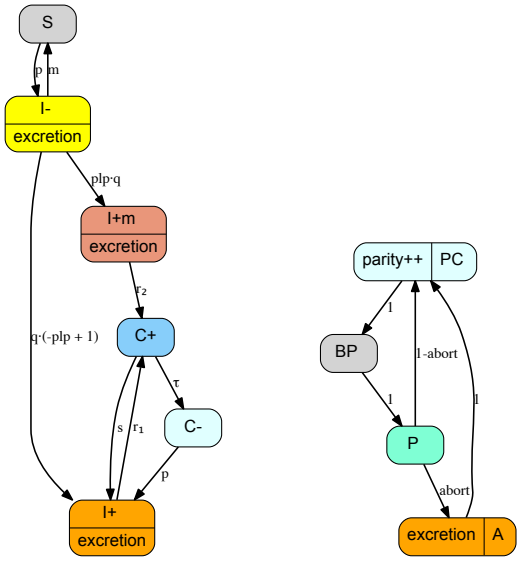
4.1 Le modèle fièvre Q

Notre implémentation du modèle de la fièvre Q dans les troupeaux bovins laitiers s'appuie sur les travaux antérieurs [9] avec 6 états de santé : sensible (S), infectieux sans réponse immunitaire (I–), infectieux avec réponse immunitaire (I+) voire excrétion bactérienne dans le lait (I+m), et porteur avec (C+) ou sans (C–) anticorps. La contamination se fait par dépôt de bactéries dans l'environnement. Par ailleurs, seules les femelles adultes sont prises en compte. Elles sont soumises à une conduite d'élevage (ou « cycle de vie ») adaptée à la production laitière, au cours de laquelle les vaches sont inséminées, deviennent gestantes (état P), vêlent (PC) ou avortent (A) selon leur état de santé, et attendent la gestation suivante (BP). Le vêlage et l'avortement entraînent un haut niveau d'excrétion bactérienne. Ces processus sont représentés sur la figure 6.

La simulation s'effectue en temps discret. À chaque pas de temps, les processus impliqués au niveau du troupeau sont les suivants : 1) mise à jour des bactéries présentes dans l'environnement (décroissance exponentielle) ; 2) réforme des vaches (retrait du troupeau) en fonction de leur parité (nombre de vêlages antérieurs) ; 3) remplacement pour introduire de nouveaux animaux dans le troupeau ; 4) infection pilotée par la machine à états affectant l'état de santé (fig. 6a) ; 5) conduite de l'élevage pilotée par celle affectant le cycle de vie (fig. 6b) ; 6) mise à jour du regroupement des animaux par parité.

4.2 Implémentation, tests et validation

L'implémentation de ce modèle pourrait suivre une approche IBM classique, mais pour bénéficier du regroupement adaptatif des individus,



(a) Processus infectieux (b) Cycle de vie

FIGURE 6 – Machines à états pour le processus infectieux et le cycle de vie des vaches adultes. Les actions à l'entrée/au cours des états sont indiquées. Les valeurs des arcs sont des probabilités.

l'architecture la plus appropriée consiste à définir d'une part une classse pour les individus (`QfeverCow`, dérivée de `AtomAgent`), et une pour le troupeau (`QfeverHerd`, sous-classe de `MultiProcessManager`).

Les processus impliqués dans la fièvre Q conduisent à regrouper les individus selon trois critères : la parité (pour la réforme), le cycle de vie et l'état de santé (ces deux dernières variables étant contrôlées par une machine à états). La quantité de bactéries excrétées par les animaux infectieux est en outre plus élevée dans les semaines qui suivent le vêlage, aussi est-il plus efficace d'utiliser une variable booléenne supplémentaire (`recent_calving`) pour contrôler le regroupement lié au processus infectieux. Cela pourrait multiplier les agents `AdaptiveView` en raison de la combinatoire des valeurs possibles de ces variables, mais en pratique le `GroupManager` n'en crée que lorsque c'est nécessaire pour héberger des atomes. L'architecture du SMA qui résulte de ces contraintes est présentée sur la figure 7.

La figure 8 montre que ce modèle implémenté dans EMuLSion reproduit les résultats du modèle de référence [9], c'est-à-dire une coïncidence des moyennes et des percentiles des prin-

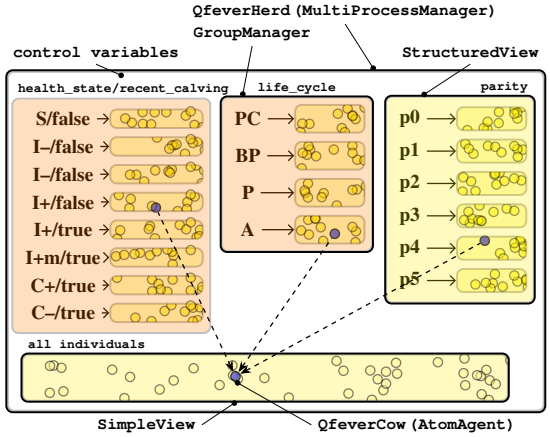


FIGURE 7 – Structure du modèle intra-troupeau de la fièvre Q. Les individus sont rassemblés en fonction des valeurs des variables clefs de chaque aspect du modèle. On accède aux individus (e.g. celui représenté en bleu, dans l'état de santé I^+, le cycle de vie A et de parité 4) soit par aspect, soit par l'agent `SimpleView` hébergeant tous les atomes.

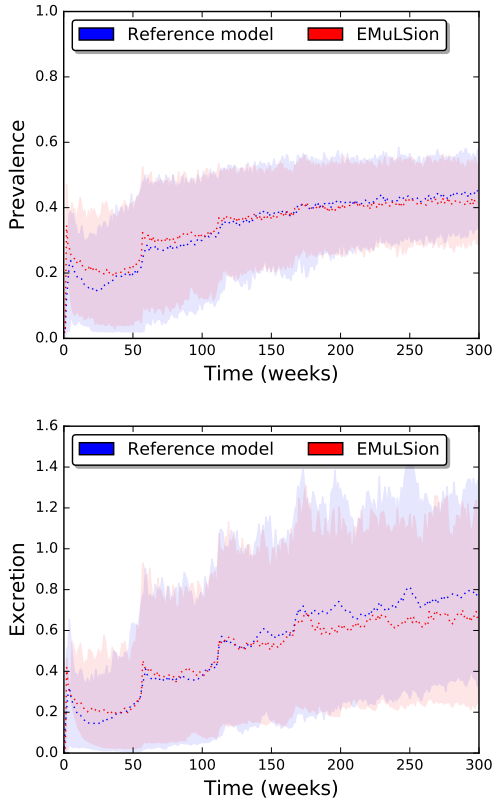


FIGURE 8 – Évolution dans le temps de la prévalence et de l'excrétion dans l'environnement, en moyenne et percentiles sur 200 simulations, pour le modèle de référence [9] et l'implémentation dans EMuLSion pour la fièvre Q.

cipales variables de sortie pertinentes d'un point de vue épidémiologique (ici, l'excrétion dans l'environnement et la prévalence, i.e. la proportion d'animaux infectieux dans le troupeau), calculées pour un nombre suffisamment important de répétitions (ici 200).

En outre, la séparation entre les aspects procéduraux et déclaratifs, mais aussi entre les divers processus du modèle, a permis de faciliter grandement la formulation et la vérification d'hypothèses alternatives, en particulier l'étude des simplifications possibles du modèle intra-troupeau pour ne garder que les mécanismes essentiels à l'explication causale de la propagation de la maladie, avant de passer à la contamination inter-troupeaux. Ce travail exploratoire fera l'objet d'une publication prochaine dans le domaine épidémiologique.

5 Conclusion et perspectives

Nous avons présenté une approche générique pour la conception et l'implémentation de modèles épidémiologiques, basée sur une structure d'hébergement d'agents répartis dans plusieurs niveaux d'organisation. Notre méthode offre une façon homogène d'intégrer facilement des paradigmes de modélisation classiques comme les modèles à compartiments ou centrés individus. Il s'appuie pour cela sur une séparation forte entre les aspects procéduraux et déclaratifs des modèles, ainsi qu'entre les divers processus et domaines d'expertise nécessaires. Ainsi, les hypothèses sous-jacentes sont toutes explicitées sous une forme intelligible et révisable, et peuvent être traitées par un moteur de simulation générique et d'une fiabilité accrue.

Ce framework a été d'ores et déjà mis en œuvre pour refondre le modèle intra-troupeau de la fièvre Q et tester quelles hypothèses peuvent être simplifiées, de façon à construire un modèle inter-troupeaux efficace et pertinent. Les modèles de référence, difficilement modifiables, comptent environ 1 000 lignes de code R (intra-troupeau) et 2 500 de code Python (inter-troupeaux). La version actuelle du framework compte environ 3 200 lignes de code Python, ce qui permet de réduire les développements spécifiques à la fièvre Q à seulement 250 lignes. Le fichier YAML qui décrit les machines à états, les processus et les paramètres, est composé d'environ 300 entrées clef-valeur (en incluant les commentaires, descriptions et sources bibliographiques pour chacun d'eux).

Les travaux en cours portent sur l'intégration de ce modèle dans une simulation multi-troupeaux, au moyen d'un agent `MultiProcessManager` pour représenter la métapopulation de troupeaux. La propagation de la fièvre Q entre troupeaux est causée par la diffusion des bactéries sous forme d'aérosols ainsi que par l'introduction d'animaux infectés au cours des échanges commerciaux. Cela n'impacte donc que deux processus du modèle intra-troupeau : la mise à jour des quantités de bactéries dans l'environnement (transport par le vent sur la base de données météorologiques), et le remplacement des animaux (à partir des données de mouvements par ventes et achats). Le volume de code à écrire est donc lui aussi très faible.

Les autres modèles développés au cours du projet MIHMES vont faire l'objet de travaux similaires pour assurer leur durabilité et faciliter le développement d'extensions, la révision d'hypothèses, et l'étude de nouveaux scénarios de maîtrise. Plus généralement, ce travail est un premier pas vers l'élaboration collective d'un cadre commun de modélisation qui permette une comparaison précise et factuelle des modèles épidémiologiques. La possibilité de rendre toutes les hypothèses explicites et de décrire finement tous les processus et interactions impliquées dans un système, ainsi que la confiance dans un moteur de simulation à la fois générique et flexible, sont une exigence cruciale pour la reproductibilité scientifique. Dans ce but, EMuLSion sera à terme distribué en *open source*. Par ailleurs, la capacité à utiliser indifféremment et de façon transparente l'un ou l'autre des multiples paradigmes existants ouvre la voie à l'exploration de techniques visant à basculer automatiquement de l'un à l'autre, comme suggéré dans [5].

Dans une perspective plus large, nous pensons aussi que la réduction des durées de conception, d'implémentation et de validation de nouveaux modèles contribue de façon essentielle à une réactivité accrue pour répondre aux urgences épidémiologiques (grippe humaine ou aviaire, maladies transmises par les moustiques), enjeu majeur face à la multiplication de situations inédites consécutives aux changements globaux en cours et à venir.

Remerciements

Ces travaux sont réalisés dans le cadre du projet MIHMES, co-financé par l'ANR (investis-

sements d'avenir ANR-10-BINF-07) et par le Fonds Européen de Développement Régional (FEDER) des Pays-de-la-Loire. Ils bénéficient également d'un financement du département Santé Animale de l'INRA.

Références

[1] M. Ajelli, B. Gonçalves, D. Balcan, V. Colizza, H. Hu, J.J. Ramasco, S. Merler, and A. Vespignani. Comparing large-scale computational approaches to epidemic modeling : Agent-based versus structured metapopulation models. *BMC Infectious Diseases*, 10(1), 2010.

[2] E. Amouroux, S. Desvaux, and A. Drogoul. Towards virtual epidemiology : An agent-based approach to the modeling of H5N1 propagation and persistence in north-vietnam. In *11th Pacific Rim Int. Conf. on Multi-Agents (PRIMA)*, volume 5357 of *LNCS*, pages 26–33. Springer, 2008.

[3] E. Amouroux, B. Gaudou, S. Desvaux, and A. Drogoul. O.D.D. : A promising but incomplete formalism for individual-based model specification. In *IEEE RIVF Int. Conf. on Computing and Communication Technologies*. IEEE, 2010.

[4] G. Beaunée, E. Vergu, and P. Ezanno. Modelling of paratuberculosis spread between dairy cattle farms at a regional scale. *Veterinary Research*, 46(1), 2015.

[5] G.V. Bobashev, D.M. Goedecke, F. Yu, and J.M. Epstein. A hybrid epidemic model : Combining the advantages of agent-based and equation-based approaches. *Winter Simulation Conference*, 2007.

[6] T.M.A. Bui, S. Stinckwich, M. Ziane, B. Roche, and T.V. Ho. KENDRICK : A domain specific language and platform for mathematical epidemiological modelling. In *IEEE RIVF Int. Conf. on Computing and Communication Technologies*, pages 132–137. IEEE, 2015.

[7] T.M.A. Bui, M. Ziane, S. Stinckwich, T.V. Ho, B. Roche, and N. Papoulias. Separation of concerns in epidemiological modelling. In *15th Int. Conf. on Modularity*, pages 196–200. ACM, 2016.

[8] B. Camus, C. Bourjot, and V. Chevrier. Multi-level modeling as a society of interacting models. In *Agent-Directed Simulation Symposium (in Spring-Sim)*. SCS/ACM, 2013.

[9] A. Courcoul, H. Monod, M. Nielen, D. Klinkenberg, L. Hogerwerf, F. Beaudeau, and E. Vergu. Modelling the effect of heterogeneity of shedding on the within herd *Coxiella burnetii* spread and identification of key parameters by sensitivity analysis. *J. Theor. Biol.*, 284(1) :130–141, 2011.

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

[11] V. Díaz-Zuccarini and C. Pichardo-Almarza. On the formalization of multi-scale and multi-science processes for integrative biology. *Interface Focus*, 1(3) :426–437, 2011.

[12] B. Grenfell and J. Harwood. (meta)population dynamics of infectious diseases. *Trends in Ecology & Evolution*, 12(10) :395–399, 1997.

[13] V. Grimm and et al. A standard protocol for describing individual-based and agent-based models. *Ecological Modelling*, 198(1–2) :115–126, 2006.

[14] M.J. Keeling, L. Danon, M.C. Vernon, and T.A. House. Individual identity and movement networks for disease metapopulations. *PNAS*, 107(19) :8866–8870, 2010.

[15] W.O. Kermack and A.G. McKendrick. A contribution to the mathematical theory of epidemics. *Proc. R. Soc.*, (A115) :700–721, 1927.

[16] N. Le Novere and et al. The systems biology graphical notation. *Nat. Biotech.*, 27(8) :735–741, 2009.

[17] C. Marcé, P. Ezanno, H. Seegers, D. Pfeiffer, and C. Fourichon. Predicting fadeout versus persistence of paratuberculosis in a dairy cattle herd for management and control purposes : a modelling study. *Veterinary Research*, 42(1) :36, 2011.

[18] B.D.L. Marshall and S. Galea. Formalizing the role of agent-based modeling in causal inference and epidemiology. *Am. J. Epidemiology*, 181(2) :92–99, 2014.

[19] P. Mathieu, G. Morvan, and S. Picault. Simulations multi-agents multi-niveaux : quatre patterns de conception. In *24e Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, pages 117–126. Cépaduès, 2016. (présentation courte).

[20] G. Morvan, A. Veremme, and D. Dupont. IRM4MLS : The influence reaction model for multi-level simulation. In *Multi-Agent-Based Simulation XI*, volume 6532 of *LNCS*, pages 16–27. Springer Nature, 2011.

[21] P. Pandit, T. Hoch, P. Ezanno, F. Beaudeau, and E. Vergu. Spread of *Coxiella burnetii* between dairy cattle herds in an enzootic region : modelling contributions of airborne transmission and trade. *Veterinary Research*, 47(1), 2016.

[22] J. Parker and J.M. Epstein. A distributed platform for global-scale agent-based models of disease transmission. *ACM Trans. Model. Comput. Simul.*, pages 2 :1–2 :25, 2011.

[23] N. Perra, D. Balcan, B. Gonçalves, and A. Vespignani. Towards a characterization of behavior-disease models. *PLoS ONE*, 6(8) :e23084, 2011.

[24] S. Picault, Y.-L. Huang, V. Sicard, and P. Ezanno. Enhancing sustainability of complex epidemiological models through a generic multilevel agent-based approach. In *26th Int. Joint Conf. on Artificial Intelligence (IJCAI)*. AAAI, 2017. (in press).

[25] S. Picault and P. Mathieu. An interaction-oriented model for multi-scale simulation. In *22nd Int. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 332–337, 2011.

[26] J. Robins, S. Bogen, A. Francis, A. Westhoek, A. Kanarek, S. Lenhart, and S. Eda. Agent-based model for Johne's disease dynamics in a dairy herd. *Veterinary Research*, 46(1), 2015.

[27] B. Roche, J.-F. Guégan, and F. Bousquet. Multi-agent systems in epidemiology : a first step for computational biology in the study of vector-borne disease transmission. *BMC Bioinformatics*, 9, 2008.

Simuler l'activité humaine en combinant un système multi-agent avec des approches statistiques basées sur les « enquêtes emploi du temps »

Q. Reynaud[a] quentin.reynaud@limsi.fr
F. Sempé[b] sempe.francois@gmail.com
Y. Haradji[c] yvon.haradji@edf.fr
N. Sabouret [a] nicolas.sabouret@limsi.fr

[a]LIMSI, CNRS, Univ. Paris-Sud, Université Paris-Saclay, Bât 508, Campus Universitaire, 91405 Orsay, France,
[b]François Sempé AE, Paris, France

[c]EDF R&D, EDF Lab Paris-Saclay, 7 Boulevard Gaspard Monge, 91120 Palaiseau, France

Résumé

En simulation multi-agent (SMA), l'une des principales difficultés est de disposer de données pour calibrer le modèle. Dans cet article, nous nous intéressons à la simulation multi-agent de l'activité humaine : notre objectif est de pouvoir calibrer et quantifier la représentativité des activités simulées. Nous proposons pour cela d'utiliser des données statistiques bien formalisées provenant d'enquêtes sur la vie quotidienne : les enquêtes « emploi du temps ». Ces enquêtes sont conçues pour décrire et reproduire l'activité humaine d'une journée à un niveau macroscopique (à l'échelle de la population d'un pays). Nous proposons une nouvelle méthode de génération de l'activité humaine qui est à la fois statistiquement juste à un niveau agrégé et individuellement réaliste, grâce à la modélisation multi-agent d'individus dotés de capacités réactive, adaptative, et collaborative.

Mots-clés *: Simulation multi-agent du comportement humain, enquêtes emploi du temps*

Abstract

Multi-agent based simulations of human activity often lack data to calibrate and qualify the representativeness of the simulated activities. In this paper, we will show that massive statistical investigations such as "time-use surveys" allow us to obtain this type of data. While these surveys are mostly used to validate the realism of human activity on a macroscopic level (population scale), we offer a new method of human activity generation that combines these methods with a multi-agent system. This enables the simulated activities to gain realism on a microscopic level (individual scale) by giving them reactivity, adaptability, coordination and coherence through extended periods of time.

Keywords*: Multi-Agent Based Simulation, Human Behavior Simulation, Time Use Survey.*

1 Introduction

Dans le domaine de la simulation multi-agent de l'activité humaine, on peut identifier trois types d'approches : 1) les approches utilisant des comportements « scriptés » dans lesquelles les agents suivent un comportement prédéfini [1-2] ; 2) les approches orientées vers des comportements autonomes et/ou réactifs, par exemple dans [3-4] , dans lesquelles il faut implémenter les connaissances expertes sur l'activité des individus; 3) les approches hybrides visant à combiner les deux précédentes [5-6]. Dans cette dernière catégorie, l'idée générale est de définir des « scénarios d'activité » servant de cadre comportemental de « haut-niveau » aux agents, et de rajouter un mécanisme autonome de sélection de l'action qui gère les comportements à un niveau plus élémentaire.

La difficulté de ce type de modèles est d'arriver à calibrer les scénarios d'activités. En effet, les comportements sont autonomes et les éléments du scénario définissent des « bornes » à ne pas dépasser. Il est donc nécessaire de calibrer ces bornes. Une façon de les définir est de s'appuyer sur des données sur l'activité humaine. Ces données peuvent être obtenues de différentes manières. Une possibilité est d'utiliser la « simulation participative », qui consiste à confronter des humains à la simulation de leur propre comportement [7-8]. Cependant cette méthode ne permet pas de passer à l'échelle : elle nécessite une modélisation au cas par cas des ménages et demande un trop grand nombre d'entretiens individuels.

1.1 Enquêtes emploi du temps

Afin de permettre le passage à l'échelle, nous pouvons utiliser une approche statistique. Des enquêtes statistiques nationales de type « enquête emploi du temps » (« *Time-Use Survey* ») [9] ont été menées dans différents pays afin de caractériser l'emploi du temps des gens. Dans ces enquêtes, les participants remplissent des questionnaires (ou « carnets ») portant sur l'emploi de leur temps au cours de la journée. Ils sont constitués d'un ensemble de plages horaires (typiquement, toutes les 10 minutes pendant 24h). Les participants indiquent quelle activité ils ont réalisée lors de chaque plage horaire. Ensuite des experts regroupent les activités déclarées en catégories. Un intérêt majeur de ces enquêtes est qu'elles respectent un formalisme mondialement partagé[1].

Ces enquêtes sont couramment utilisées par des approches statistiques [10-14] pour reproduire l'activité humaine quotidienne. Le réalisme des activités produites est mesuré en termes de proximité statistique avec les comportements observés à l'échelle de la population étudiée.

Toutefois, comme nous allons le montrer dans cet article, ce réalisme statistique n'est pas suffisant à la reproduction d'activités individuellement réalistes. C'est pourquoi nous proposons de le coupler avec un modèle SMA hybride afin de reproduire des comportements réalistes au niveau macroscopique (niveau de la population) et au niveau microscopique (niveau des individus).

1.2 Contexte

Le contexte de nos travaux est celui de la réduction de la consommation énergétique. Nous nous intéressons à la simulation d'activités humaines réalistes dans le but d'inférer les consommations électriques des ménages et d'étudier l'effet de politiques de réduction de consommation. Dans ce cadre, le réalisme macroscopique est indispensable afin de simuler des consommations agrégées réalistes et de mesurer l'impact de phénomènes globaux (par exemple les pics de consommation). Mais le réalisme microscopique est également indispensable à la simulation de consommations, puisque c'est l'action au niveau de chaque ménage que nous voulons étudier. La modélisation d'individus adaptatifs est par ailleurs nécessaire afin de simuler des modifications de comportements liées à des événements, des tarifs, de nouveaux types de consommation, etc.

1.3 Notre proposition

Nous proposons une nouvelle méthode de simulation de l'activité humaine basée sur un système multi-agent hybride doté d'un modèle de comportements prescrits de « haut-niveau » calibrés grâce à des données statistiques, ainsi que d'un mécanisme autonome de sélection de l'action, plus « bas-niveau ».

Dans cet article nous présenterons en détail la manière dont le modèle de comportements prescrits est calibré. Nous rappelons brièvement le fonctionnement du SMA dans les sections 3.1 et 3.2. Les détails sont

[1] http://www.timeuse.org/

disponibles dans [15-16].

Les avantages de notre méthode sont : 1) les données statistiques permettent au SMA de produire des activités réalistes à grande échelle (> 1000 ménages). 2) Le SMA permet de simuler des activités plus réalistes à un niveau individuel, grâce à la simulation d'individus réactifs, adaptatifs, et collaboratifs.

De plus, notre méthode permet de produire des scénarios d'activités de plusieurs jours, ce que les méthodes statistiques ne permettent pas.

2 Méthodes statistiques utilisant les enquêtes emploi du temps

2.1 Etat de l'art

Les enquêtes emploi du temps sont des enquêtes sur la vie quotidienne dans lesquelles les personnes interrogées retranscrivent leur journée dans des « carnets ». On distingue deux tendances dans l'utilisation de ces enquêtes pour la reproduction d'activités humaines : les approches « *top-down* » et « *bottom-up* ».

2.1.1 Approches « top-down »

Les données provenant des enquêtes sont utilisées pour calculer une matrice qui détermine, à chaque moment de la journée, la probabilité (pour un type d'individu donné) de passer d'une activité à une autre. La génération des activités est ensuite effectuée à chaque pas de temps, en sélectionnant l'activité suivante selon cette table de transition [10-12].

Ces approches sont limitées par trois facteurs [14]. 1) Elles nécessitent l'accès aux données brutes des enquêtes, qui ne sont pas toujours disponibles librement selon les pays. 2) Elles manquent de précision quant à la durée des différentes activités (ces approches se concentrent sur les transitions entre les activités, et non sur leurs durées effectives). 3) Il n'est pas possible de gérer la coordination entre les membres d'un même ménage : les matrices de changement d'activités ne prennent pas l'environnement en compte.

2.1.2 Approches « bottom-up »

Les données des enquêtes sont utilisées pour calculer la durée moyenne et la répartition des activités pendant la journée. A partir de ces informations, les approches « bottom-up » construisent des emplois du temps en sélectionnant itérativement l'activité suivante grâce à une distribution probabiliste des durées. [13] montre que cette approche ne nécessite pas un accès aux données brutes des enquêtes. Les seules informations nécessaires sont : la durée moyenne (et les écarts types associés) de chaque activité, ainsi que le pourcentage d'individus qui adoptent une activité spécifique à un moment donné.

[14] améliore cette méthode en ajoutant le concept de « comportement de routine ». Un emploi du temps partiel est généré, uniquement constitué des activités « dormir », « travailler », « aller à l'école », ainsi que celles liées aux repas et à l'hygiène. Les autres comportements sont ajoutés à cet emploi du temps de manière à combler les « vides ». Le traitement différent des activités de routine (les premières à être placées), permet une coordination basique des individus (mais limitée à ces activités).

Ces méthodes génèrent des durées d'activité plus précises que les approches « top-down », mais sont toujours confrontées à des limites.

2.2 Les limites des approches statistiques

Toutes ces méthodes statistiques visent à reproduire des activités réalistes au niveau macroscopique. Cependant, ces méthodes souffrent de 4 limites. Elles ne permettent pas de produire des comportements réalistes à un niveau individuel, ni de générer des activités sur des fenêtres temporelles au-delà de la journée, ni de reproduire de manière satisfaisante des activités collectives, ni de gérer des comportements réactifs ou adaptatifs.

2.2.1 Réalismes macroscopique et individuel

Regardons les résultats issus de la dernière

enquête emploi du temps en France[2]. La figure 1 présente le pourcentage d'individus des catégories « actifs », « étudiants » et « retraités » ayant déclaré être en train de dormir au cours de la journée.

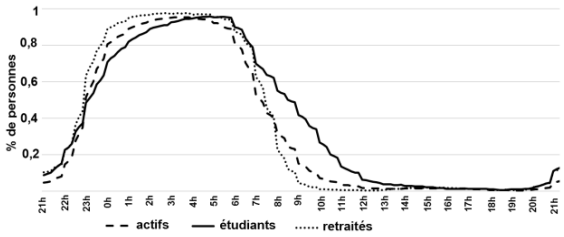


Figure 1: L'activité sommeil au cours de la journée

Ce type de données donne l'illusion d'une uniformité des comportements. Or l'étude des carnets individuels montre que dans 10% des journées, l'activité « sommeil » a été rapportée plusieurs fois (jusqu'à 7 fois). Il est impossible de retrouver cette variabilité à partir des courbes agrégées. Elles ne sont pas suffisantes pour reproduire des comportements individuellement réalistes (les interruptions et répétitions ne sont pas repérables).

2.2.2 Variabilité des activités au cours du temps

Comme le montrent [8], [17], l'activité humaine est caractérisée par des activités de routine qui sont régulièrement les mêmes d'un jour à l'autre, et par des variations autour de ces routines. La question qui se pose est donc : à partir des données des enquêtes emploi du temps, qui ne portent que sur des journées individuelles, comment construire un emploi du temps sur une période plus longue, par exemple sur une semaine ? Générer des copies d'une même journée ne permet pas d'obtenir une variabilité dans les comportements. D'un autre côté, construire un emploi du temps d'une semaine, constitué de journées générées sans lien les unes avec les autres ne permet pas d'obtenir de comportements de routine.

2.2.3 Comportements collaboratifs et adaptatifs

Aucune des méthodes statistiques existantes ne permet de simuler de manière satisfaisante les comportements collaboratifs ([14] ne prend en compte qu'un petit nombre d'activités). Or, l'organisation d'un ménage émerge de la collaboration de ses membres. Ne pas traiter cet aspect limite donc fortement le réalisme des activités simulées à l'échelle du ménage.

De plus, ces méthodes ne peuvent que reproduire les activités recensées. Elles ne sont pas capables d'expliquer les comportements produits, ni de simuler des réactions face à un événement, ou des modifications de l'environnement.

3 Notre modèle

Afin de dépasser ces quatre limites, nous proposons un SMA en deux parties : 1) une partie « comportements prescrits », calibrés à partir de données statistiques, qui assure la validité macroscopique des comportements simulés ; 2) un modèle d'agent autonome, centré sur un module de sélection de l'action qui permet de simuler des agents réactifs, adaptatifs, collaboratifs, et donc d'augmenter le réalisme des activités individuelles.

Le modèle d'agent utilisé est celui de la plate-forme de Simulation Multi-Agent de Comportement Humain SMACH [15].

3.1 Le modèle agent

SMACH est spécialisée sur les activités se déroulant à l'intérieur des logements. Sa capacité à reproduire des comportements individuels a déjà été validée, grâce à des simulations participatives [16]. Dans cette plate-forme, chaque individu est modélisé comme un agent avec des objectifs (activités à réaliser), des connaissances (sur les autres individus et sur l'environnement) et des préférences (en termes de confort, de comportement et d'emploi du temps). Les agents sont capables d'échanger des informations, de coordonner leurs activités, de planifier leurs journées et leurs semaines.

[2] https://www.insee.fr/fr/metadonnees/source/s1224

Chaque jour de simulation, les agents reçoivent une liste d'activités à effectuer : leur emploi du temps. A chaque pas de temps, ils sélectionnent leur activité courante dans cette liste, grâce à un mécanisme de sélection de l'action basé sur un niveau de priorité qui varie dynamiquement en fonction des connaissances de l'agent, de ses préférences, et de la situation.

3.2 Le modèle d'activité prescrites

Les activités du modèle *SMACH* sont dotées des paramètres suivants :

• **Durée**. Chaque activité est définie par une durée minimum et une durée maximum.

• **Rythme**. A chaque activité est attribué un nombre de répétitions par jour ou par semaine.

• **Période préférentielle (PP)**. A chaque activité est associée une période préférentielle (notée PP) indiquant les périodes de la journée (ou de la semaine) qui sont préférées pour la réalisation de l'activité.

• **Niveau de collectivité**. Ce niveau indique si l'activité est réalisée en solitaire ou en groupe. Exemple : « dîner » est une activité collective alors que « se laver les dents » ne l'est pas.

Ces paramètres vont être calibrés grâce aux données statistiques des enquêtes emploi du temps, à l'exception du paramètre de collectivité qui, dans la version actuelle, est initialisé manuellement par le modélisateur.

4 Utilisation des enquêtes emploi du temps

4.1 Calibration des paramètres d'activité

4.1.1 Formalisation des données statistiques

Les données statistiques d'une enquête emploi du temps sont composées d'un ensemble *C* de « carnets » *c* : $C = \{c_1, ..., c_n\}$, d'un ensemble *A* d'activités *a* : $A = \{a_1, ..., a_p\}$, et d'un ensemble *IND* d'individus *ind* (les personnes interrogées) : $IND = \{ind_1, ..., ind_k\}$. Les individus sont dotés d'un sexe, d'un âge, et d'un statut de travail (actif, retraité, chômeur, etc.). Chaque carnet *c* est lié à un individu *ind*, et à un jour *j*. Il contient un ensemble d'instances d'activités, telles que :
$c_{ind,j} = \{a_1, ..., a_m\}$.

Soient *i* le « type d'individu » (défini par les caractéristiques de sexe, âge et statut) et *j* le « type de jour » (typiquement jour de semaine/week-end). Ces catégories permettent de créer des groupes de carnets correspondant à des situations similaires, c'est-à-dire remplis par des individus d'un même type, pour un même type de journée. Grâce à ces groupes de carnets, nous pouvons établir des hypothèses statistiques sur la façon dont les individus d'un certain type emploient leur temps pendant des jours d'un certain type.

Soit *J* l'ensemble des types de jour *j* : $J = \{j_1, ..., j_q\}$ et *I* l'ensemble des types d'individus *i* : $I = \{i_1, ..., i_y\}$

4.1.2 Extraction des données

Soit $N(a, c)$ le nombre d'instances de l'activité *a*, dans le carnet *c*.
Soit $C_{ij} = \{c_1, \ldots, c_n\}$ l'ensemble des carnets *c* étant liés à un individu de type *i* et à un jour de type *j*.
Soit $P_N(a, i, j)$ la distribution de probabilité des valeurs de $N(a, c)$ pour tout $c \in C_{ij}$.
Soit $C_N(a, i, j) = \{c_1, ..., c_m\}$ l'ensemble des carnets *c* (liés à un individu de type *i* et à un jour de type *j*) dans lesquels il y a *N* instances de *a*.
Soit $D_N(a, i, j)$ la collection de tous les couples (heure de début, heure de fin), des instances de *a* dans tous les carnets *c* de $C_N(a, i, j)$.

Notre modèle nécessite l'extraction de tous les $P_N(a, i, j)$ et tous les $D_N(a, i, j)$ pour chaque activité *a*, pour chaque valeur possible de *N*, pour chaque type d'individu *i*, et pour chaque type de jour *j*.

Pour chaque $D_N(a,i,j)$, on calcule $moy(D_N(a,i,j))$ et $et(D_N(a,i,j))$, la durée moyenne et l'écart-type des activités de $D_N(a,i,j)$.

On construit ensuite $P_{ST}(D_N(a,i,j))$ et $P_{FT}(D_N(a,i,j))$, respectivement la distribution probabiliste des heures de début et des heures de fin des activités de $D_N(a,i,j)$.

4.1.3 Conclusion sur la calibration

Les paramètres de durées de nos activités sont calculés à partir de $D_N(a,i,j)$. Les périodes préférentielles (PP) sont déduites de $P_{ST}(D_N(a,i,j))$ et $P_{FT}(D_N(a,i,j))$. Le paramètre de rythme quant à lui, est déduit des $P_N(a,i,j)$.

Cependant, la question de la variabilité des comportements au cours du temps reste à traiter.

4.2 Les emplois du temps « de routine »

4.2.1 Création des données de variabilité

Les enquêtes emploi du temps ne documentent que des journées isolées, et jamais plus de deux pour un même individu. A partir de ces données il n'est pas possible de générer directement un emploi du temps sur une période temporelle plus longue qu'une journée. Pour dépasser cette limitation, nous allons poser une hypothèse (*H*), nous permettant de « créer » l'information manquante.

H : « les comportements humains les plus atypiques à un niveau macroscopique (les comportements les moins représentés dans l'enquête) doivent être des comportements inhabituels à un niveau individuel (c'est-à-dire qu'ils ne peuvent pas être des comportements de routine d'un individu). »

Par exemple, si dans une enquête donnée, très peu de carnets recensent une activité de sommeil d'une durée inférieure à 2 heures, alors, d'après *H*, nous considérerons qu'aucun individu n'a pour habitude de dormir moins de deux heures par nuit. Les individus simulés peuvent toujours dormir moins de 2 heures, mais uniquement de manière exceptionnelle.

Cette hypothèse est une simplification de la réalité, puisqu'il existe des individus qui dorment en moyenne 2 heures par nuit. Cependant, la marge d'erreur de cette hypothèse se limite à des cas particuliers rares. Nous acceptons notre incapacité à modéliser ces cas particuliers.

Grâce à *H*, il est possible de créer les informations liées à la variabilité des comportements qui manquent dans les données statistiques. Ainsi, pour chaque activité, on peut séparer les instances de cette activité qui sont « habituelles », des instances « inhabituelles ». Les instances habituelles sont les comportements « de routine » propres à l'activité en question, alors que les instances inhabituelles représentent les variations individuelles autour de ces routines.

Pour chaque individu, il devient possible de créer un emploi du temps « de routine », uniquement basé sur les instances de routine de chaque activité. En pratique, on considère qu'une instance est habituelle si sa durée est comprise dans l'intervalle :
$[moy(D_N(a, i, j))-et(D_N(a, i, j)); moy(D_N(a, i, j))+et(D_N(a, i, j))]$
c'est-à-dire si sa durée est située à moins d'un écart-type de la moyenne. D'après la définition de l'écart-type, 68,2% des instances de chaque activité sont dans cet intervalle, donc les emplois du temps de routine se basent sur les 68,2% des instances les plus « habituelles ».

4.2.2 Génération des emplois du temps de routine

Tableau 1: Rappel des notations

$N(a,c)$	Nombre d'instances de l'activité *a*, dans le carnet *c*
$P_N(a, i, j)$	Distribution de probabilité de $N(a,c)$ pour tous les carnets *c* liés à des individus de type *i* et des jours de type *j*
$D_N(a, i, j)$	Couples (heure de début, heure de fin), des instances de $P_N(a, i, j)$
$moy(D_N(a, i, j))$	Durée moyenne des couples de $D_N(a,i,j)$
$et(D_N(a, i, j))$	Ecart-type des couples de $D_N(a, i, j)$
$P_{ST}(D_N(a, i, j))$	Distribution probabiliste des heures de début de $D_N(a, i, j)$
$P_{FT}(D_N(a, i, j))$	Distribution probabiliste des heures de fin de $D_N(a, i, j)$

Notons $D_R(ind, j) = \{activité_1, ..., activité_m\}$ l'emploi du temps de routine de l'individu simulé *ind* (dont le type d'individu est *i*), pour le type de jour *j*.
Pour chaque activité *a* dans *A*, on calcule *N**, le nombre d'instances de l'activité *a* dans $D_R(ind,j)$, grâce à un tirage au sort dans $P_N(a,i,j)$.

Les caractéristiques de *a*, pour un individu de type i, et pour un jour de type *j* sont : $\mathbf{Durée_{min}(a)} = moy(D_{N*}(a,i,j)) - et(D_{N*}(a,i,j))$

$Durée_{max}$(a) = moy(D_{N*}(a,i,j)) + et(D_{N*}(a,i,j))
PP(a) : les heures de début et de fin de PP(a) sont tirées aléatoirement respectivement dans P_{ST}(D_{N*}(a, i, j)) et P_{FT}(D_{N*}(a, i, j))
Rythme(a) : *N** répétitions.

Note : chaque individu aura un emploi du temps de routine unique, en raison des tirages aléatoires pour déterminer le N* de chaque activité, pour chaque type de jour.

4.2.3 Réintroduction de la variabilité

L'emploi du temps de routine d'un individu peut être copié sur plusieurs jours : il correspond à sa routine. Cependant, le comportement de l'individu doit varier autour de cette routine. Pour représenter cela, chaque jour, chaque instance *a* de l'emploi du temps de routine a une probabilité de 31,8% de devenir une instance inhabituelle (l'emploi du temps de routine ne représente que 68,2% des instances). Cela se traduit par une modification des durées de l'instance *a* :
$Durée_{min}$(a)=moy(D_{N*}(a,i,j)) - 2×et(D_{N*}(a,i,j))
$Durée_{max}$(a) = moy(D_{N*}(a,i,j)) - et(D_{N*}(a,i,j))
OU :
$Durée_{min}$(a) = moy (D_{N*}(a,i,j)) + et(D_{N*}(a,i,j))
$Durée_{max}$(a)= moy(D_{N*}(a,i,j))+2*et(D_{N*}(a,i,j))

Remarque : les durées de 4,5% des instances sont situées à plus de 2 écarts-type de la moyenne. Nous préférons ne pas les modéliser, en raison du manque de représentativité des données statistiques associées.

4.2.4 Conclusion sur l'algorithme

Cet algorithme permet de respecter la répartition statistique des durées, écart-type, et nombre de répétitions des activités simulées. De plus, sur des plages temporelles longues, les individus simulés présentent des activités de routine et des variations autour de celles-ci.

Les emplois du temps ainsi construits sont envoyés au processus de sélection de l'action de SMACH, qui décidera du comportement effectif des agents en prenant en compte l'environnement. C'est ce mécanisme qui permet de donner de la réactivité aux agents.

5 Expérimentations

Afin de valider le modèle, nous l'avons implémenté et testé avec les données d'une enquête emploi du temps : la dernière enquête emploi du temps de l'INSEE. Le but de cette expérimentation est de montrer que les activités simulées par notre modèle sont à la fois réalistes à un niveau macroscopique et à un niveau microscopique.

5.1 Les activités

Dans cette enquête, 140 activités sont recensées (« lire un journal », « lire un livre », « regarder un film », « regarder la TV », etc.). Nous les rassemblons en 30 activités plus cohérentes par rapport à nos objectifs de simulation (« lire », « regarder la TV », « dîner », « travailler », etc.).

5.2 Validation macroscopique

Grâce au mécanisme de sélection de l'action, les agents ont un certain degré d'autonomie. Cette particularité leur permet de prendre du recul par rapport à leurs emplois du temps ayant été générés à partir des données statistiques. Il est nécessaire de vérifier que les comportements effectivement simulés correspondent toujours aux statistiques.

Nous allons vérifier cette hypothèse pour le type d'individu i_1 : « homme actif entre quarante et cinquante ans », et pour le type de jour j_1 : « jour ouvrable ».

On génère 100 emplois du temps quotidiens correspondant à un individu de type i_1 et un jour de type j_1. Ces emplois du temps sont ensuite simulés dans la plate-forme.

5.2.1 Résultats

La figure 2 illustre la répartition de l'activité « sommeil » pendant les 100 jours simulés.

La seule différence notable est un écart vertical le soir et le matin. Il est principalement causé par les 4,5% d'activités les plus rares, non simulées par notre modèle.

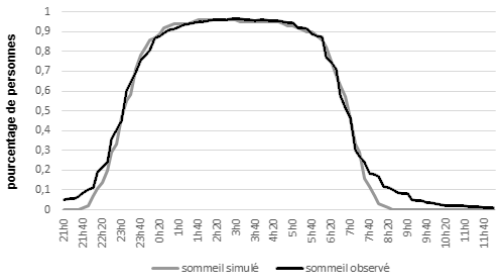


Figure 2: Comparaison des activités de sommeil simulées et observées

La figure 3 donne un autre exemple d'activité : « préparer le dîner ».

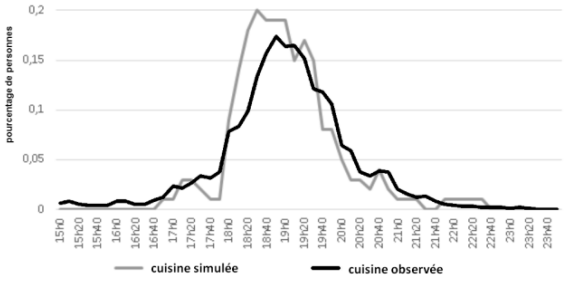


Figure 3: Comparaison des activités de cuisine simulées et observées

Les écarts sont faibles : jamais plus de 3 points (écart vertical) et 20 minutes (écart horizontal) de différence.

En ce qui concerne le nombre de répétitions des activités au cours de la journée, regardons les figures 4 et 5, qui indiquent le nombre de répétitions des activités « préparer le dîner » et « faire le ménage » au cours d'une journée.

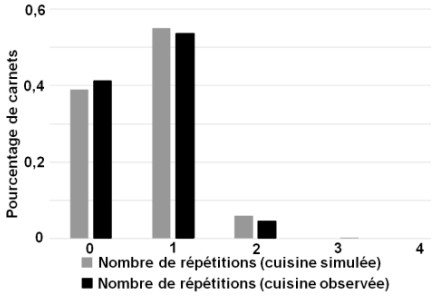


Figure 4: Nombre de répétitions des activités de cuisine

Les écarts sont très limités pour l'activité de cuisine (<2 points) et un peu plus importants (<4 points) pour le ménage. En ce qui concerne le ménage, la raison principale est la difficulté pour nos agents simulés de trouver le temps durant leur journée de faire plus de 2 fois le ménage.

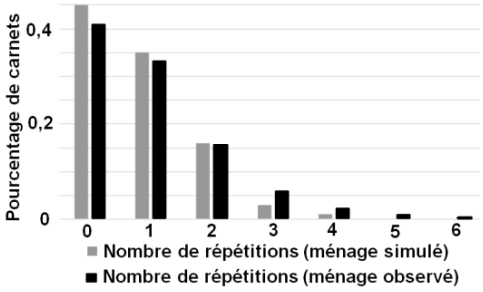


Figure 5: Comparaison du nombre de répétitions des activités "ménages" simulées et observées

5.2.2 Conclusions sur la validation macroscopique

Malgré les capacités d'autonomie de nos agents, les comportements simulés restent très proches des comportements observés.

5.3 Validation microscopique

L'objectif est de mesurer la qualité de l'adaptation de nos agents à des modifications environnementales. En raison de la difficulté à mesurer objectivement la qualité d'un comportement, nous préférons présenter des exemples de comportements d'adaptation.

Le scénario d'activité considéré est celui d'un agent de type i_2 : « femme active dans la trentaine », pour un jour de type j_2 : « jour ouvrable ». Nous allons générer 3 emplois du temps correspondant à cette situation, et les simuler. Chaque emploi du temps correspond à un déroulement possible d'une journée de semaine pour une femme active. Ils auront donc des similarités (travail, sommeil), mais également des différences (ménage, sorties, etc.). Chaque emploi du temps sera ensuite simulé une deuxième fois, mais avec l'activation d'un événement « panne de réveil », qui a pour conséquence de laisser dormir l'agent une heure de plus que d'habitude.

5.3.1 Exemples de scénarios

Ci-dessous, les 3 couples de simulations : journée « habituelle » sans évènement, et

journée « inhabituelle » avec évènement :

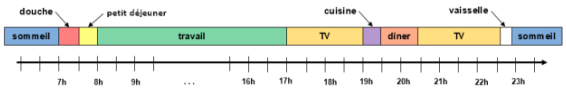


Figure 6 : diagramme d'activité (journée habituelle 1)

1)a) Journée habituelle (figure 6) : l'individu se réveille à 7h, prend une douche, prend son petit déjeuner et se rend au travail à 8h. Il revient du travail à 17h et passe la soirée à la maison (télévision, cuisine, dîner, à nouveau télévision, puis vaisselle et sommeil à 23h).

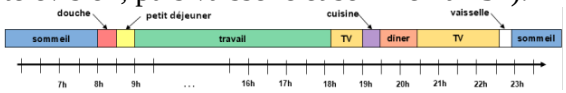


Figure 7 : diagramme d'activité (journée inhabituelle 1)

b) Journée « inhabituelle » (figure 7) : l'individu se réveille à 8h à cause de la panne de réveil. Il **décale son emploi du temps du matin** et part une heure en retard au travail. Il rentre une heure plus tard (18h10). Une fois chez lui, il réduit le temps passé devant la télévision d'une heure environ (**réduction du temps d'une activité**).

2)a) Le deuxième individu a une journée « habituelle » similaire à la journée 1)a), si ce n'est qu'il se lève plus tôt, et qu'il occupe différemment sa soirée.
b) Sa journée « inhabituelle » se traduit par un **déplacement** de la douche dans la soirée (au lieu du matin), et une **réduction** du temps de travail.

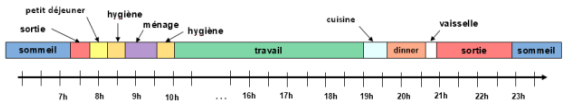


Figure 8: diagramme d'activité (journée habituelle 3)

3)a) Le troisième individu a une journée « habituelle » assez différente (voir figure 8).
b) Lors de la panne de réveil, il **réduit** le temps passé à faire du ménage et **supprime** un passage dans la salle de bain (voir figure 9).

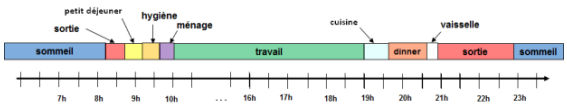


Figure 9: diagramme d'activité (journée inhabituelle 3)

5.3.2 Conclusion sur la validation individuelle

Ces exemples montrent que les agents sont capables de réagir à un événement et de s'adapter à un changement de situation en modulant leur emploi du temps. Ils peuvent décaler, supprimer, réduire, ou déplacer des activités.

5.3.3 Conclusion sur cette étude empirique

Cette expérimentation n'est pas suffisante pour valider le réalisme microscopique de nos activités simulées. Notre proposition (non implémentée à l'heure actuelle) pour faire cela est de générer des carnets d'emploi du temps à partir de simulations, puis d'effectuer un processus de classification englobant à la fois des carnets réellement observés et des carnets simulés. En se basant sur des travaux précédents [18], nous pensons que si l'on obtient des clusters mixtes (c'est-à-dire comportant des carnets réellement observés et des carnets simulés), alors les activités simulées ne peuvent pas être distinguées des réelles, par rapport aux variables considérées.

6 Conclusion et perspectives

Dans cet article nous avons montré comment calibrer des activités humaines dans un SMA à l'aide de données statistiques (enquêtes emploi du temps). Ces activités prescrites sont ensuite couplées avec un module de sélection de l'action autonome pour former un modèle multi-agent hybride, qui permet la modélisation d'agents réactifs, adaptatifs, et collaboratifs. Par ailleurs, nous avons présenté une méthode de génération d'emploi du temps permettant la simulation d'activités réalistes sur de longues plages temporelles, grâce à la modélisation de la variabilité des activités autour des comportements de routine.

Grâce à des expérimentations macroscopiques et microscopiques, nous avons montré que ce modèle permet de coupler les réalismes des 2 niveaux. Nous modélisons des individus autonomes, capables d'adapter leur comportement à l'environnement, tout en respectant un cadre comportemental calculé à

partir de données statistiques.

Ce modèle est utilisé au sein de la plate-forme de simulation SMACH, couplé à un générateur de population permettant de générer une sous-population représentative d'une population cible. Cette sous-population est ensuite simulée et ses interactions avec les divers équipements électriques produit une courbe de charge de la consommation électrique réaliste au niveau individuel, et au niveau agrégé.

La prochaine étape de notre travail est d'étudier le réalisme individuel des activités simulées, en les comparant de manière systématique avec des comportements réels.

Références

[1] S. Sharma and S. Otunba, "Collaborative virtual environment to study aircraft evacuation for training and education," *Collab. Technol. Syst.*, pp. 569–574, 2012.

[2] B. Ulicny and D. Thalmann, "Crowd simulation for interactive virtual environments and VR training systems," in *Computer Animation and Simulation*, Springer Vienna, 2001, pp. 163–170.

[3] D. Traum, J. Rickel, J. Gratch, and S. Marsella, "Negotiation over tasks in hybrid human-agent teams for simulation-based training," *Proc. Second Int. Jt. Conf. Auton. agents multiagent Syst.*, pp. 441–448, 2003.

[4] A. Rao and M. Georgeff, "Modeling rational agents within a BDI-architecture" in *Proc. of the 2nd International Conference on Principles of Knowledge Representation and Reasoning*, 1991, pp. 473--484.

[5] V. Lanquepin, K. Carpentier, and D. Lourdeaux, "HUMANS: a HUman models based artificial eNvironments software platform" *Proc. Virtual Real. Int. Conf. Laval Virtual*, p. 9., 2013.

[6] J. Hubner and J. Sichman, "Developing organised multiagent systems using the MOISE+ model: programming issues at the system and agent levels," *Int. J. Agent-Oriented Softw. Eng.*, vol. 1, pp. 370–395, 2007.

[7] A. Drogoul, D. Vanbergue, and T. Meurisse, "Multi-agent based simulation: Where are the agents?," *Int. Work. Multi-Agent Syst. Agent-Based Simul.*, pp. 1–15, 2002.

[8] Y. Haradji, G. Poizat, and F. Sempé, *Human activity and social simulation*. Boca Raton, FL : CRC Press, 2012.

[9] L. Stinson, "Measuring how people spend their time: a time-use survey design," *Mon. Lab. Rev.*, vol. 122, p. 12, 1999.

[10] Y. Chiou, "Deriving US household energy consumption profiles from american time use survey data a bootstrap approach," *11th Int. Build. Perform. Simul.*, 2009.

[11] I. Richardson, M. Thomson, D. Infield, and C. Clifford, "Domestic electricity use: A high-resolution energy demand model," *Energy Build.*, vol. 42, no. 10, pp. 1878–1887, 2010.

[12] J. Widén, A. Molin, and K. Ellegård, "Models of domestic occupancy, activities and energy use based on time-use data: deterministic and stochastic approaches with application to various building-related," *J. Build. Perform. Simul.*, vol. 5, no. 1, pp. 27–44, 2012.

[13] J. Tanimoto, A. Hagishima, and H. Sagara, "A methodology for peak energy requirement considering actual variation of occupants' behavior schedules," *Build. Environ.*, vol. 43, no. 4, pp. 610–619, 2008.

[14] Y. Yamaguchi and Y. Shimoda, "Evaluation of a behavior model of occupants in home based on Japanese national time use survey," *Proc. BS2015*, 2015.

[15] E. Amouroux, T. Huraux, F. Sempé, and N. Sabouret, "SMACH: Simuler l'activité humaine pour limiter les pics de consommation électrique.," in *JFSMA*, 2013.

[16] É. Amouroux, T. Huraux, F. Sempé, N. Sabouret, and Y. Haradji, "SMACH: Agent-Based Simulation Investigation on Human Activities and Household Electrical Consumption," *Commun. Comput. Inf. Sci.*, vol. 449, pp. 194–210, 2014.

[17] M. Feldman and B. Pentland, "Reconceptualizing organizational routines as a source of flexibility and change," *Adm. Sci. Q.*, 2003.

[18] K. Darty, J. Saunier, and N. Sabouret, "Behavior Clustering and Explicitation for the Study of Agents' Credibility: Application to a Virtual Driver Simulation," *Int. Conf. Agents Artif. Intell.*, pp. 82–99, 2014.

Deploiement de graphes de facteurs pour l'exécution d'algorithmes DCOP sur des infrastructures ouvertes

P. Rust[a,b] pierre.rust@orange.com
G. Picard[b] gauthier.picard@emse.fr
F. Ramparany[a] fano.ramparany@orange.com

[a]Orange Labs, OLPS/SOFT/NEC, France

[b]Univ Lyon, MINES Saint-Étienne, CNRS, Laboratoire Hubert Curien UMR 5516, F-42023 Saint-Étienne, France

Abstract

Dans le cadre des problèmes d'optimisation distribuée sous contraintes, utiliser des algorithmes par propagation de croyances nécessite de déployer les éléments du graphe de facteurs sur lequel le processus de résolution opère. Ici, nous nous intéressons au cas particulier de la configuration d'environnements intelligents et ouverts, dans lesquels plusieurs équipements connectés doivent se coordonner afin de trouver une configuration optimale, sous certaines contraintes partagées (e.g. modèles physiques et règles utilisateur). Le problème du déploiement du graphe de facteurs sous-jacent peut être vu comme un problème d'optimisation, solvable de manière centralisée. Mais, en la présence de dynamiques environnementales, on ne peut se permettre un redémarrage et une résolution centralisée. Ainsi, le système doit effectuer des adaptations locales et en cours de fonctionnement du déploiement. Nous proposons ici des solutions et les évaluons par simulation.

Keywords: *intelligence ambiante, optimisation, graphe de facteurs, DCOP*

Abstract

Using belief-propagation based algorithms to solve distributed constraint optimization problems (DCOPs) requires deploying the factor graph elements on which the distributed solution operates. In some utility-based multi-agent settings, this deployment is straightforward. However, when the problem gains in complexity by adding other interaction constraints (like shared costs or dependencies), the question of deploying these shared factors arises. Here, we address this problem in the particular case of dynamic and open environments, where several incoming and outcoming devices have to coordinate as to reach an optimal configuration, under some shared constraints. The factor graph deployment problem (FGDP) over a set of constrained devices can be mapped to an optimization problem, then solvable in a centralized manner. But, when dealing with the openness of the environment (e.g. new devices) restarting the system or relying on a centralized solver is not affordable. Thus, the system has to perform on-line and local deployment adaptations from prior state. In this paper, we present some solutions and experiment them on a simulated smart home environment.

Keywords: *ambient intelligence, optimization, factor graph, DCOP*

1 Introduction

Un problème classique lors de l'usage de techniques par propagation de croyances, comme Max-Sum [6], est de décider où héberger les calculs relatifs aux évaluations des variables et des facteurs. En effet, Max-Sum opère sur un graphe de facteurs [1] représentant le problème à résoudre, en échangeant des messages entre variables et facteurs connectés. Évaluer les messages à envoyer nécessite des calculs devant être hébergés par des agents. Dans certains contextes, cette association est directe. C'est le cas dans les problèmes purement utilitaires, où chaque agent possède une seule variable et un facteur d'utilité connecté à d'autres variables appartenant à d'autres agents [5]. Cependant, dans des contextes plus complexes où certains facteurs ou variables sont partagés par plusieurs agents, la question du déploiement du graphe de facteurs se pose.

Un exemple de tel problème est le problème de configuration d'environnements intelligents (SECP) [11]. Ici, plusieurs objets connectés doivent s'auto-configurer afin de répondre aux objectifs de l'utilisateur et minimiser la consommation électrique. SECP est modélisé comme un problème d'optimisation sous contraintes distribué (DCOP), représenté sous la forme d'un

1. ou *factor graph*, graphe bipartite composé des variables de décision et des contraintes, ou facteurs

graphe de facteurs. A cause de la dynamique de cet environnement (e.g. apparition/disparition d'objets, ajout/suppression/modification d'objectifs, mise à jour de propriétés captées) et des capacités limitées de communication et calcul de ces objets, déployer le graphe de facteurs de manière efficace est une problématique clé qui ne peut seulement reposer sur une solution hors-ligne et centralisée.

L'article est structuré comme suit. La section 2 rappelle brièvement le cadre des DCOP et le modèle graphique sous-jacent, le graphe de facteurs. La section 3 identifie le problème de déploiement de graphe de facteurs (FGDP) et présente un programme linéaire pour le résoudre de manière optimale et centralisée. Les sections suivantes discutent des différents cas d'ouverture pouvant impacter le déploiement : ajout (section 4), ou retrait d'équipement (section 5). Des résultats d'expérimentations par simulation et leur analyse sont fournis en section 6. Enfin, la section 7 conclut ce papier.

2 Contexte et notions importantes

Cette section présente le contexte applicatif, le cadre des DCOP et le modèle de graphe de facteurs sur lequel les méthodes d'optimisation par propagation de croyances opèrent.

2.1 Problème de configuration d'environnements intelligents

Dans de précédents travaux [11], nous avons abordé le problème de configuration spontanée de scènes dans des environnements intelligents, via le paradigme multi-agents – plus particulièrement le cadre de l'optimisation distribuée (DCOP), où des algorithmes par envoi de messages mettent en œuvre un protocole de configuration.

Ici, les objets connectés font partie d'un SMA dont la tâche consiste à trouver une configuration optimale, sans supervision. Dans ce modèle, chaque effecteur est représenté par une variable de décision et doté d'une fonction de coût représentant sa configuration énergétique. L'influence des effecteurs sur l'environnement est modélisée par des modèles de dépendance physique, composés d'une contrainte et d'une variable. Les souhaits de l'utilisateur sont exprimés sous forme de règles, modélisées par une fonction d'utilité dépendant de la distance entre les objectifs de la règle et l'état de l'environnement. Ce modèle permet d'éviter à l'utilisateur de spécifier explicitement le rôle de chaque objet, facilitant la définition de règles et l'introduction de nouveaux objets en cours de fonctionnement. Nous proposions d'utiliser des solveurs de DCOP par inférence, comme DPOP et Max-Sum, pour mettre en œuvre le protocole de coordination. En nous basant sur nos expérimentations sur un scénario réaliste simulé, Max-Sum est le plus approprié pour les objets aux capacités limitées que nous rencontrons dans ce contexte. De plus, notre modèle SECP est une approche viable pour la coordination autonome entre de tels objets. Pour une présentation plus détaillée de ce modèle, et des résultats obtenus, nous renvoyons le lecteur vers [11]. Cette étude, cependant, n'abordait que brièvement la problématique du déploiement des éléments du graphe sur les équipements.

Problème 1. *Étant donné un ensemble d'effecteurs (et leurs coûts respectifs), un ensemble de capteurs, un ensemble de règles de scènes (et leurs utilités respectives), et un ensemble de modèles de dépendance physique,* **le problème de configuration d'environnements intelligents** *(ou* **SECP***, pour* **Smart Environment Configuration Problem***) consiste à trouver la configuration des effecteurs qui maximise l'utilité des règles définies par l'utilisateur, tout en minimisant la consommation énergétique globale et en respectant les dépendances physiques.*

2.2 Le cadre des DCOP

Une façon de modéliser un problème distribué de coordination est de le formaliser comme un problème d'optimisation sous contraintes distribué (DCOP) [10].

Définition 1 (DCOP)**.** *Un problème d'optimisation sous contraintes distribué (ou* **DCOP** *pour* **Distributed Constraint Optimization Problem***) est un tuple* $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C}, \mu \rangle$*, où :*

- $\mathcal{A} = \{a_1, \ldots, a_{|A|}\}$ *est un ensemble d'agents ;*
- $\mathcal{X} = \{x_1, \ldots, x_N\}$ *sont les variables appartenant aux agents ;*
- $\mathcal{D} = \{\mathcal{D}_{x_1}, \ldots, \mathcal{D}_{x_N}\}$ *est un ensemble de domaines finis tels que la variable* x_i *prend ses valeurs dans* $\mathcal{D}_{x_i} = \{v_1, \ldots, v_k\}$*;*
- $\mathcal{C} = \{c_1, \ldots, c_M\}$ *est un ensemble de contraintes souples, où chaque* c_i *définit un coût* $\in \mathbb{R} \cup \{\infty\}$ *pour chaque combinaison d'affectation de valeurs au sous-ensemble de variables impliquées dans cette contrainte* $\mathcal{X}_i \subseteq \mathcal{X}$*.*

Une solution *à un DCOP est une affectation de valeurs à toutes les variables qui minimise la somme totale de coûts* $\sum_{m=1}^{M} c_m(\mathcal{X}_m)$.

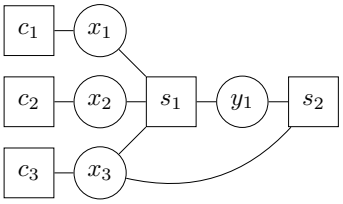
Comme souligné par [3], les DCOP ont été largement étudiés et appliqués à de nombreux domaines de référence, et présentent de nombreuses propriétés intéressantes : (i) focus sur des approches de résolution décentralisées où les agents négocient une solution jointe au travers d'échanges locaux de message ; (ii) techniques de résolution exploitant la structure du domaine (en l'encodant dans des contraintes) pour s'attaquer à des problèmes computationnels difficiles ; (iii) grande variété de méthodes de résolution allant de méthodes exactes à des techniques heuristiques et approchées. Dans cet article, nous nous intéressons plus particulièrement aux algorithmes basés sur la propagation de croyances, comme Max-Sum [6], où la notion de valeurs marginales décrit la dépendance de la fonction de coût globale vis-à-vis de chaque variables. De tels algorithmes opèrent sur un modèle graphique en échangeant les valeurs marginales via des messages entre variables et contraintes (appelées *facteurs*), et *vice versa*. Dans ce contexte, la charge de calculs résulte de l'évaluation des messages (principalement des facteurs), et la charge de communication résulte des messages envoyés (dont la taille est proportionnelle à la variable destinataire ou émettrice).

2.3 Graphe de facteurs et déploiement

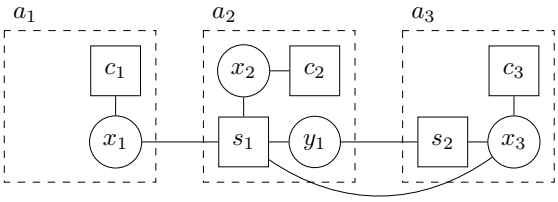
Dans le cas général avec facteurs n-aires, un DCOP peut être représenté par un graphe de facteurs : un graphe bipartite non dirigé, dans lequel les nœuds représentent des variables et les contraintes (ou facteurs) et les arcs représentent les liens entre ces variables et facteurs, comme illustré dans la figure 1a .

Définition 2. *Le* graphe de facteurs *(ou* **factor graph, FG***) du DCOP* $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ *est le graphe bipartite* $FG = \langle V_x, V_f, E \rangle$*, où* $V_x = \mathcal{X}$ *est l'ensemble de nœuds variables ;* $V_f = \mathcal{C}$ *est l'ensemble des nœuds facteurs ;* $E = \{(x_i, f_j) \mid x_i \in \mathcal{X}_j\}$ *est l'ensemble des arcs.*

Le graphe de facteurs est indépendant de l'ensemble des agents $\mathcal{A}$. Cependant, en pratique, les variables et les facteurs doivent être déployés sur des agents, ce qui impacte les charges de calcul et de communication. Notons que dans la suite de cet article nous utiliserons les termes



(a) Graphe de facteurs



(b) Déploiement

FIGURE 1 – Un graphe de facteurs (1a) et un déploiement possible (1b) sur 3 agents (a_1, a_2 et a_3)

"agents" et "nœuds de calcul" interchangeablement.

Exemple. *Soient les agents* a_1 *à* a_3*. La figure 1a présente un graphe de facteurs avec des coûts privés unaires (*c_i*'s) attachés à des variables privées* x_i *(chacune appartenant à un agent* a_i*), un facteur partagé binaire (*s_2*) liant la variable partagée* y_1 *et la variable privée* x_3*, et un facteur partagé n-aire (*s_1*) liant les* x_i *et* y_1*. Un déploiement possible est illustré dans la figure 1b.*

Dans le graphe de facteurs, certains éléments sont *possédés* par des agents (e.g. a_1 possède x_1). Notons $\phi(e) \in \mathcal{A} \cup \{\emptyset\}$ le propriétaire de l'élément e, avec $\phi(e) = \emptyset$ si est seulement si e n'appartient à aucun agent dans $\mathcal{A}$. Notons $\phi_x^{-1}(a_k)$ (resp. $\phi_f^{-1}(a_k)$) l'ensemble des variables (resp. facteurs) appartenant à l'agent a_k. Qu'ils aient ou non un propriétaire, tous les facteurs et variables doivent être *hébergés* par un agent (e.g. pour des raisons de calcul, e.g. a_2 héberge s_1). Notons $\mu(e) \in \mathcal{A}$ l'hôte de l'élément e, et $\mu_x^{-1}(a_k)$ (resp. $\mu_f^{-1}(a_k)$) l'ensemble de variables (resp. facteurs) hébergées par l'agent a_k. Les éléments possédés sont toujours hébergés par leur propriétaire, i.e. si $\phi(e) = a_k$ alors $\mu(e) = a_k$.

Définition 3. *L'application* $\mu : V_x \cup V_f \to \mathcal{A}$ *qui associe chaque élément du FG à un agent est dénommée* **déploiement**.

La décision de choisir quel agent héberge quelle

variable ou facteur partagé revient à construire cette application de déploiement. Dans les sections suivantes, nous formalisons la définition de cette décision lorsque les agents ont des capacités limitées, et proposons des techniques pour prendre la décision hors-ligne (déploiement initial) ou la réparer en-ligne (en cours de fonctionnement).

3 Déploiement optimal de FG

Le problème de décision du déploiement des éléments d'un graphe de facteurs à un ensemble d'agents est équivalent à un problème de partitionnement de graphe, qui est NP-difficile. Typiquement, de tels problèmes peuvent être modélisés comme des problèmes d'optimisation.

3.1 Définition du problème

Nous proposons ici un programme linéaire en nombres entiers (ILP), inspiré par les techniques de partitionnement de graphes de [4], et nous introduisons des contraintes spécifiques à nos agents aux capacités limitées.

Problème 2 (FGDP)**.** *Étant donné un graphe de facteurs $FG = \langle V_x, V_f, E\rangle$ et un ensemble d'agents $\mathcal{A} = \{a_1, \ldots, a_{|\mathcal{A}|}\}$ avec des capacités mémoire limitées, chacun possédant certains éléments de FG,* **le problème de déploiement de graphe de facteurs (FGDP)** *revient à affecter chaque élément partagé du FG à un agent, sans dépasser ses capacités mémoire, tout en minimisant la charge de communication globale.*

Tout d'abord, introduisons quelques notations. Notons $\mathbf{com}(x_i, f_j)$ la charge de communication induite par l'interaction entre x_i et f_j. Par exemple, $\mathbf{com}(x_i, f_j)$ est la taille des messages échangés entre la variable et le facteur :

$$\mathbf{com}(x_i, f_j) = \begin{cases} a.|\mathcal{D}_{x_i}| + b, & \text{si } (x_i, f_j) \in E \\ 0, & \text{sinon} \end{cases} \tag{1}$$

où a est le nombre d'octets pour représenter une valeur du domaine de la variable x_i et b est la taille de l'entête des messages. Soit $\mathbf{mem}(e), e \in V_x \cup V_y$ l'empreinte mémoire pour le calcul de l'élément e. C'est par exemple la taille en octets de la matrice de coûts d'une contrainte. Nous notons également $\mathbf{cap}(a_k)$ la capacité mémoire en octets de l'agent $a_k \in \mathcal{A}$.

Introduisons maintenant les variables qui associent les éléments du graphe de facteurs à des agents : x_i^k (resp. f_j^k) spécifie si la variable x_i (resp. le facteur f_j) est déployée sur l'agent a_k.

$$\forall x_i \in V_x, \quad x_i^k = \begin{cases} 1, & \text{si } \mu(x_i) = a_k \\ 0, & \text{sinon} \end{cases} \tag{2}$$

$$\forall f_j \in V_f, \quad f_j^k = \begin{cases} 1, & \text{si } \mu(f_j) = a_k \\ 0, & \text{sinon} \end{cases} \tag{3}$$

Afin de linéariser le problème, nous introduisons un autre ensemble de variables (les α_{ijk}) qui relient les variables du FG aux facteurs :

$$\forall x_i \in V_x, f_j \in V_f, a_k \in \mathcal{A}, \alpha_{ijk} = x_i^k.f_j^k \tag{4}$$

Nous pouvons maintenant modéliser le problème de déploiement d'un graphe de facteurs comme un programme linéaire en $0/1$:

$$\min_{x_i^k, f_j^k} \sum_{\substack{(x_i,f_j)\in E \\ a_k \in \mathcal{A}}} \mathbf{com}(x_i, f_j).(1 - \alpha_{ijk}) \tag{5}$$

avec

$$\forall x_i \in V_x, \quad \sum_{a_k \in \mathcal{A}} x_i^k = 1 \tag{6}$$

$$\forall f_j \in V_f, \quad \sum_{a_k \in \mathcal{A}} f_j^k = 1 \tag{7}$$

$$\forall a_k \in \mathcal{A}, \quad \sum_{x_i \in V_x} x_i^k + \sum_{f_j \in V_f} f_j^k \geq 1 \tag{8}$$

$$\forall (x_i, f_j) \in E, \quad \alpha_{ijk} \leq x_i^k \tag{9}$$

$$\forall (x_i, f_j) \in E, \quad \alpha_{ijk} \leq f_j^k \tag{10}$$

$$\forall (x_i, f_j) \in E, \quad \alpha_{ijk} \geq x_i^k + f_j^k - 1 \tag{11}$$

$$\forall a_k \in \mathcal{A}, x_i \in \phi_x^{-1}(a_k), \quad x_i^k = 1 \tag{12}$$

$$\forall a_k \in \mathcal{A}, f_j \in \phi_f^{-1}(a_k), \quad f_j^k = 1 \tag{13}$$

$$\forall a_k \in \mathcal{A}, \quad \sum_{x_i \in V_x} \mathbf{mem}(x_i).x_i^k + \sum_{f_j \in V_f} \mathbf{mem}(f_j).f_j^k \leq \mathbf{cap}(a_k) \tag{14}$$

L'objectif (5) minimise les communications entre éléments du graphe de facteurs qui ne sont pas déployés sur le même agent. Les contraintes (6) et (7) forcent chaque élément du graphe de facteurs à être déployé sur exactement un

agent. La contrainte (8) force l'utilisation de tous les agents disponibles. Enfin, inspirées des techniques de linéarisation proposées par [1], les contraintes (9) à (11) relient les x_i^k et f_j^k aux α_{ijk} de manière linéaire. Nous ajoutons les contraintes d'appartenance (12) et (13) qui réduisent l'espace de recherche. Nous forçons les variables (x_i) et facteurs (f_j's) à être hébergés par leur propriétaire, le cas échéant. Enfin, comme les agents ont une capacité mémoire limitée, la contrainte (14) restreint l'utilisation mémoire sur les agents.

Problème 3 (ILP-FGDP)**.** *Nous appelons* **ILP-FGDP** *le programme linéaire en* $0/1$*, constitué de l'objectif (5) et des contraintes (6) à (14), qui encode FGDP.*

Bien que ILP-FGDP soit NP-difficile, il peut être résolu en temps raisonnable de manière centralisée avec un algorithme de type branch-and-cut [9], surtout lorsque la matrice de coefficients est creuse (ce qui est notre cas ici). Comme décrit dans [11], à chaque fois qu'un utilisateur modifie hors-ligne le graphe de facteurs en ajoutant/supprimant/modifiant des facteurs, le déploiement optimal peut être déterminé en résolvant ILP-FGDP, et ensuite poussé sur les agents.

3.2 Le FG en action

Une fois le graphe de facteurs déployé, les agents mettent en œuvre l'algorithme Max-Sum pour trouver la configuration optimale en fonction de l'état de l'environnement capté [5, 11]. Ceci signifie que chaque fois que l'état de l'environnement change, les agents doivent continuer à propager des messages. En effet, les changements environnementaux peuvent activer de nouveaux facteurs (e.g. une règle utilisateur est déclenchée quand une personne est détectée dans une pièce), et la nouvelle configuration optimale doit être trouvée. Bien que Max-Sum ne soit pas strictement *anytime* il s'avère particulièrement efficace dans un contexte dynamique en maintenant à jour en continu une estimation des coûts et de la meilleur affectation. A chaque fois qu'un changement survient, les variables et facteurs impliqués envoient de nouveaux messages et les tables de coûts correspondantes sont mises à jour sans repartir de zéro. Cependant, dans des environnements dynamiques et ouverts comme ceux que nous considérons dans cet article, certains évènements peuvent directement impacter le déploiement du graphe de facteurs sur lequel Max-Sum opère, en particulier lorsque des agents apparaissent et disparaissent.

Calculer une solution à ILP-FGDP sur un équipement contraint n'est pas réaliste, par conséquent nous discutons dans les sections suivantes de techniques pour réparer en-ligne des déploiements à la suite de changement dans l'infrastructure. Nous souhaitons prendre en charge les changements dans des infrastructures ouvertes –i.e. ajout/retrait d'agents : (i) comment redéployer des facteurs et des variables lorsqu'un nouvel équipement/agent apparaît dans le système ? (ii) comment gérer les facteurs et variables hébergés par un équipement qui disparaît ? Une contrainte majeure est que lorsqu'une telle apparition ou disparition survient, les agents doivent s'auto-adapter sans l'aide d'une entité centrale.

4 Adaptation à l'arrivée d'agents

Ici, le déploiement actuel peut être révisé afin de bénéficier des nouvelles capacités de calcul et de mémoire du nouvel arrivant. Cependant, les équipements pouvant avoir des capacités assez limitées, ILP-FGDP (qui porte sur l'ensemble du graphe) ne peut être résolu sur un seul agent. Ainsi, nous proposons de restreindre la révision du déploiement à une sous-partie du graphe de facteurs et un sous-ensemble d'agents.

4.1 Notion de voisinage

Une fois qu'un équipement est ajouté, au lieu de résoudre le problème ILP-FGDP, nous pouvons envisager de le résoudre en ne considérant qu'un ensemble réduit d'agents (le nouvel arrivant et ses voisins), et une portion du graphe de facteurs (l'ensemble des éléments hébergés par le voisinage). Résoudre ce problème peut ne pas conduire à un optimum global du point de vue ILP-FGDP, mais nécessite beaucoup moins de calculs qu'une résolution sur l'intégralité du graphe de facteurs. Nous considérerons deux situations notables :

(A) soit le nouvel arrivant possède déjà des variables ou des facteurs, et par conséquent son voisinage est l'ensemble des agents hébergeant des facteurs ou des variables connectés à ses éléments ;

(B) soit le nouvel arrivant ne fournit que des capacités de calcul et de mémoire, sans posséder de variables ou de facteurs : dans ce cas

le nouvel arrivant n'a aucun voisinage immédiat dans le graphe de facteurs et considérera tous les autres agents comme des voisins.

Définissons la notion de voisinage comme suit :

Définition 4. *Etant donné l'affectation courante μ, le* voisinage *d'un agent a_k est défini comme :*

- $\mathcal{A}[a_k] = \{a_\ell \mid \exists (x_i, f_j) \in E, \mu(x_i) = a_k, \mu(f_j) = a_\ell\} \cup \{a_\ell \mid \exists (x_i, f_j) \in E, \mu(f_j) = a_k, \mu(x_i) = a_\ell\} \cup \{a_k\}$*, si a_k héberge au moins un élément du graphe de facteurs,*
- $\mathcal{A}[a_k] = \mathcal{A}$*, sinon.*

Nous définissons également les ensembles d'arcs et de nœuds voisins : $E[a_k] = \{(x_i, f_j) \mid \mu(x_i), \mu(f_j) \in \mathcal{A}[a_k]\}$; $V_x[a_k] = \{x_i \mid (x_i, f_j) \in E[a_k]\}$; et $V_f[a_k] = \{f_j \mid (x_i, f_j) \in E[a_k]\}$.

Dans la suite nous présentons deux approches pour adapter les affectations en ligne :

(i) résoudre ILP-FGDP sur une sous-partie du graphe de facteurs (section 4.2) ;

(ii) résoudre un problème de décision centré agent (le problème de décision du nouvel arrivant, cf. section 4.3).

4.2 Résolution par restriction du problème

L'idée ici est d'encoder le problème de révision du déploiement comme une version limitée d'ILP-FGDP, restreinte au voisinage du nouvel arrivant. Pour chaque agent, le problème consiste à choisir les éléments à héberger, en respectant les capacités de communication et de mémoire.

Problème 4 (ILP-FGDP$[a_k]^+$)**.** *Le problème ILP-FGDP$[a_k]^+$ correspond à ILP-FGDP restreint à l'ensemble d'agents $\mathcal{A}[a_k]$ et au graphe de facteurs $\langle V_x[a_k], V_f[a_k], E[a_k]\rangle$.*

Ce problème peut être résolu soit par un seul agent (si la taille du problème n'est pas trop large), soit par les agents composant le voisinage du nouvel arrivant. Dans les deux cas, cela ne nécessite qu'une connaissance limitée et locale sur le DCOP, ce qui est approprié pour des systèmes larges et complexes. Avant de résoudre ce problème, les agents doivent partager leurs éléments et coûts de communication avec les agents impliqués dans la réparation. Lorsqu'un nouvel arrivant ne possède aucun élément du graphe de facteur (et est donc considéré comme étant connecté à tous les autres agents) , le problème revient à résoudre ILP-FGDP sur la totalité du graphe, ce qui ne peut être raisonnablement réalisé en terme de temps et de charge réseau.

Dans le cas d'une résolution distribuée par tous les membres du voisinage, plusieurs techniques d'optimisation distribuée peuvent respecter ces exigences. Par exemple, nous pouvons considérer l'algorithme du simplexe distribué conçu pour l'allocation de ressources multi-agents [2], préservant le même codage que ILP-FGDP, ou des méthodes par décomposition duale comme la méthode AD^3 [8], qui nécessite d'encoder le problème en utilisant uniquement des contraintes sous forme de *tractable high order potentials* [12], et d'implanter un processus distribué de décodage de la solution à la relaxation continue pour affecter des valeurs entières aux variables de décision. Néanmoins, tout en fournissant des solutions de très bonne qualité, ces deux méthodes peuvent nécessiter plusieurs tours (et ainsi des échanges de messages) pour atteindre ces solutions de qualité. Par exemple, d'après une conjecture dans [2], la complexité en temps moyenne du simplexe distribué en fonction du le diamètre du graphe ($\mathcal{O}(diam(FG))$), avec une charge réseau polynomiale.

4.3 Résolution par appel à propositions

Afin d'éviter la charge de communication et de calcul potentiellement importante induite par les approches précédentes, nous pouvons considérer une approche centrée sur le nouvel arrivant : ce dernier émet un appel à propositions pour déplacer certains calculs. En se basant sur l'ensemble des propositions reçues et sa capacité mémoire, le nouvel arrivant doit choisir quels éléments du graphe de facteurs accueillir. Formulons ce problème de décision, comme suit :

Problème 5 (NDP)**.** *Etant donné un nouvel arrivant et un ensemble de calculs proposés à la migration par les voisins,* **le problème de décision du nouvel arrivant (NDP)** *revient à choisir les calculs à héberger parmi les calculs proposés, de telle sorte que la charge de communication soit minimisée et les contraintes de mémoire respectées.*

Chaque voisin a_ℓ de $\mathcal{A}[a_k]$ envoie ses propositions dans un message de la forme

$\langle V^{\ell\to k}, E^{\ell\to k}, \mathbf{com}\rangle$, où : $V^{\ell\to k} \subset V_x \cup V_f$ est l'ensemble des éléments proposés ; $E^{\ell\to k} = \{(e_i, e_j) \mid (e_i, e_j) \in E, e_i \in V^{\ell\to k} \text{ or } e_j \in V^{\ell\to k}\}$ est l'ensemble des arcs connectés à des éléments proposés ; et $\mathbf{com}$ est le coût de communication restreint aux éléments de $E^{\ell\to k}$.

Notons $V^k = \bigcup_\ell V^{\ell\to k}$ et $E^k = \bigcup_\ell E^{\ell\to k}$. Le coût de communication $\mathbf{com}(e_i, e_j)$peut être évalué en utilisant uniquement les coûts envoyés par les proposants. Soit e_i^k une variable binaire spécifiant si le nouvel arrivant a_k choisit ou non d'héberger le calcul e_i. Le coût de sélection d'un calcul peut être formulé comme suit :

$$\sum_{(e_i,e_j)\in E^k} \mathbf{com}(e_i, e_j)(e_i^k + e_j^k - 2.e_i^k.e_j^k) \tag{15}$$

$$- \sum_{(e_i,e_j)\in E^k} \mathbf{com}(e_i, e_j).e_i^k.e_j^k \tag{16}$$

qui est composé de la somme des coûts de communication pour l'ensemble des arcs qui sont hébergés par deux agents distincts dans la nouvelle distribution (15), i.e. ceux pour qui e_i^k XOR e_j^k est vrai ; moins le coût de communication pour l'ensemble des arcs maintenant hébergés sur le même agent (16), i.e. ceux pour qui x_i^k AND e_j^k est vrai. Cette somme peut être simplifiée et être utilisée comme objectif d'optimisation pour le nouvel arrivant a_k, comme suit :

$$\min_{e_i^k, e_j^k} \sum_{(e_i,e_j)\in E^k} \mathbf{com}(e_i, e_j)(e_i^k + e_j^k - 3.e_i^k.e_j^k) \tag{17}$$

$$\textbf{avec} \quad \sum_{e_i\in V^k} \mathbf{mem}(e_i).e_i^k \leq \mathbf{cap}(a_k) \tag{18}$$

Problème 6 (IQP-NDP)**.** *NPD est modélisé par* **IQP-NDP***, le programme quadratique en* $0/1$ *composé de l'objectif quadratique (17) et des contraintes linéaires (18).*

Ce problème tombe dans le cadre du problème quadratique du sac à dos (QKP). En effet, l'équation (17) peut être reformulée comme suit :

$$\min_{e_i^k, e_j^k} \sum_{e_i\in V^k} e_i^k.\mathbf{p}(e_i) + \sum_{\substack{e_i\in V^k \\ e_j\in V^k}} e_i.e_j.\mathbf{P}(e_i, e_j) \tag{19}$$

avec

$$\mathbf{p}(e_i) = \sum_{e_j\in V^{k+}} \mathbf{com}(e_i, e_j), \quad \forall e_i \in V^k \tag{20}$$

$$\mathbf{P}(e_i, e_j) = \begin{cases} -3.\mathbf{com}(e_i, e_j), & \text{si } (e_i, e_j) \in E^k \\ 0, & \text{sinon} \end{cases} \tag{21}$$

et $V^{k+} = \{e_i \mid (e_i, e_j) \in E^k \text{ ou } (e_j, e_i) \in E^k\}$ est l'ensemble des éléments connectés à au moins un arc dans E^k, même ceux qui ne sont pas déplaçables, et donc non proposés à la migration.

Le problème QKP peut être efficacement résolu par programmation dynamique, mais sans garanties d'optimalité [7]. Ne nécessitant que $\mathcal{O}(\mathbf{cap}(a_k).|V^k|)$ en espace mémoire, et $\mathcal{O}(\mathbf{cap}(a_k).|V^k|^2)$ en temps de calcul, une telle approche semble réaliste dans notre cas.

En plus, au lieu d'utiliser la totalité de sa capacité mémoire $\mathbf{cap}(a_k)$, l'agent a_k peut aussi limiter sa capacité d'accueil offerte (e.g. en la fixant à la moyenne d'utilisation dans le voisinage) afin de ne pas héberger plus de calcul que les autres, en général. Afin de respecter la contrainte (8) dans ILP-FGDP, nous pouvons ajouter les contraintes suivantes à la décision de l'agent a_k.

$$|\phi^{-1}(a_k)| + \textstyle\sum_{e_i\in V^k} e_i^k \geq 1 \tag{22}$$

$$\forall a_\ell \in \mathcal{A}[a_k] \setminus a_k, \ |\mu^{-1}(a_\ell)| - \sum_{e_i\in V^{\ell\to k}} e_i^k \geq 1 \tag{23}$$

Notons également que du point de vue du proposant, la décision de choisir quels éléments proposer est également un problème en lui-même, qui peut impacter fortement la décision du nouvel arrivant. Dans cet article, nous considérons que les agents proposent l'intégralité de leur calculs déplaçables, qui ne leur appartiennent pas.

5 Adaptation à la disparition d'agents

Dans les environnements ouverts, certains agents peuvent être retirés ou tomber en panne. Dans ce cas nous avons besoin de réparer le déploiement du graphe de facteurs : les facteurs et variables possédés par l'agent partant disparaissent tout simplement du graphe de facteurs, alors que les éléments hébergés sur cet agent doivent être déplacés. Nous pouvons distinguer deux cas : le retrait *planifié* (l'équipement peut déplacer ses calculs avant de partir), et le retrait

à chaud (l'équipement disparait sans avoir migré ses calculs hébergés sur d'autres agents).

Le retrait *planifié* est équivalent à l'arrivée d'agent, mais l'allocation des éléments est effectuée au sein du voisinage, excluant l'agent à disparaître. Ici, résoudre IQP-NDP n'est pas pertinent car aucun agent n'est au centre de la décision. Le retrait *à chaud* est plus complexe, et implique des conséquences techniques. Nous supposons que les agents sont conscients de la disparition des agents de leur voisinage (via des signaux de type *keepalive*, potentiellement en utilisant les messages déjà envoyé par Max-Sum, pour économiser des messages et de l'énergie). Dans de tels cas, il est alors possible de résoudre ILP-FGDP pour le nouvel ensemble d'agents, mais pas au sein d'un seul agent pour des raisons de complexité. Ainsi, nous optons encore pour une approche plus locale. L'idée est de résoudre ILP-FGDP restreint aux voisins directs de l'agent disparu. Notons $V_x[a_k]^- = V_x[a_k] \setminus \phi_x^{-1}(a_k)$, $V_f[a_k]^- = V_f[a_k] \setminus \phi_f^{-1}(a_k)$ et $E[a_k]^- = E[a_k] \cap (V_x[a_k]^- \times V_f[a_k]^-)$ les ensembles d'éléments et d'arcs impliqués dans la redistribution. Nous notons $\mathbf{cap}^-(a_k) = \mathbf{cap}(a_k) - \sum_{e_i \in \phi^{-1}(a_k)} \mathbf{mem}(e_i)$ la capacité mémoire d'un agent a_k, obtenue ou soustrayant de $\mathbf{cap}(a_k)$ l'empreinte mémoire des calculs hébergés par a_k mais non impliqués dans la redistribution.

Problème 7 (ILP-FGDP$[a_k]^-$)**.** *Le problème ILP-FGDP$[a_k]^-$ correspond à ILP-FGDP restreint à l'ensemble d'agents $\mathcal{A}[a_k] \setminus \{a_k\}$ et au graphe de facteurs $\langle V_x[a_k]^-, V_f[a_k]^-, E[a_k]^- \rangle$, où $\mathbf{cap}(a_k)$ est remplacé par $\mathbf{cap}^-(a_k)$.*

Pour résoudre ILP-FGDP$[a_k]^-$ certaines exigences doivent être remplies : (a) les agents doivent savoir comment calculer les éléments qui partagent un arc avec un élément qu'ils hébergent et les coûts de communication correspondants ; (b) les agents doivent savoir à quels agents envoyer des messages ; (c) les agents doivent avoir assez de mémoire pour héberger les éléments orphelins.

L'exigence (a) implique de fournir cette information au moment du déploiement initial et des phases de réparation. L'exigence (b) repose sur un mécanisme de découverte et d'exposition de services, par exemple lorsqu'un nouvel agent est ajouté. Enfin, l'exigence (c) peut ne pas être respectée si le voisinage de l'agent qui disparaît n'a pas assez de mémoire au total. Dans ce cas, le voisinage peut être étendu par des voisins de voisins jusqu'à ce que la mémoire soit suffisante. Une fois ces exigences respectées, les agents peuvent résoudre ILP-FGDP$[a_k]^-$ sur l'ensemble limité d'éléments et sur les agents voisins, comme dans le cas d'un nouvel arrivant (voir section 4).

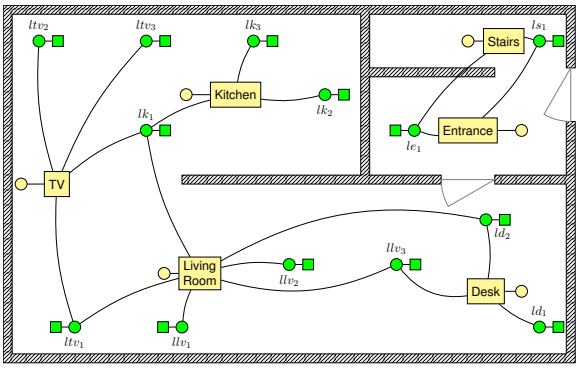


FIGURE 2 – Plan de la maison intelligente simulée, avec les modèles physiques, les actuateurs et leurs coûts respectifs.

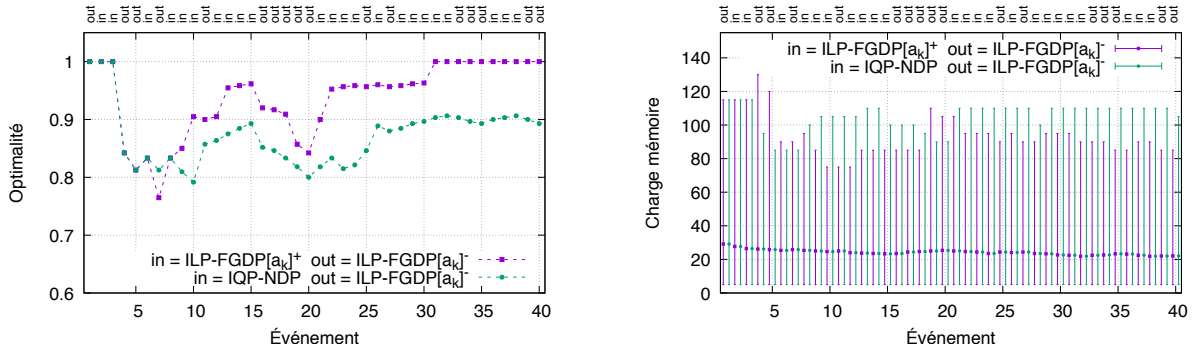
6 Evaluation expérimentale

Afin d'évaluer les performances des techniques de réparation proposées, nous avons simulé une maison intelligente où les équipements exécutent Max-Sum afin de trouver la configuration optimale au regard des préférences utilisateurs et de la consommation énergétique, comme dans [11].

Dans nos simulations, deux types d'événements peuvent survenir : arrivée d'équipement (in) et retrait à chaud d'équipement (out). Dans le cas de l'arrivée d'équipement, nous résolvons soit ILP-FGDP$[a_k]^+$, grâce à un solveur classique (centralisé) au sein d'un équipement (nous utilisons GLPK [2]), comme défini en section 4.2, soit IQP-NDP, grâce à un programme dynamique dédié (implémenté en Python dans notre simulateur) [3], comme défini en section 4.3. En cas de retrait d'équipement, comme discuté en section 5, ILP-FGDP$[a_k]^-$ est résolu en utilisant GLPK. Quel que soit le type d'événement, la meilleure solution à ILP-FGDP est également calculée (de manière centralisée avec GLPK, sur l'ensemble du graphe) pour évaluer la qualité des méthodes de réparation.

2. GLPK (GNU Linear Programming Kit) est un solver LP et MIP : https ://www.gnu.org/software/glpk/

3. De telles différences d'implémentation sont la raison de ne pas faire de comparaisons en terme de temps de calcul.



(a) (b)

FIGURE 3 – Optimalité (3a), et occupation mémoire moyenne, min et max (3b) en cours de simulation.

Dans une première série d'expériences, nous simulons une maison initialement composée de 13 actuateurs (ampoules intelligentes et leurs coûts énergétiques respectifs), 6 modèles physiques (un par espace de vie), et 5 règles utilisateurs. Le graphe de facteurs correspondant, présenté par la figure 2 (par soucis de lisibilité les règles utilisateurs ne sont pas représentées) possède 19 variables, 24 facteurs et 43 arcs. Concernant les coûts de communication, $a = 5$, $b = 100$, et la capacité mémoire par défaut des agents($\mathbf{cap}$) est fixée à 200 unités mémoire (une unité représente l'espace requis pour stocker une valeur de coût, e.g. 32 bits). La figure 3 trace les performances des solutions de réparation sur un scénario scripté où des équipement sont ajoutés ou retirés en cours d'exécution. Chacun de ces 40 événements est suivi d'une phase de réparation utilisant les méthodes précédemment discutées. La figure 3a montre l'optimalité des déploiements réparés, qui est calculée comme le rapport entre le coût de la solution réparée et le coût de la meilleur solution possible (optimum d'ILP-FGDP). Clairement les événements out tendent à dégrader l'optimalité du déploiement, tout en maintenant un niveau très compétitif, en comparaison d'un re-calcul complet du déploiement sur l'ensemble du graphe de facteurs. Il est intéressant de remarquer qu'avec avec les deux méthodes de réparation proposées les événements in améliorent le niveau d'optimalité. Cela signifie que dans un système réel où les événements out sont à peu près tous compensés par des événements in, le déploiement devrait garder un très bon niveau de qualité. La figure 3b présente l'occupation mémoire moyenne (ainsi que le min et le max) sur tout les équipements, après chaque événement. Alors que nos approches ne sont pas

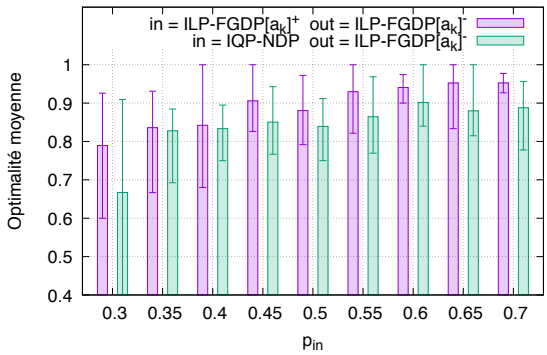


FIGURE 4 – Influence de p_{in} sur l'optimalité

spécifiquement conçues pour assurer une occupation mémoire équitable sur tous les équipements, les deux méthodes ne mènent pas à une accumulation excessive de calcul sur un seul équipement. Résoudre ILP-FGDP$[a_k]^+$ est clairement un meilleur choix suivant ce critère, ce qui peut être expliqué par le fait qu'il permet le déplacement de calcul sur tout le voisinage, alors que résoudre IQP-NDP ne permet des déplacements que vers le nouvel arrivant.

Dans une seconde série d'expériences, nous simulons une maison plus grande avec 23 actuateurs, 9 modèles physique et 9 règles utilisateur. Ici, nous évaluons la robustesse de chacune des techniques de réparation avec de plus en plus de pannes d'équipements. La figure 4 montre les performances moyennes sur 10 simulations après 20 événements, en terme d'optimalité (calculée comme précédemment), en faisant varier la probabilité d'avoir certains types d'événements. A chaque génération d'événement, son type est déterminé suivant la proba-

bilité p_{in}, i.e.la probabilité que l'événement soit in. Plus p_{in} est élevée, plus aisée est l'adaptation, car plus d'équipements sont disponibles. L'approche ILP-FGDP$[a_k]^+$ combinée à ILP-FGDP$[a_k]^-$ présente une très bonne résilience, car elle offre plus de 80% d'optimalité avec $p_{\mathsf{in}} \geq 0.35$ (environ 2 pannes pour 1 arrivée). L'approche IQP-NDP combinée avec ILP-FGDP$[a_k]^-$ est toujours entre 5 et 15% en deçà.

An final, l'approche ILP-FGDP$[a_k]^+$ présente une meilleure optimalité, mais nécessite bien plus d'échange d'information pour être calculée, alors que IQP-NDP est en moyenne 10% moins bonne en coût de communication du déploiement, et équivalente en terme d'occupation mémoire moyenne.

7 Conclusions

Nous avons discuté et analysé le problème du déploiement des éléments d'un graphe de facteurs sur une infrastructure ouverte composée d'équipements aux capacités limitées, les agents. Nous modélisons ce problème de déploiement comme un problème de partitionnement de graphe, encodé en programme linéaire en $0/1$, devant être résolu chaque fois que l'utilisateur pousse de nouvelles règles (facteurs) dans le système. Nous avons également proposé plusieurs techniques de réparation pour faire face à l'arrivée et la disparition d'agents survenant en cours d'exécution, en résolvant le problème de déploiement initial sur un ensemble restreint d'agents et d'éléments du graphe de facteurs (ILP-FGDP$[a_k]^+$ et ILP-FGDP$[a_k]^-$), ou mettant en œuvre une approche centrée sur le nouvel arrivant (IQP-NDP). Les expérimentations que nous avons effectuées sur un environnement simulé montrent que les techniques locales et approchées que nous proposons sont compétitives en terme d'optimalité du déploiement, en comparaison d'un déploiement réinitialisé et optimal. Dans cette article nous nous sommes focalisé sur la qualité des solutions obtenues par ces techniques de réparation locale. Comme discuté dans les sections concernées, ces techniques sont distribuables. Cela nécessite toutefois de concevoir des algorithmes de codage et de décodage idoines afin d'obtenir une adaptation pleinement distribuée des graphes de facteurs. De plus, comme nous l'avons mentionné lors de la prise en charge des nouveaux arrivants, la décision de choisir quels éléments proposer à la migration est également une piste de travail que nous n'avons pas explorée. Ici, nous pourrions considérer de baser les décisions des agents sur des préférences ou sur l'historique de calculs passés et de messages échangés, afin d'évaluer les éléments à envoyer au nouvel arrivant. Nous laissons cette réflexion pour de futurs travaux.

Références

[1] M. Boulle. Compact mathematical formulation for graph partitioning. *Optimization and Engineering*, 5(3) :315–333, 2004.

[2] M. Bürger, G. Notarstefano, F. Bullo, and F. Allgöwer. A distributed simplex algorithm for degenerate linear programs and multi-agent assignments. *Automatica*, 48(9) :2298 – 2304, 2012.

[3] J. Cerquides, A. Farinelli, P. Meseguer, and S. D. Ramchurn. A tutorial on optimization for multi-agent systems. *The Computer Journal*, 57(6) :799–824, 2014.

[4] N. Fan and P. M. Pardalos. Linear and quadratic programming approaches for the general graph partitioning problem. *Journal of Global Optimization*, 48(1) :57–71, 2010.

[5] A. Farinelli, A. Rogers, and N. R. Jennings. Agent-based decentralised coordination for sensor networks using the max-sum algorithm. *Autonomous Agents and Multi-Agent Systems*, 28(3) :337–380, May 2014.

[6] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS'08)*, pages 639–646, 2008.

[7] Franklin Djeumou Fomeni and Adam N. Letchford. A dynamic programming heuristic for the quadratic knapsack problem. *INFORMS Journal on Computing*, 26(1) :173–182, 2014.

[8] André F. T. Martins, Mário A. T. Figueiredo, Pedro M. Q. Aguiar, Noah A. Smith, and Eric P. Xing. Ad3 : Alternating directions dual decomposition for map inference in graphical models. *Journal of Machine Learning Research*, 16 :495–545, 2015.

[9] J. E. Mitchell. *Handbook of Applied Optimization*, chapter Branch-and-Cut Algorithms for Combinatorial Optimization Problems, pages 65–77. Oxford University Press, 2002.

[10] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *International Joint Conference on Artificial Intelligence (IJCAI'05)*, pages 266–271, 2005.

[11] P. Rust, G. Picard, and F. Ramparany. Using message-passing DCOP algorithms to solve energy-efficient smart environment configuration problems. In *International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2016.

[12] Daniel Tarlow, Inmar E. Givoni, and Richard S. Zemel. Hop-map : Efficient message passing with high order potentials. In Yee W. Teh and D. M. Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS-10)*, volume 9, pages 812–819, 2010.

Modélisation et simulation des Systèmes de Transport Intelligents Coopératifs : un comparatif

J. Sobieraj[a] jeremy.sobieraj@ibisc.univ-evry.fr
G. Hutzler[a] guillaume.hutzler@ibisc.univ-evry.fr
H. Klaudel[a] hanna.klaudel@ibisc.univ-evry.fr

[a]IBISC, Univ Evry, Université Paris-Saclay, 91025, Evry, France

Résumé

Les Systèmes de Transport Intelligents Coopératifs (C-ITS), notamment les véhicules autonomes, représentent le défi de demain dans le monde de l'automobile. Ces systèmes peuvent être assimilés à des ensembles d'agents autonomes coopératifs ayant une perception locale de leur environnement et une interaction avec celui-ci. La problématique majeure est d'obtenir un système sécurisé, tout en proposant une conduite fluide et confortable. Pour répondre à cette problématique, une multitude de simulateurs informatiques, notamment de type trafic ou multi-agent, existent pour modéliser l'ensemble des agents et de leur environnement, simuler et analyser leurs comportements. Nous proposons un comparatif de ces simulateurs à partir d'un ensemble de critères d'évaluation.

Mots-clés : *Simulation multi-agents, Communication, Coopération, Interaction, Trafic, Véhicules autonomes*

Abstract

Cooperative Intelligent Transport Systems (C-ITS), particularly autonomous cars, represent a major challenge in the automotive world. These systems can be assimilated to sets of cooperative autonomous agents having a local perception of their environment and an interaction with it. The major problem is to get a safe system, while offering a fluid and comfortable driving. To address this question, lots of computer simulators, especially traffic or multi-agent, exist to model all the agents and their environment, simulate and analyze their behaviors. We propose a comparison of these simulators based on evaluation criteria.

Keywords: *Multi-agent simulation, Communication, Cooperation, Interaction, Traffic, Autonomous cars*

1 Introduction

La voiture est le mode de transport le plus utilisé en France [4]. Il reste néanmoins le moyen de transport représentant le plus grande part d'accidents suivi des deux roues et des piétons [3]. On pourrait donc imaginer un véhicule dont la principale fonction, la conduite, ne serait pas tenue uniquement par l'homme : c'est le cas des véhicules autonomes. Ces véhicules intelligents, qui vont progressivement remplacer le parc actuel des véhicules, pourraient respecter les critères suivants :

- La *sécurité* : garantir qu'à chaque instant, le véhicule respecte les distances de sécurité et limite au maximum le risque d'accident ;
- La *fluidité* : optimiser la vitesse de chaque véhicule et réduire le phénomène de stop-and-go (embouteillages), impliquant une meilleure consommation de carburant ;
- Le *confort* : proposer aux passagers une conduite souple leur permettant de se sentir en sécurité.

On distingue actuellement deux approches pour les véhicules intelligents. La première, dans laquelle la voiture est totalement indépendante et où seuls les capteurs embarqués fournissent les informations nécessaires pour conduire (position des autres véhicules, des panneaux de signalisation, obstacles...). La seconde, dans laquelle est ajoutée la possibilité de communiquer avec d'autres véhicules (V2V) ou l'infrastructure de la route (V2I / I2V) [17].

La seconde approche apparaît comme plus adaptée pour un environnement constitué de véhicules intelligents et non intelligents. En effet, les véhicules non intelligents (conducteurs humains) peuvent être assimilés à des agents ayant une connaissance imparfaite (impossible d'observer l'environnement local entièrement à chaque instant). De plus, l'environnement routier est constitué d'évènements imprévus qu'il est nécessaire d'anticiper pour assurer la sécurité et la fluidité sur la route (accident, obstacle sur la route, embouteillage...).

Les agents vont donc prendre des décisions

qui dépendront d'une part de l'environnement, d'autre part de la décision des autres véhicules. L'interaction entre les agents, présente dans cette seconde approche, apparaît donc comme une propriété fondamentale pour un système constitué d'agents ayant une perception différente de l'environnement.

L'étude comportementale des agents et de leurs interactions avec l'environnement peut être faite par simulation informatique afin de répondre aux critères de sécurité, d'écologie et de confort dans un système temps-réel et ouvert. Il reste cependant difficile de trouver un simulateur qui prenne en compte de manière satisfaisante à la fois la modélisation routière et la définition de protocoles de communication et de coopération entre agents, tout en étant accessible au monde académique. L'objectif de cet article est donc d'établir un comparatif aussi large que possible des solutions actuelles pour la réalisation et la simulation de véhicules autonomes communicants.

Pour cela, nous définirons les différentes catégories de simulateurs pouvant représenter un tel système et nous définirons les critères d'évaluation permettent la comparaison entre chaque simulateur (section 2). Ces simulateurs seront présentés (section 3) puis analysés entre eux (section 4) afin d'établir un tableau comparatif résumant les critères pour chaque simulateur (section 5).

2 Exigences

2.1 Catégories de simulateurs

Différentes approches ont été développées autour de la simulation des véhicules autonomes parmi lesquelles on pourrait distinguer trois grandes familles.

Une première approche consiste à reproduire de façon réaliste le comportement des véhicules en termes de respect des lois de la physique en fonction de leurs paramètres spécifiques. La simulation peut évaluer différents aspects comme l'étude d'une trajectoire précise [30] ou encore, dans le cadre des véhicules communicants, une étude de la fiabilité et de l'intégrité de l'information transmise entre les véhicules [6].

Une deuxième approche s'intéresse en priorité au trafic, notamment son évolution au cours du temps à travers trois points de vue [29] : macroscopique, microscopique et mésoscopique. En particulier, dans le cas de l'étude microscopique (véhicules traités de façon individuelle dans une petite zone), cette approche permet la mise en circulation des véhicules à partir des lois de poursuite, dont la majorité est garantie sans collision tout en respectant les distances de sécurité (IDM [28]), et des modèles de changement de voie (MOBIL [18]).

Enfin, la troisième approche est orientée agents dans le sens où on assimile un véhicule à un agent réagissant selon la perception de son environnement [12]. Chaque agent a également la possibilité de communiquer avec d'autres agents ou avec son environnement pour échanger une information ou négocier une future décision à prendre.

La combinaison de ces trois approches pourrait permettre de constituer un simulateur "idéal". Toutefois, il est très rare de trouver un simulateur qui mette en oeuvre les trois approches à la fois. Par exemple, si on considère un simulateur orienté agents, il est souvent difficile de représenter l'aspect trafic. De même qu'un simulateur conçu pour bien représenter le trafic ne présentera pas directement le niveau de réalisme adéquat. Pour des raisons similaires, un simulateur reproduisant bien le réalisme des véhicules va souvent manquer de l'aspect agent. Selon les approches, les simulateurs vont permettre de modéliser et d'étudier des propriétés différentes à des niveaux de réalisme et d'échelle différents. Nous allons donc présenter un ensemble de simulateurs couvrant chacun un sous-ensemble plus ou moins grand de ces trois approches, en les comparant à partir de critères d'évaluation. Ainsi, chaque utilisateur pourra s'orienter vers l'outil correspondant le plus à ses attentes, de par son sujet d'étude et ses exigences techniques.

2.2 Critères d'évaluation

Un ensemble de critères d'évaluation sont nécessaires afin d'orienter la recherche vers un simulateur respectant les approches citées précédemment. Nous les avons catégorisés afin de déterminer l'ensemble des aspects qui peuvent être intéressants pour effectuer ce choix.

Caractéristiques générales. Il s'agit ici d'analyser les caractéristiques principales du logiciel, c'est-à-dire de savoir si le logiciel est gratuit, libre et multi-plateforme. Il est également important de voir si l'installation se fait instanta-

nément ou demande un ensemble d'étapes préliminaires.

Critères : Prix - Open-source - Multiplateforme - Installation

Prise en main. Lorsque le simulateur est installé, il est important d'avoir une interface et une documentation claires pour une bonne prise en main. De plus, en cas de problème, il est nécessaire de savoir si il est possible de contacter les concepteurs et/ou la communauté du simulateur concerné.

Critères : Simplicité - Documentation - Communauté

Simulation et analyse. Pour lancer une simulation et effectuer son analyse, il peut être intéressant d'avoir un environnement de simulation permettant d'effectuer certains réglages rapidement et efficacement. Il faut également savoir comment nous pouvons observer et analyser notre simulation (aspect graphique, courbes, exportation dans un fichier...). Enfin, il est nécessaire d'analyser les performances du simulateur qui peuvent être limitées sur une machine moins performante que les machines actuelles.

Critères : Environnement de simulation - Sortie de simulation - Outils d'analyse - Performances

Simulation de trafic. Dans le cas de l'échelle microscopique, il s'agit de voir si l'implémentation des lois de poursuite et des changements de voie a été faite dans le simulateur, les possibilités en termes de modélisation des routes ainsi que la diversité des moyens de transport que l'on peut modéliser.

Critères : Lois de poursuite existantes - Modélisation de la route à partir de données géographiques - Modélisation manuelle de routes - Modélisation de voies particulières - Diversité des moyens de transports - Création de nouveaux moyens de transport

Simulation multi-agents. L'aspect agent est considéré comme très important et il est donc nécessaire d'avoir une diversité d'agents aux comportements différents. De plus, les interactions entre eux (communication, négociation) doit être également possible.

Critères : Diversifier le comportement des agents - Ajout de nouveaux comportements - Communications - Négociations

Aller plus loin. Il est évident que nous n'allons pas trouver un simulateur correspondant exactement à l'ensemble de nos attentes. Il est donc nécessaire de voir si celui-ci offre une liberté de personnalisation et de création si il est possible d'implémenter de nouveaux éléments facilement et efficacement. Ces nouveaux éléments, sous forme de modules, peuvent être intéressants pour la communauté du simulateur choisi et enrichir celui-ci.

Critères : Liberté de personnalisation et de création - Création d'extensions (plug-ins) - Clarté du code source

3 Présentation des simulateurs

3.1 Choix et justifications

Voici la liste des différents simulateurs par catégorie :

- *Simulateurs de trafic* : Aimsun, MovSim et SUMO ;
- *Simulateurs multi-agents* : AnyLogic, GAMA et MatSIM ;
- *Simulateur de conduite* : OpenDS.

Le choix des simulateurs de trafic s'est fait selon leur popularité dans le domaine du trafic. Concernant les simulateurs de systèmes multi-agents, nous avons choisi ceux proposant des premiers outils de modélisation et d'étude du trafic. De plus, nous avons fait le choix de présenter quelques outils payants généralement plus complets afin d'établir des comparaisons intéressantes. Chaque simulateur représente donc chacun une approche différente pour modéliser et simuler les Systèmes de Transport Intelligents. Nous avons toutefois choisi des simulateurs proposant des mises à jour fréquentes et/ou régulières.

3.2 Historique

Aimsun [10]. L'*Advanced Interactive Microscopic Simulator for Urban and non-urban Networks* (Aimsun) est un logiciel de modélisation et de simulation de transports créé en 1993 et développé en C++ par TSS-Transport Simulation Systems.

AnyLogic [8]. AnyLogic est un logiciel de simulation multi-méthode développé en Java conçu en 2000 par The AnyLogic Company.

GAMA [26]. GAMA (*Gis & Agent-based Modelling Architecture*) est un simulateur de systèmes multi-agents conçu en 2010 par l'unité de recherche UMMISCO (*Unité mixte internationale de Modélisation Mathématique et Informatique des Systèmes COmplexes*).

MatSIM [7]. Le *Multi-Agent Transport SIMulation*, permettant la représentation des systèmes de transport à grande échelle, a été créé vers 2005 par deux groupes de recherche dirigés par Kai Nagel et Kay W. Axhausen ainsi que la société Senozon, spécialisée dans les transports et la planification des systèmes de transport.

MovSim [29]. Le *Multi-model Open-source Vehicular-traffic SIMulator* (MovSim) a été initié en 2011 par Arne Kesting à l'Université Technique de Dresde en Allemagne.

SUMO [19]. Le logiciel de *Simulation of Urban MObility* (SUMO) a été développé en 2001 en collaboration entre le Centre d'Informatique Appliqué de Cologne (ZAIK) et l'Institut des Systèmes de Transport au Centre Allemand pour l'Aéronautique (DLR).

OpenDS [20]. OpenDS (*Open-source Driving Simulator*) est un simulateur de conduite en trois dimensions créé en 2012 par le groupe automobile du Centre allemand de recherche pour l'intelligence artificielle (DFKI GmbH).

4 Analyse comparative

Nous allons faire à présent une analyse comparative de chaque simulateur avec leurs possibilités mais aussi leurs limites. L'analyse critique de chaque simulateur se fait sur la base des critères détaillés précédemment.

Nous avons installé puis testé chaque simulateur sur une machine récente de type bureautique. Afin de bien explorer les fonctionnalités proposées, nous avons simulé puis analysé un premier modèle de route. Enfin, pour les simulateurs open-source, nous avons exploré le code source pour observer les possibilités de personnalisation et d'extension.

4.1 Caractéristiques principales

Prix. Seuls GAMA, MatSIM, MovSIM et SUMO proposent des outils totalement gratuits. OpenDS dispose d'une version gratuite ainsi que de versions plus complètes mais payantes. Aimsun propose plusieurs versions payantes mais il permet la simulation microscopique dans une version gratuite limitée (sans sauvegarde). Enfin, AnyLogic est payant mais possède une version gratuite pour une utilisation hors recherche (personnelle et éducation).

Open-source. Exceptés Aimsun et AnyLogic qui sont des logiciels propriétaires, tous les outils proposés sont open-source (et disponibles sur GitHub).

Multiplateforme. Tous les simulateurs fonctionnent sur les systèmes d'exploitation principaux (Windows, Linux, Mac OS).

Installation. L'installation de Aimsun, AnyLogic et GAMA se fait à partir d'un exécutable. MatSIM et OpenDS s'exécutent directement à partir d'un fichier au format .jar (Java). L'installation de SUMO et de MovSim, demandant quelques étapes préliminaires, se fait assez simplement et des efforts ont été faits au niveau de la documentation.

4.2 Prise en main

Documentation. Les simulateurs disposant de la documentation la plus riche sont Aimsun et AnyLogic (manuel PDF, tutoriels textuels et vidéos, vidéos de démonstration, études de cas, articles de recherche…). MatSIM propose un manuel de documentation très complet et des tutoriels textuels pour les utilisateurs ainsi qu'un guide pour les développeurs. La documentation de GAMA est complète avec un manuel, une base de données de chaque fonctionnalité du langage GAML ainsi qu'un grand nombre d'exemples de tous types. Le site officiel de SUMO possède un wiki complet ainsi qu'un tutoriel détaillé avec un premier exemple à modéliser. MovSim fournit une documentation basée sur des exemples afin de démarrer assez facilement sur une première simulation. De plus, l'ensemble des modèles implémentés se trouvent dans le livre très complet *Traffic Flow Dynamics* [29], recensant l'ensemble des modèles existants dans la littérature. La combinaison de ces deux documentations ainsi qu'un contact possible avec les concepteurs par mail rend la prise en main du logiciel plutôt accessible. On regrettera cependant l'absence d'une documentation de départ ou d'un wiki spécialisé pour le simulateur. Concernant OpenDS, la documentation dépend de la version choisie. Ainsi, la version gratuite propose des tutoriels vidéo

tandis que les versions payantes donnent accès à une documentation sous forme d'un wiki.

Communauté. Pour Aimsun et AnyLogic, une équipe est disponible pour répondre aux différentes questions (délai de 48 heures pour AnyLogic). Ils est possible et simple de contacter les principaux concepteurs des autres simulateurs. En outre, Aimsun, MatSIM, Sumo et OpenDS possèdent des groupes de discussion où la communauté est active.

Simplicité. De par la présence d'une interface "user-friendly", Aimsun et AnyLogic se présentent comme les plus simples d'utilisation sans nécessité de regarder à tout prix la documentation. GAMA est simple d'utilisation notamment grâce à la présence de GAML qui est un langage orienté agent intuitif proche de Java et simple d'utilisation (seul un peu de documentation autour du langage est nécessaire). MatSIM, MovSIM, OpenDS et SUMO proposent une modélisation à partir de fichiers en entrée et une simulation à partir d'un environnement graphique (pour SUMO, possibilité également de passer par un terminal de commandes). Bien qu'assez simple d'utilisation, une documentation est nécessaire afin d'écrire les fichiers d'entrée correctement. Toutefois, OpenDS est moyennement simple d'utilisation en raison d'une documentation un peu faible sur la conception de ses fichiers en entrée.

4.3 Simulation et analyse

Environnement de simulation. Comme nous l'avons dit précédemment, Aimsun et AnyLogic disposent d'un environnement de simulation "user-friendly" constitué de nombreuses fonctionnalités classées par catégories. MovSim, SUMO, GAMA présentent une interface simple permettant de lancer, accélérer, ralentir, mettre en pause ou arrêter la simulation. En plus de ces éléments, AnyLogic et GAMA ont l'avantage de créer une interface adaptée à chaque utilisation (visuel seul, visuel et courbes…). L'interface de MatSIM et d'OpenDS permet uniquement de lancer la simulation en choisissant les fichiers d'entrée de notre choix.

Sortie de simulation. Sur Aimsun tout comme sur AnyLogic, il est possible d'observer l'avancement de chaque moyen de transport en temps réel sur une route en deux ou trois dimensions, le tout dans un visuel très clair et personnalisable. Bien que l'aspect visuel ne soit pas aussi développé que pour les deux outils précédents, GAMA, MovSim et SUMO possèdent également cette fonctionnalité et peuvent également envoyer les résultats de la simulation dans un fichier de sortie. Sur OpenDS, il est possible d'observer la simulation en trois dimensions à partir d'un véhicule que l'on peut contrôler avec les touches du clavier. Sur l'outil d'origine, MatSIM peut uniquement mettre les résultats dans un fichier de sortie. Toutefois, une extension, Via [1], permet d'avoir un aspect graphique en deux dimensions personnalisable. Celle-ci est payante pour un usage recherche (gratuit et limité pour un usage personnel).

Outils d'analyse. Aimsun est constitué d'un grand nombre d'outils d'analyse prédéfinis. Il peut afficher plusieurs aspects de la route (densité, vitesse, pollution, consommation d'essence…) directement sur le visuel en deux dimensions ou encore via des courbes. AnyLogic, GAMA, OpenDS, MovSim peuvent étudier les résultats par l'intermédiaire de courbes ou directement sur la simulation (trajectoire du véhicule, vitesse…). GAMA peut effectuer ceci grâce à une extension permettant une étude de trafic routier [25]. MatSIM permet analyser via des courbes ou directement sur la simulation à partir de Via. SUMO manque d'un ensemble d'outils d'analyse permettant l'étude de ces résultats (nécessité d'effectuer l'analyse nous-mêmes à partir du fichier contenant les données).

Performances. Nous donnons un ordre de grandeur en nombre de kilomètres de voies (remplies de véhicules). Ainsi, Aimsun et AnyLogic permettent de simuler et d'observer facilement au moins 100 km de voies, au moins 20 km de voies pour GAMA, MatSIM, MovSim, SUMO et au moins 5 km de voies pour OpenDS.

4.4 Simulation de trafic

Lois de poursuite existantes. MovSim est l'outil proposant le plus large choix de lois de poursuite et de modèles de changement de voie (Gipps [14], IDM [28], MOBIL [18]…). SUMO utilise les modèles de Gipps mais des extensions permettent un modèle dérivé de Krauss [2]. Aimsun utilise les modèles de Gipps également. GAMA utilise un modèle dérivé d'IDM. AnyLogic, MatSIM et OpenDS n'utilisent pas de modèles de lois de poursuite.

Modélisation de la route à partir de données géographiques. Excepté MovSim, tous

les logiciels peuvent modéliser une route à partir de données géographiques. Le plus simple et le plus complet se trouve sur Aimsun et AnyLogic qui permet d'importer les routes de son choix (à partir de données OpenStreetMap (OSM), GIS, CAD…). Il est possible de retirer les routes qui ne nous intéressent pas afin d'avoir un modèle sur-mesure. GAMA, MatSIM et SUMO permettent d'importer des fichiers au format shapefile et de choisir les types de routes souhaités ou non (outil assez intuitif sur MatSIM et SUMO, moyennement intuitif sur GAMA). OpenDS peut récupérer des routes et bâtiments construits à l'aide du logiciel CityEngine (version payante).

Modélisation manuelle de routes. Excepté GAMA, tous les logiciels peuvent modéliser une route manuellement. Sur Aimsun et AnyLogic, la modélisation est extrêmement simple car son interface est basée sur le drag-n-drop et permet donc de construire tout type de route (autoroute, voie d'insertion, rond-point…) avec un simple clic de souris. MatSIM, MovSim et SUMO passent par un ensemble de fichiers au format XML. Ainsi, on peut imaginer la route de notre choix en l'assimilant à un graphe indiquant la direction des véhicules ainsi que le nombre de voies (MovSim respecte le standard OpenDrive [16]). Sur OpenDS, la modélisation manuelle peut se faire également via CityEngine.

Modélisation de routes particulières. Seuls Aimsun, AnyLogic et SUMO permettent de modéliser des routes particulières telles que les voies de bus, les rails et les passages pour piétons (également parkings pour Aimsun).

Diversité des moyens de transports. Aimsun, AnyLogic et SUMO proposent la plus grande variété de moyens de transport : tous types de quatre roues, deux roues, trains et piétons. GAMA et MatSIM permet de modéliser tous types de quatre roues (sauf les bus pour GAMA) et les piétons. Quant à MovSim et OpenDS, ils peuvent créer uniquement des véhicules à quatre roues (excepté les bus).

Création de nouveaux moyens de transport. Sur AnyLogic, bien qu'il possède une grande variété de transports, il est très simple de créer le moyen de transport de son choix en imposant des règles précises (voies autorisées, vitesse maximale, accélération). GAMA possède également cette fonctionnalité en utilisant le langage GAML (simple de définir un agent plus petit qu'une voiture pour un piéton par exemple). Il est assez simple de le faire sur MatSIM, MovSim, OpenDS et SUMO directement sur le code source. Quant à Aimsun, il n'est pas possible de créer de nouveaux moyens de transport.

4.5 Simulation de systèmes multi-agents

Diversifier le comportement des agents. Sur AnyLogic, GAMA et SUMO, il est simple de donner un comportement précis à chaque agent et de définir des probabilités (le chemin que doit effectuer le véhicule, vitesse maximale, probabilité de changer de voie, d'avoir un accident, de respecter les priorités…). Sur Aimsun, il est possible de modifier facilement les caractéristiques d'une portion (le chemin que doit effectuer le véhicule, vitesse maximale, détection d'évènements…). Sur MovSim, il est possible de choisir parmi plusieurs lois de poursuite et de modifier les paramètres de chacun facilement. Sur MatSIM, cela se fait à partir du fichier d'entrée où l'on peut définir le chemin que doit effectuer le véhicule mais pas de prises de décisions différentes (comportement prédéfini). Sur OpenDS, il est assez simple de définir la trajectoire de chaque véhicule en leur donnant un ensemble de points à suivre.

Ajout de nouveaux comportements. Sur AnyLogic, il est possible de modéliser le comportement du véhicule selon notre choix : par une loi de poursuite que nous définissons nous-même ou par l'intermédiaire d'un arbre de décision. Sur GAMA, au delà de la loi de poursuite, il est possible d'affecter simplement à chaque véhicule son propre comportement. En effet, chaque agent entre dans une catégorie d'espèces (*species*) que l'on peut définir grâce au langage GAML. Sur MovSim, il est simple de créer une nouvelle classe Java définissant un nouveau comportement à suivre pour un ou plusieurs véhicules. Via un kit de développement, il est assez simple de définir ses propres lois de poursuite sur Aimsun. Sur OpenDS, on peut effectuer des modifications de vitesses et d'accélérations selon les situations à partir d'une classe Java existante. De plus, une mise à jour récente permet une fonctionnalité d'auto-pilotage afin d'avoir des véhicules autonomes. Sur SUMO, il n'est pas aussi simple de proposer son propre comportement pour chaque véhicule ou de simuler des situations imprévues (accidents, perte de communication, véhicule d'urgence…). Enfin, sur MatSIM, il est plus difficile de créer une diversité de comportements pour les agents. En

effet, l'outil paraît plus adapté pour une étude de flux de trafic ce qui fait que la flexibilité de l'outil se situe plus, par exemple, sur le choix de la densité de véhicule sur une portion de route ou encore le chemin qu'ils réalisent selon l'heure de la journée par exemple.

Communications. Exceptés MatSIM et OpenDS, tous les simulateurs ont la capacité de faire communiquer les différents types d'agents entre eux. Aimsun possède une extension permettant la conception de véhicules connectés (ITS). Sur SUMO, l'extension Veins [24], permet la communication entre les véhicules. Sur AnyLogic, il est possible d'envoyer des messages aux autres agents. Sur GAMA, une extension a été créée respectant les standards de la FIPA avec la possibilité d'utiliser le langage de communication ACL (*Agent Communication Language*) [13] ainsi que la négociation entre agents grâce à l'implémentation du modèle de négociation CNP (*Contract Net Protocol*) [23]. Enfin, une contribution récente [15] reprend la base de MovSim tout en ajoutant un aspect multi-agent et communicatif à celui-ci (non disponible pour le moment).

4.6 Aller plus loin

Liberté de personnalisation et de création. AnyLogic et GAMA sont les outils proposant la plus grande liberté de création. En effet, au-delà de la diversité des routes, moyens de transport et comportements possibles, il est possible de créer toutes sortes d'éléments (agent virtuel, obstacle, accidents, évènements imprévus) et de leur donner l'aspect visuel de notre choix. MovSim, bien que limité de termes de moyens de transport, offre une bonne liberté en termes de construction de routes et de véhicules aux comportements différents. Aimsun propose une grande liberté de personnalisation et de création que ce soit au niveau de la modélisation des routes, des moyens de transport ainsi qu'une assez bonne flexibilité sur les comportements de véhicules. OpenDS permet assez facilement de créer un contrôleur pour un véhicule. Toutefois, de par son aspect plus réaliste, il est plus difficile de lui attribuer des actions élémentaires (changement de voie, tourner à droite…). SUMO et MatSIM sont moins personnalisables que les autres simulateurs. Toutefois, SUMO offre une bonne liberté de création de routes et de moyens de transport différents.

Création d'extensions (plug-ins). Sur Aimsun, la construction de fonctionnalités sous forme de blocs rend le logiciel de simulation modulaire et il est possible d'obtenir une application en Java du modèle créé. GAMA, MatSIM, MovSim, OpenDS et SUMO permettent la création d'extensions à partir du code source. Sur Aimsun il est possible, grâce aux kits de développement, de créer ses propres extensions sur certains éléments (lois de poursuite, changements de voie…). On peut également automatiser un ensemble de tâches via des scripts en Python.

Clarté du code source. GAMA, MovSim et SUMO propose un code clair et bien commenté notamment dans les parties du code susceptibles d'être modifiées. MatSIM et OpenDS possèdent un code source bien commenté mais manquent parfois de clarté sur les noms de certaines classes. Aimsun et AnyLogic sont des logiciels propriétaires, le code source n'est donc pas disponible.

5 Tableau comparatif

Suite à une description détaillée de chaque simulateur, il convient de proposer un tableau comparatif (Table 1) résumant les possibilités de chacun selon les critères détaillés précédemment. Pour plus de visibilité, nous avons créé un code couleur. Ainsi, pour chaque case du tableau, plus la nuance de gris est sombre, plus la caractéristique est positive pour le simulateur correspondant à la case.

Grâce à ce code couleur, on remarque assez rapidement que deux simulateurs possèdent plus d'éléments positifs que les autres : *Aimsun* et *AnyLogic*. Toutefois, ce sont deux outils propriétaires et payants, deux caractéristiques pouvant être rédhibitoires. Si ce n'est pas le cas, *AnyLogic* s'adresse à des utilisateurs souhaitant plus de liberté pour créer des comportements différents et nouveaux pour les différents moyens de transport. *Aimsun* se présente comme un outil très riche en fonctionnalités de trafic prédéfinies (modélisation, simulation et analyse). D'autres alternatives à Aimsun existent parmi les simulateurs de trafic les plus populaires [22] : c'est le cas de PTV-VISSIM [11] ou encore Paramics [9].

Les autres simulateurs étudiés ne proposent pas de fonctionnalités aussi riches que les simulateurs précédents mais sont open-source et gratuits. Il convient donc de se diriger vers l'outil répondant le mieux aux critères de chacun en faisant un choix basé sur des compromis.

Dans ce contexte, *SUMO* et *MatSIM* sont deux bons outils de modélisation de trafic mais deviennent rapidement limités lorsque l'on souhaite gérer des situations "anormales" de trafic. De plus, *MatSIM* se présente plus comme un outil d'étude de flux de trafic. Quant à *MovSim*, il paraît plus complet pour une étude microscopique du système mais offre également des outils d'analyse plus complets que *SUMO*. Son extension [15] (pas encore disponible), ajoutant l'aspect agent et communication, l'enrichit encore plus et permet de proposer à la fois l'aspect trafic et agent. Concernant l'aspect agent, *GAMA* y répond totalement, d'une part grâce à son langage orienté agents GAML permettant une modélisation simple pour tout chercheur, même non-informaticien, d'autre part grâce à l'extension ajoutant le langage de communication FIPA-ACL [13] et l'aspect négociation grâce au *Contract Net Protocol* (CNP) [23]. Toutefois, l'outil de modélisation et d'analyse est limité par rapport à des simulateurs comme *SUMO* et *MovSim*. Il possède également l'avantage d'avoir une grande liberté de création. Enfin, *OpenDS* propose une alternative intéressante. En effet, bien que ce soit un simulateur de conduite, il permet la modélisation de véhicules plus réalistes à travers un environnement en trois dimensions. Son code est facilement modifiable et il devient facile de modifier le comportement de chaque véhicule. Il peut donc être intéressant de se diriger vers cet outil si l'on souhaite avoir des modèles plus réalistes tout en ayant une certaine flexibilité sur la création de nouveaux comportements.

Les simulateurs testés permettent de couvrir les aspects de tous les simulateurs existants. En effet, d'autres simulateurs de trafic existent mais ne remplissaient pas toujours les critères de départ (communauté trop petite, documentation faible, pas de communication entre les agents…). Concernant les simulateurs multi-agents, le choix de présenter *GAMA* s'est fait parmi une multitude de simulateurs proposés par un comparatif complet et assez récent [5], du fait de la présence d'une extension trafic. Il est tout à fait possible de se diriger vers un autre simulateur multi-agent si l'aspect agent est suffisant. Enfin, l'environnement de développement de calcul numérique *MatLab* [27], généralement associé à *Simulink* [21] est très utilisé dans l'industrie automobile. Il est bien adapté pour l'approche proposant des modèles de véhicules très réalistes mais sera plus limité sur les aspects trafic et agent.

6 Conclusion

Le choix d'un simulateur dans le contexte des Systèmes de Transport Intelligents ne paraît pas évident de par des approches divergentes. Il nous semblait donc naturel et utile de proposer un comparatif d'outils fournissant les propriétés fondamentales de ces approches (réalisme, trafic, agent, communications). Ces outils ont été résumés dans un tableau comparatif permettant de se diriger vers tel ou tel simulateur selon les besoins et l'approche souhaitée. Bien que propriétaires et payants, *Aimsun* répond le plus aux besoin d'un simulateur de trafic et *Anylogic* pour un simulateur agent ayant une grande flexibilité. Les outils gratuits et open-source suivent généralement une approche particulière : *OpenDS* pour l'aspect réalisme, *SUMO*, *MatSIM* et *MovSim* pour l'aspect trafic et *GAMA* pour l'aspect agent.

7 Perspectives

Dans le cadre des travaux que nous effectuons, nous nous intéressons à des aspects de coopération en particulier des protocoles de communication et de négociation entre les véhicules intelligents. Nous souhaitions un simulateur gratuit et open-source proposant un aspect agent et offrant une liberté de conception. Notre choix s'est donc porté sur le simulateur multi-agent *GAMA*. Toutefois, comme nous l'avons dit précédemment, l'aspect trafic paraît un peu limité.

Il convient donc d'enrichir l'aspect trafic de *GAMA* à partir de ce qui a déjà été implémenté. Nous souhaitons donc, à l'avenir :

- Proposer un outil simple permettant la construction de routes manuellement ;
- Ajouter un ensemble de lois de poursuites existants dans la littérature (de la même manière que *MovSim*) ;
- Simplifier la création de comportements différents pour chaque agent ;
- Donner le choix entre une génération aléatoire d'un flux d'agents et l'initialisation de cas précis.

Une première ébauche de cette extension existe déjà. A partir de paramètres en entrée, il est possible de créer un nombre de routes prédéfinies constituées d'un ensemble de voies (possibilité de déterminer la taille longitudinale et latérale de chaque voie). A partir d'une fonction simple, il est possible de placer un véhicule ou un groupe de véhicules sur la voie de notre choix. On peut définir le comportement que l'on

		Aimsun	AnyLogic	Gama	MatSIM	MovSim	OpenDS	SUMO
Date de création		1993	2000	2010	2005	2010	2012	2001
Langage de programmation		Langage propriétaire / C++	Langage propriétaire / Java	Java / GAML	Java	Java	Java	C++
Caractéristiques principales	Prix	Version gratuite limitée / Version payante	Version gratuite (hors recherche) / Version payante	Gratuit	Gratuit	Gratuit	Version gratuite / Version payante plus complète	Gratuit
	Open-source	Non	Non	Oui	Oui	Oui	Oui	Oui
	Multiplateforme	Oui	Oui	Oui	Oui	Oui	Oui	Oui
	Installation	Simple	Simple	Simple	Simple	Assez simple	Simple	Assez simple
Prise en main	Documentation	Très complète	Très complète	Complète	Complète	Assez complète	Assez complète	Complète
	Communauté	Très active	Très active	Active (concepteurs)	Active	Active (concepteurs)	Active	Active (concepteurs)
	Simplicité	Très simple	Très simple	Simple	Assez simple	Assez simple	Moyennement simple	Assez simple
Simulation et analyse	Environnement de simulation	Très bon	Très bon	Bon	Assez bon	Bon	Assez bon	Bon
	Sortie de simulation	Très complet	Très complet	Complet	Complet (outil payant)	Complet	Complet	Complet
	Outils d'analyse	Très complet	Complet	Complet	Complet (outil payant)	Complet	Complet	Limité
	Performances	Très performant	Très performant	Performant	Performant	Performant	Assez performant	Performant
Simulation de trafic	Lois de poursuite existantes	Une loi	Aucune	Une loi	Aucune	Grand nombre de lois	Aucune	Plusieurs lois
	Modélisation de routes à partir de données géographiques	Très simple et très complet	Très simple et très complet	Moyennement simple et assez complet	Assez simple et assez complet	Non	Assez simple et complet (logiciel tierce)	Assez simple et assez complet
	Modélisation manuelle de routes	Très simple et très complet	Très simple et très complet	Non	Assez simple et assez complet	Assez simple et assez complet	Assez simple et complet (logiciel tierce)	Assez simple et très complet
	Modélisation de voies particulières (arrêt de bus, rail, passage pour piétons, parking)	Très complet	Complet	Non	Non	Non	Non	Complet
	Diversité des moyens de transports	Quatre roues et deux roues (de tout type), trains, piétons	Quatre roues et deux roues (de tout type), trains, piétons	Voitures, poids lourds, piétons	Quatre roues (de tout type), piétons	Voitures, poids lourds	Voitures, poids lourds	Quatre roues et deux roues (de tout type), trains, piétons
	Création de nouveaux moyens de transport	Impossible	Très simple	Simple	Assez simple	Assez simple	Assez simple	Assez simple
Simulation multi-agents	Diversifier le comportement des agents	Simple et complet	Simple et complet	Simple et complet	Simple et assez complet	Simple et complet	Assez simple et assez complet	Simple et complet
	Ajouter de nouveaux comportements	Assez simple	Très simple	Simple	Moyennement simple	Simple	Assez simple	Moyennement simple
	Communications	Oui	Oui	Oui	Non	Oui (prochainement disponible)	Non	Oui
Aller plus loin	Liberté de personnalisation et de création	Bonne	Très bonne	Très bonne	Moyennement bonne	Assez bonne	Assez bonne	Moyennement bonne
	Création d'extensions (plug-ins)	Assez simple mais limité	Simple	Assez simple	Assez simple	Assez simple	Assez simple	Assez simple
	Clarté du code source	Code source non disponible	Code source non disponible	Clair et bien commenté	Assez clair et bien commenté	Clair et bien commenté	Assez clair et bien commenté	Clair et bien commenté

TABLE 1 – Tableau comparatif des simulateurs présentés (plus la case est grisée, plus la catégorie est positive).

souhaite pour chaque véhicule. Pour le moment, seule la loi de poursuite *Intelligent Driver Model* (*IDM*) [29] a été mise en place mais il est assez simple d'ajouter les autres lois de poursuite existant dans la littérature. Chaque véhicule peut changer de voie en choisissant la position d'arrivée et le temps nécessaire pour atteindre celle-ci. Bien que la modélisation de la route soit encore limitée, on peut déjà effectuer différentes analyses pour observer les différents comportements dans des cas d'autoroutes. Cela nous permet donc d'étudier l'apport de la communication et plus précisément de la négociation dans un environnement mixte constitué de véhicules intelligents et non intelligents.

Références

[1] Site officiel de via (senozon). http://www.senozon.com/products/via.

[2] *Microscopic Modeling of Traffic Flow : Investigation of Collision Free Vehicle Dynamics*. PhD thesis, Universität zu Köln, 1998.

[3] Bilan définitif de l'accidentalité routière. http://www.securite-routiere.gouv.fr, 2015.

[4] Chiffres clés du transport (pages 13 et 14). http://www.statistiques.developpement-durable.gouv.fr, 2017.

[5] Rob Allan. Survey of Agent Based Modelling and Simulation Tools. *Engineering*, 501(October) :57–72, 2009.

[6] Fan Bai and Hariharan Krishnan. Reliability Analysis of DSRC Wireless Communication for Vehicle Safety Applications. *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 355–362, 2006.

[7] M Balmer, K Meister, M Rieser, K Nagel, and K W Axhausen. Agent-based simulation of travel demand : Structure and computational performance of MATSim-T. *2nd TRB Conference on Innovations in Travel Modeling Portland Juni 2008*, (June) :1–33, 2008.

[8] Andrei Borshchev. Getting Started with AnyLogic - a Multi-Method Simulation Modeling Tool , 2010.

[9] Gordon D. B. Cameron and Gordon I. D. Duncan. PARAMICS Parallel microscopic simulation of road traffic. *The Journal of Supercomputing*, 10 :25–53, 1996.

[10] Jordi Casas, Jaime L. Ferrer, David Garcia, Josep Perarnau, and Alex Torday. Traffic simulation with Aimsun. In *International Series in Operations Research and Management Science*, volume 145, pages 173–232. 2010.

[11] Martin Fellendorf and Peter Vortisch. Microscopic traffic flow simulator VISSIM. In *International Series in Operations Research and Management Science*, volume 145, pages 63–93. 2010.

[12] Jacques Ferber. *Multi-Agent Systems : An Introduction to Distributed Artificial Intelligence*, volume 222. 1999.

[13] Foundation For and Intelligent Physical. FIPA Communicative Act Library Specification. *Architecture*, 2000(SC00037J), 2002.

[14] P. G. Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B*, 15(2) :105–111, 1981.

[15] Maxime Gueriau, Romain Billot, Salima Hassas, Frederic Armetta, and Nour Eddin El Faouzi. An extension of MovSim for multi-agent cooperative vehicles modeling. In *2014 International Conference on Connected Vehicles and Expo, ICCVE 2014 - Proceedings*, pages 859–860, 2015.

[16] Tobias Haubrich, Sven Seele, Rainer Herpers, Martin E. Muller, and Peter Becker. A Semantic Road Network Model for traffic simulations in virtual environments : Generation and integration. In *2014 IEEE 7th Workshop on Software Engineering and Architectures for Realtime Interactive Systems, SEARIS 2014*, pages 43–50, 2014.

[17] Tanuja K, Sushma T M, Bharathi M, and Arun K H. A Survey of VANET Technologies. *International Journal of Computer Applications*, 121(9) :661–671, 2015.

[18] Arne Kesting, Martin Treiber, and Dirk Helbing. General Lane-Changing Model MOBIL for Car-Following Models. *Transportation Research Record : Journal of Transportation Research Board*, 1999(1) :86–94, 2007.

[19] Daniel Krajzewicz and G Hertkorn. SUMO (Simulation of Urban MObility) An open-source traffic simulation. ... *Symposium on Simulation* ..., pages 63–68, 2002.

[20] Rafael Math, Angela Mahr, Mohammad Mehdi Moniri, and Christian Müller. OpenDS : A new open-source driving simulator for research. *Adjunct Proceedings of the 4th International Conference on Automotive User Interfaces and Interactive Vehicular Appilcations*, pages 7–8, 2012.

[21] Mathworks. Introduction to Simulink. *Matlab Simulink User's Guide R2014b*, pages 1–69, 2014.

[22] Mustapha Saidallah, Abdeslam El Fergougui, and Abdelbaki Elbelrhiti Elalaoui. A Comparative Study of Urban Road Traffic Simulators. *computer Science Department, ReSI team, Moulay Ismail University*, 2016.

[23] Reid G. Smith. Correction to "The Contract Net Protocol : High-Level Communication and Control in a Distributed Problem Solver", 1981.

[24] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1) :3–15, January 2011.

[25] Patrick Taillandier. Traffic simulation with the GAMA platform. *International Workshop on Agents in Traffic and Transportation*, 2014.

[26] Patrick Taillandier, Duc An Vo, Edouard Amouroux, and Alexis Drogoul. GAMA : A simulation platform that integrates geographical information data, agent-based modeling and multi-scale control. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 7057 LNAI, pages 242–258, 2012.

[27] Inc. The MathWorks. MATLAB, 2015.

[28] Martin Treiber and Arne Kesting. Elementary Car-Following Models. In *Traffic Flow Dynamics*, pages 157–180. 2013.

[29] Martin Treiber and Arne Kesting. *Traffic Flow Dynamics*. 2013.

[30] Sumin Zhang, Weiwen Deng, Qingrong Zhao, Hao Sun, and Bakhtiar Litkouhi. Dynamic trajectory planning for vehicle autonomous driving. In *Proceedings - 2013 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2013*, pages 4161–4166, 2013.

Jeux de coalitions hédoniques à concepts de solution multiples

Thibaut Vallée[a] thibaut.vallee@unicaen.fr
Grégory Bonnet[a] gregory.bonnet@unicaen.fr

[a]Normandie Université, UNICAEN, GREYC, CNRS UMR 6072

Résumé

Dans un système d'agents autonomes, ces derniers peuvent être amenés à se demander avec qui coopérer sachant que chaque agent préfère interagir avec certains agents plutôt que d'autres. Ce problème est étudié par la formation de coalitions hédoniques qui caractérise les solutions stables au regard des préférences des agents par des concepts de solution. Toutefois, ces derniers ne modélisent qu'un a priori sur le comportement des agents. Par exemple, la stabilité au sens de Nash modélise des agents qui rejoignent les coalitions qu'ils préfèrent sans se soucier des autres. Il pourrait alors être intéressant de concevoir des agents hétérogènes dans leur définition des solutions stables. Pour ce faire, nous proposons un modèle où les agents expriment non seulement leurs préférences sur les autres, mais aussi des préférences sur les concepts de solution.

Mots-clés : *Coalitions, Modèles de comportement agent, Théorie des jeux*

Abstract

In multiagent systems, agents may be led to ask themselves with whom to cooperate, knowing that each of them expresses its own preferences. This problem is studied in hedonic games with solution concepts characterizing the stability of outcomes with respect to the agents' preferences. However, this framework considers an a priori about agents' behaviours. For instance, Nash stability modelizes agents which join coalitions that they prefers without any considerations about the others. Thus, it might also interesting to consider agents which are heterogeneous in their definition of stable solutions. For this purpose, we propose a new hedonic game where agents express preferences on both the coalitions and the solutions concepts.

Keywords: *Behavior models, Coalitions, Game theory*

1 Introduction

Dans un système d'agents autonomes, il est fréquent que ces derniers soient amenés à devoir décider avec qui coopérer et former ainsi des coalitions. Les jeux de coalitions hédoniques modélisent ce problème en considérant des agents hétérogènes, au sens où ses derniers expriment des préférences sur les groupes d'agents qu'ils peuvent rejoindre [6]. Une solution d'un tel jeu est une partition stable : aucun agent n'a d'intérêt à quitter la coalition qui lui a été affectée au regard d'un critère appelé concept de solution. Ce concept de solution est alors un a priori sur le comportement des agents. Par exemple, la stabilité au sens de Nash suppose que chaque agent va chercher à rejoindre une coalition déjà formée s'il la préfère à celle à laquelle il a été affecté. Or, dans un système où les agents sont hétérogènes, ceux-ci n'appliquent pas nécessairement les mêmes règles de stabilité. Il est en effet envisageable que, dans un même jeu, un agent applique la stabilité au sens de Nash et que, dans le même temps, un autre agent plus respectueux des autres ne quitte sa coalition que si cela ne nuit à personne.

Pour répondre à un tel cas de figure, nous proposons dans cet article deux nouveaux modèles de jeux de coalitions hédoniques. Le premier, appelé *jeu de coalitions hédonique à concepts de solution multiples*, permet à chaque agent de considérer un concept de solution qui lui est propre. Dans le second modèle, appelé *jeu de coalitions hédonique à double profil*, les agents expriment des préférences sur un ensemble de concepts de solution. Nous montrons les principales propriétés de ces modèles et proposons une nouvelle notion de stabilité qui minimisent les *concessions* des agents sur les concepts de solution pour trouver une solution non vide.

Après avoir introduit en Section 2 les jeux de coalitions hédoniques canoniques, nous présentons en Section 3 les jeux de coalitions hédoniques à concepts de solution multiples. Nous

étendons ensuite dans la section 4 ce modèle aux jeux de coalitions hédoniques à double profil. Nous consacrons les Sections 3.2 et 4.2 à des analyses formelles de ces modèles, et les Sections 3.3 et 4.3 à des analyses empiriques.

2 Jeux hédoniques canoniques

Lorsqu'un ensemble d'agents doit coopérer temporairement dans le but de réaliser des objectifs qui leur sont propres, l'une des problématiques est de décider avec qui coopérer. Il s'agit d'un problème de formation de coalitions (ou jeu de coalitions) consistant à trouver un partitionnement des agents qui les satisfait tous. De nombreux modèles de jeux de coalitions ont été étudiés dans la littérature [1, 3, 6, 7]. Certains s'intéressent à des modèles quantitatifs où les agents cherchent à maximiser leurs utilités, d'autres – les jeux de coalitions hédoniques – s'intéressent à des modèles qualitatifs [6, 7] où chaque agent évalue sa satisfaction en fonction de la coalition à laquelle il appartient.

Définition 1 (HG)**.** *Un* jeu de coalitions hédoniques *est un tuple* $HG = \langle N, \succeq \rangle$ *où* $N = \{a_1, \ldots, a_n\}$ *est l'ensemble des agents, et* $\succeq = \{\succeq_1, \ldots, \succeq_n\}$ *est l'ensemble des profils de préférence des agents, c'est-à-dire des préodres totaux avec indifférences sur l'ensemble* $G_i = \{C \subseteq N : a_i \in C\}$ *des coalitions auxquelles l'agent* a_i *peut apartenir.*

Trouver la solution d'un jeu de coalitions consiste à trouver une partition stable, c'est-à-dire qu'aucun agent ne peut ou ne veut dévier de sa coalition actuelle. Les concepts de solution caractérisent des propriétés que doit satisfaire toute partition stable. Par exemple, la Pareto-Optimalité est le concept de solution regroupant toutes les partitions telles qu'aucun agent ne puisse quitter sa coalition actuelle pour une autre une coalition qu'il préfère sans dégrader la solution pour au moins un autre agent. De nombreux concepts de solution ont été proposés et étudiés dans la littérature [1, 10, 14]. Remarquons qu'un jeu de coalitions canonique est vu comme un problème statique : les préférences des agents n'évoluent pas en cours de résolution du jeu bien que certains travaux s'intéressent à ces aspects dynamiques [8].

Dans cet article, nous considérons un jeu hédonique statique et les différents concepts de solution canoniques présentés dans la Table 1. Nous désignons par Π une partition des agents, par $C_i(\Pi)$ la coalition de l'agent a_i dans cette partition. Une partition Π est stable au sens d'un concept de solution SC si et seulement si elle satisfait la propriété qui caractérise SC. De manière intéressante, les concepts de solution peuvent être définis en composant différentes caractérisations de déviations autorisées [14]. Par exemple, la *stabilité contractuelle de Nash* est le concept de solution autorisant un agent à dévier de sa coalition actuelle pour en rejoindre une qu'il préfère (stabilité de Nash), sous réserve que cette déviation ne dégrade pas la solution pour les autres membres de la coalition qu'il quitte (stabilité contractuelle). Ainsi, les déviations autorisées représentent des comportements d'agents et leurs caractérisations sont porteuses d'un a priori sur eux. Par exemple, la *stabilité de Nash* modélise des agents individualistes qui ne considèrent que leurs propres préférences, contrairement à la *stabilité individuelle contractuelle* qui est une prise en compte de la collectivité puisque l'agent ne déviera que si les autres l'acceptent.

3 Concepts de solution locaux

Si les concepts de solution sont porteurs d'a priori sur le comportement individuel des agents, ce sont surtout des *concepts de solution globaux* porteurs d'un a priori qui s'applique à tous les agents comme l'indiquent les termes soulignés dans la Table 1. Afin de considérer des agents hétérogènes dans leur définition de la stabilité, nous proposons alors un jeu hédonique qui ne considère non plus le concept de solution comme une donnée externe au jeu, mais comme un paramètre du jeu exprimé par chaque agent sous forme de *concepts de solution locaux*.

3.1 Du global au local

Si les concepts de solution canoniques caractérisent des propriétés qui doivent être vraies *pour tous les agents*, nous exprimons ces propriétés du point de vue de chaque agent : un concept de solution local caractérise une propriété vraie *pour un agent fixé*.

Définition 2 (LSC_i)**.** *Soit* $HG = \langle N, \succeq \rangle$ *un jeu de coalitions hédonique,* SC *un concept de solution global et* $a_i \in N$ *un agent. Le* concept de solution local LSC_i *caractérise l'ensemble des partitions* Π *de* N *qui vérifient les propriétés de* SC *pour l'agent* a_i*.*

Pour chaque concept canonique x, nous définissons Lx_i comme le concept de solution local

Concept de solution canonique	Acronyme	Propriété
Rationnalité Individuelle	RI	$\underline{\forall a_i \in N}, C_i(\Pi) \succeq_i \{a_i\}$
Stabilité de Nash	NS	$\underline{\forall a_i \in N}, \nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi)$
Stabilité Individuelle	IS	$\underline{\forall a_i \in N}, \nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi)$ $\wedge \forall a_j \in C, C \cup \{a_i\} \succeq_j C$
Stabilité Individuelle Contractuelle	ICS	$\underline{\forall a_i \in N}, \nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi)$ $\wedge \forall a_j \in C, C \cup \{a_i\} \succeq_j C$ $\wedge \forall a_k \in C_i(\Pi), a_k \neq a_i, C_i(\Pi) \setminus \{a_i\} \succeq_k C_i(\Pi)$
Stabilité Contractuelle de Nash	CNS	$\underline{\forall a_i \in N}, \nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi)$ $\wedge \forall a_k \in C_i(\Pi), a_k \neq a_i, C_i(\Pi) \setminus \{a_i\} \succeq_k C_i(\Pi)$
Stabilité du Cœur	CS	$\underline{\forall a_i \in N}, \nexists C \in G_i : C \succ_i C_i(\Pi)$ $\wedge \forall a_j \in C, C \succeq_j C_j(\Pi)$
Optimalité	O	$\underline{\forall a_i \in N}, \nexists C \in G_i : C \succ_i C_i(\Pi)$
Pareto-Optimalité	PO	$\nexists \Pi_2 : \underline{\forall a_i \in N}, C_i(\Pi_2) \succeq_i C_i(\Pi)$ $\wedge \underline{\exists a_j \in N}, C_j(\Pi_2) \succ_j C_j(\Pi)$

TAB. 1 – Principaux concepts de solution canoniques

pour un agent $a_i \in N$ (résumés en Table 2). Nous pouvons ainsi modéliser des jeux où une partition Π est stable du point de vue d'un agent a_i s'il n'existe pas de coalitions qu'il désirerait rejoindre ($\Pi \in LNS_i$) et est stable du point de vue d'un autre agent a_j si toute déviation de sa part dégrade la solution pour au moins un autre agent ($\Pi \in LPO_j$).

Définition 3 (MHG)**.** *Un* jeu de coalitions hédonique à concepts de solution multiples *est un triplet* $MHG = \langle N, \succeq, LSC \rangle$ *où* $N = \{a_1, \ldots, a_n\}$ *est l'ensemble des agents,* $\succeq = \{\succeq_1, \ldots, \succeq_n\}$ *l'ensemble des profils de préférence des agents vis-à-vis des coalitions, et* $LSC = \{LSC_1, \ldots, LSC_i\}$ *l'ensemble des concepts de solution locaux des agents.*

Trouver une solution d'un MHG revient à trouver une partition qui satisfait le concept de solution local de chaque agent.

Définition 4 (Stabilité d'un MHG)**.** *Soit un MHG. Une partition* Π *est* stable *si* $\forall a_i \in N, \Pi \in LSC_i$*. L'ensemble des partitions stables d'un MHG est noté* MS*.*

Exemple 1. *Considérons le jeu* $HG = \langle N, \succeq \rangle$ *avec* $N = \{a_1, a_2, a_3\}$ *et les profils de préférence suivants :*

$\succeq_1 = \{a_1, a_2\} \succ \{a_1\} \succ \{a_1, a_2, a_3\} \succ \{a_1, a_3\}$

$\succeq_2 = \{a_2, a_3\} \succ \{a_1, a_2, a_3\} \succ \{a_1, a_2\} \succ \{a_2\}$

$\succeq_3 = \{a_2, a_3\} \succ \{a_1, a_2, a_3\} \succ \{a_3\} \succ \{a_1, a_3\}$

Avec trois agents, il y a cinq partitions :

$$\Pi_1 = \{\{a_1, a_2, a_3\}\}$$
$$\Pi_2 = \{\{a_1\}, \{a_2, a_3\}\}$$
$$\Pi_3 = \{\{a_1, a_3\}, \{a_2\}\}$$
$$\Pi_4 = \{\{a_1, a_2\}, \{a_3\}\}$$
$$\Pi_5 = \{\{a_1\}, \{a_2\}, \{a_3\}\}$$

La Table 3 montre l'appartenance de ces partitions aux concepts de solution locaux du point de vue de l'agent a_1*. Si par exemple* $LSC = \{LNS_1, LR_2, LPO_3\}$ *alors* $MS = \{\Pi_2, \Pi_4\}$*. Notons que* Π_1 *et* Π_3 *ne satisfont aucun des concepts de solution locaux de* a_1*. Donc, quoi que* a_1 *exprime, ces deux partitions ne peuvent pas être stables.*

3.2 Propriétés des MHG

Les concepts de solution locaux ont les mêmes propriétés que leurs équivalents globaux et, donc, ne les dénaturent pas. En premier lieu, les concepts de solution globaux ont des propriétés d'inclusions [3]. Par exemple, $NS \subseteq IS \subseteq ICS$. Trivialement, au vu de leur définition, les concepts de solution locaux satisfont les mêmes propriétés, résumées dans la Figure 1. Par exemple, une partition LNS ne tient pas compte des préférences des autres agents et est

Concept local	**Propriété**
LRI_i	$C_i(\Pi) \succeq_i \{a_i\}$
LNS_i	$\nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi)$
LIS_i	$\nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi) \wedge \forall a_j \in C, C \cup \{a_i\} \succeq_j C$
$LICS_i$	$\nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi) \wedge \forall a_j \in C, C \cup \{a_i\} \succeq_j C$ $\wedge \forall a_k \in C_i(\Pi), a_k \neq a_i, C_i(\Pi) \setminus \{a_i\} \succeq_k C_i(\Pi)$
$LCNS_i$	$\nexists C \in \Pi \cup \{\emptyset\} : C \cup \{a_i\} \succ_i C_i(\Pi) \wedge \forall a_k \in C_i(\Pi), a_k \neq a_i, C_i(\Pi) \setminus \{a_i\} \succeq_k C_i(\Pi)$
LCS_i	$\nexists C \in G_i : C \succ_i C_i(\Pi) \wedge \forall a_j \in C, C \succeq_j C_j(\Pi)$
LO_i	$\nexists C \in G_i : C \succ_i C_i(\Pi)$
LPO_i	$\nexists \Pi_2 : C_i(\Pi_2) \succ_i C_i(\Pi) \wedge \forall a_j \in N, C_j(\Pi_2) \succeq_j C_j(\Pi)$

TAB. 2 – Concepts de solution locaux pour un agent $a_i \in N$

LSC	Π_1	Π_2	Π_3	Π_4	Π_5
LR_1		✓		✓	✓
LNS_1		✓		✓	
LIS_1		✓		✓	
$LICS_1$		✓		✓	
$LNCS_1$		✓		✓	
LCS_1		✓		✓	
LO_1				✓	
LPO_1		✓		✓	

TAB. 3 – Partitions stables selon a_1

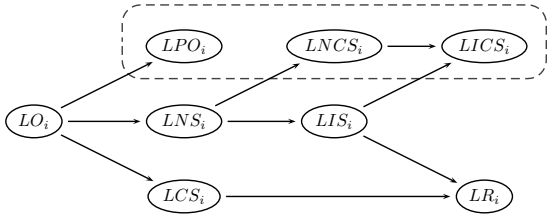


FIG. 1 – Relations d'inclusions entre Lx_i

donc nécessairement incluse dans LIS, $LCNS$, $LICS$, LR. L'hyperarête en pointillés indique les concepts de solution *irrationnels*, c'est-à-dire que les concepts dont la satisfaction ne garantit pas à l'agent d'être dans une coalition a minima équivalente en termes de préférences à sa coalition singleton. En second lieu, si tous les agents expriment le même concept de solution local, nous retrouvons les partitions stables du concept de solution global associé.

Propriété 1. *Soit un MHG. Si* $\forall a_i, a_j \in N, LSC_i = LSC_j$ *et* SC *le concept global dont dérive* LSC_i *alors* $\bigcap_{a_i \in N} LSC_i = SC$.

Nous présentons ici uniquement la preuve pour PO. Les preuves pour les autres concepts de solution globaux sont similaires.

Démonstration. Soit un MHG tel que l'ensemble des agents expriment l'optimalité au sens de Pareto locale. L'ensemble LPO des partitions Localement Pareto-Optimale pour tous les agents est tel que :

$$LPO = \bigcap_{a_i \in N} LPO_i$$

Montrons dans un premier temps que, pour toute partition $\Pi \in LPO$, nous avons $\Pi \in PO$. Supposons que $\Pi \notin PO$. Par définition de PO, nous avons nécessairement $\exists \Pi_2, \exists a_i \in N$ tels que $C_i(\Pi_2) \succ_i C_i(\Pi)$ et que $\forall a_j \in N \setminus \{a_i\}, C_j(\Pi_2) \succeq_j C_j(\Pi)$. Ainsi, selon la définition de LPO, nous avons $\Pi \notin LPO_i$, ce qui est en contradiction avec notre hypothèse.

Montrons maintenant que, pour toute partition $\Pi \in PO$, nous avons $\Pi \in LPO$. Supposons qu'il existe un agent a_i tel que $\Pi \notin LPO_i$. Par définition de LPO_i, nous avons $\exists \Pi_2$: $C_i(\Pi_2) \succ_i C_i(\Pi)$ et $\forall a_j \in N \setminus \{a_i\}, C_j(\Pi_2) \succeq_j C_j(\Pi)$. Ceci est en contradiction avec $\Pi \in PO$.

Par conséquent, $\Pi \in LPO$.

Donc, nous avons $PO = \bigcap_{a_i \in N} LPO_i$. □

Remarquons que les MHG peuvent avoir des solutions stables qui ne correspondent à aucun concept de solution canonique.

Propriété 2. *Il existe des MHG tels que* $\exists \Pi \in MS$ *où, pour tout concept de solution canonique* SC *indiqué en Table 1,* $\Pi \notin SC$.

L'Exemple 2 illustre ce phénomène.

Exemple 2. *Considérons le jeu* $MHG = \langle N, \succeq, LSC \rangle$ *avec* $N = \{a_1, a_2, a_3\}$, $LSC = \{LIS_1, LICS_2, LRI_3\}$ *et les profils de préférence suivants :*

$\succeq_1 = \{a_1, a_2, a_3\} \succ \{a_1, a_2\} \succ \{a_1\} \succ \{a_1, a_3\}$

$\succeq_2 = \{a_2, a_3\} \succ \{a_2\} \succ \{a_1, a_2, a_3\} \succ \{a_1, a_2\}$

$\succeq_3 = \{a_1, a_2, a_3\} \succ \{a_1, a_3\} \succ \{a_2, a_3\} \succ \{a_3\}$

Ce jeu a trois partitions stables : $\{\{a_1, a_2, a_3\}\}$, $\{\{a_1, a_3\}, \{a_2\}\}$ *et* $\{\{a_1, a_2\}, \{a_3\}\}$. *Si les deux premières satisfont des concepts de solution canoniques, la partition* $\{\{a_1, a_2\}, \{a_3\}\}$ *n'en satisfait aucun. Dans cette partition, la grande coalition est préférée par tous les agents à leurs coalitions respectives mais, en exprimant la rationalité comme concept de solution local, l'agent* a_3 *désire jsute être satisfait et ne cherche pas à dévier vers sa coalition optimale. De plus, les agents* a_1 *et* a_2 *pourraient former la grande coalition s'ils considéraient des déviations collectives, ce qui n'est pas le cas ici. Ainsi, chaque agent attend que se soient les autres qui dévient.*

Intéressons-nous maintenant à l'existence des partitions stables dans un MHG. Si tous les concepts de solution globaux ne garantissent pas l'existence d'une partition stable, tous les concepts de solution locaux garantissent l'existence d'une partition stable selon a_i.

Propriété 3. *Soit un MHG. Pour tout agent* $a_i \in N$ *et tout concept de solution local* LSC_i *indiqué dans la* TAB. 2*, il existe au moins une partition* Π *de* N *telle que* $\Pi \in LSC_i$.

En effet, indépendamment de l'agent et du concept de solution local que cet agent exprime, l'ensemble des partitions contenant sa coalition optimale sont nécessairement localement stable.

Démonstration. Soit un MHG, un agent $a_i \in N$, et $C_i^* \in G_i$ l'une des coalitions telles qu'aucune autre coalition C ne soit strictement préférée à C_i^* par a_i : $\forall C \in G_i, C_i^* \succeq_i C$. C_i^* est la coalition optimale pour l'agent a_i. Par définition, de l'optimalité locale, toute partition Π contenant C_i^* est nécessairement localement optimale. Ainsi, $LO_i \neq \emptyset$. Par inclusion des concepts de solution, toute partition $\Pi \in LO_i$ appartient également aux autres concepts de solution locaux. Ainsi, quel soit LSC_i, cette partition Π est localement stable. □

Cependant, cette propriété ne garantit pas l'existence d'une partition satisfaisant les concepts de solution locaux de chaque agent. Ainsi, il existe des MHG tels que $MS = \emptyset$.

Exemple 3. *Soit un MHG tel que* $N = \{a_1, a_2\}$, $\succeq_1 = \{a_1\} \succ_1 \{a_1, a_2\}$, $\succeq_2 = \{a_1, a_2\} \succ_2 \{a_2\}$ *et* $LSC = \{LO_1, LO_2\}$ *. Ce jeu possède deux partitions :* $\Pi_1 = \{\{a_1\}, \{a_2\}\}$ *et* $\Pi_2 = \{\{a_1, a_2\}\}$. *Par définition, il n'existe pas de partition dans* O *alors que* $LO_1 = \{\Pi_1\}$ *et* $LO_2 = \{\Pi_2\}$.

La complexité des jeux de coalitions hédoniques a été largement étudiée dans la littérature [1, 2, 10]. Dans la majorité des cas, trouver une partition stable pour un concept de solution donné est un problème NP-complet [10] sauf pour quelques exceptions comme la recherche de partitions individuellement rationnelles ou de partitions stables au sens du cœur dans les jeux de coalitions à $\mathcal{W}\beta$-préférences [7].

Propriété 4. *Trouver une partition stable dans un MHG est un problème équivalent à celui de trouver une partition satisfaisant le concept de solution* SC *dans un HG canonique où* SC *est le concept de solution global appartenant au plus haut niveau de la hiérarchie polynomiale parmi tous les concepts de solution globaux associés aux concepts de solution locaux de MHG.*

Intuition de la preuve. Considérons $MHG = \langle N, LSC, \succeq \rangle$ et $HG = \langle N, \succeq \rangle$ le jeu hédonique tel que les agents et leurs profils de préférence soient identiques dans les deux jeux. Soit SC^* le concept de solution appartenant au plus haut niveau de la hiérarchie polynomiale parmi parmi tous les concepts de solution globaux associés aux concepts de solution locaux LSC et LSC_i^* son équivalent local pour l'agent a_i. Pour une partition Π donnée, vérifier que Π est stable dans MHG revient à vérifier que $\forall a_i \in N, \Pi \in LSC_i$. De même, vérifier $\Pi \in SC^*$ dans HG

revient à vérifier que $\forall a_i \in N, \Pi \in LSC_i^*$. Comme il existe au moins un agent a_i tel que $LSC_i = LSC_i^*$, les deux problèmes sont équivalents. Donc, trouver une partition stable dans un MHG peut être réduit à trouver une solution pour SC^* et est donc un problème du même niveau dans la hiérarchie polynomiale. □

3.3 Analyse empirique

De nombreux concepts canoniques n'ont en pratique que peu de solutions. Qu'en est-il des MHG ? Pour étudier cela empiriqement, considérons un ensemble de 3 à 7 agents choisissant de manière uniforme leur relation de préférence et un concept de solution local. Nos résultats reposent sur 1000 jeux aléatoires [1]

La Table 4 présente (par ordre quasi-croissant) le nombre moyen de partitions stables (colonne $|MS|$) ainsi que le nombre de solutions satisfaisant les concepts de solution globaux canoniques. La colone $\mathcal{B}_n$ présente à titre indicatif le nombre de partitions pour n agents. Le point essentiel est que ce nombre de partitions stables est plus grand que presque tous ceux des concepts de solution rationnels (NS, IS, CS et O) et plus faible que ceux des concepts de solution irrationnels (CIS, CNS, PO). Seule la rationnalité individuelle fait exception à cette règle. En effet, ce concept de solution est très faiblement contraint et permet l'existence d'un nombre important de partitions stables. Cependant, il ne garantit aux agents que d'être dans une coalition qui n'est pas moins préférée que leurs coalitions singletons respectives alors que les autres concepts assurent qu'il n'existe pas de solution préférable pour tous les agents.

Notre modèle exprime donc un compromis entre les concepts de solution rationnels et les concepts irrationnels : seuls les agents acceptant des concepts de solution locaux irrationnels peuvent être affectés à une coalition irrationnelle de leur point de vue.

La Figure 2 présente la proportion de partitions stables qui satisfont aussi un concept de solution canonique. Une grande proportion des partitions stables sont également Pareto-optimales ou individuellement contractuellement stables. Par exemple, 86% des partitions stables à 7 agents sont également Pareto-optimale. Notre modèle est donc essentiellement une restriction de ces deux concepts tout en permettant parfois des solutions ne correspondant à aucun concept canonique (voir Propriété 2 et groupe de données "Aucun" de la Figure 2).

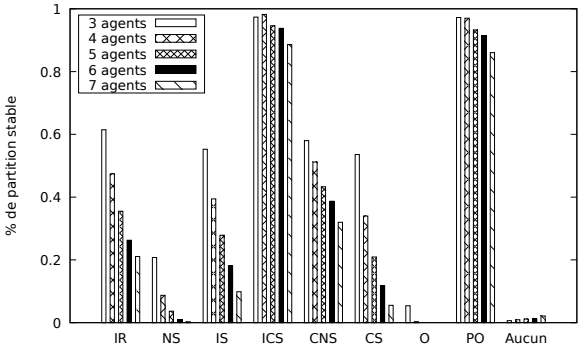


FIG. 2 – Taux de satisfaction des concepts de solution canoniques parmi les partitions stables

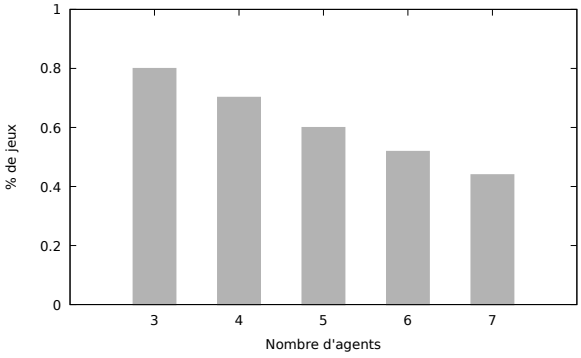


FIG. 3 – Taux de jeux ayant au moins une partition stable en fonction du nombre d'agents

Enfin, la Figure 3 indique la proportion de MHG ayant au moins une partition stable. Avec 3 agents, près de 80% des jeux ont au moins une solution. Cette proportion décroit fortement et tombe à un peu plus de 40% pour 7 agents. L'absence de solution stable peut s'expliquer par le fait que certains agents expriment des propriétés *trop restrictives*. Si ces agents pouvaient accepter de réduire leurs exigences en considérant un autre concept de solution local, le jeu aurait peut-être une solution. C'est pourquoi nous étendons notre modèle en permettant aux agents d'exprimer des préférences entre les concepts de solution locaux qu'ils acceptent de considérer.

1. Nous sommes conscients des limites de cette étude, car, en considérant des profils de préférence stricts tirés uniformément, n agents et m concepts de solution locaux, il y a $(m \times 2^{n-1}!)^n$ jeux différents, soit 7077888 jeux pour 3 agents. Notons cependant qu'un grand nombre de ces jeux sont symétriques.

	Concepts canoniques rationnels						Concepts canoniques irrationnels			
n	O	NS	CS	IS	RI	**MS**	CNS	PO	CIS	$\mathcal{B}_n$
3	$0,084$	$0,343$	$1,024$	$1,09$	$1,968$	$\mathbf{1,199}$	$1,629$	$2,949$	$3,003$	5
4	$0,007$	$0,219$	$1,12$	$1,399$	$3,269$	$\mathbf{1,547}$	$3,444$	$6,869$	$7,591$	15
5	0	$0,158$	$1,177$	$1,892$	$6,44$	$\mathbf{2,193}$	$8,5$	$18,35$	$22,49$	52
6	0	$0,095$	$1,241$	$2,836$	$13,745$	$\mathbf{3,14}$	$24,355$	$54,126$	$74,765$	203
7	0	$0,054$	$1,3$	$4,86$	$31,882$	$\mathbf{6,053}$	$77,5475$	$171,896$	$275,073$	877

TAB. 4 – Nombre moyen de partitions stables selon les concepts de solution et le nombre d'agents

4 Préférences sur les concepts

4.1 Modélisation des nouvelles préférences

Nous étendons le modèle précédent en y introduisant des profils de préférence sur les concepts de solution locaux.

Définition 5 (HG2P)**.** *Un* jeu de coalitions hédoniques à double profil *est un tuple* $HG2P = \langle N, LSC, \succeq^C, \succeq^{LSC} \rangle$ *où* $N = \{a_1, \ldots, a_n\}$ *est l'ensemble des agents,* LSC *un ensemble de concept de solution locaux,* $\succeq^C = \{\succeq^C_1, \ldots, \succeq^C_n\}$ *l'ensemble des profils de préférence des agents sur les coalitions, et* $\succeq^{LSC} = \{\succeq^{LSC}_1, \ldots, \succeq^{LSC}_n\}$ *l'ensemble des profils de préférence des agents sur des sous-ensembles non vides de* LSC.

Commee chaque agent exprime ses préférences sur un sous-ensemble de LSC, les relations $\succeq^{LSC}_i$ sont incomplètes. Par exemple, un agent a_1 qui désire garantir à minima une rationalité individuelle peut ne pas exprimer de préférence sur les concepts de solution irrationnels (CNS, PO, CIS). De plus, les concepts de solution exprimés peuvent être distincts entre les agents. Trivialement, un MHG est un cas particulier de HG2P : le sous-ensemble de LSC considéré par chaque agent est un singleton. Il est en de même pour un HG canonique. Dans la suite, par abus de notation, nous désignons par $LSC' \in \succeq^{LSC}_i$ le fait que l'agent a_i considère le concept de solution local LSC' dans son profil de préférence. Nous notons aussi $r_i(LSC')$ le rang du concept de solution local LSC' dans $\succeq^{LSC}_i$. Si $LSC' \notin \succeq^{LSC}_i$ nous considérons que $r_i(LSC) = \infty$.

4.2 Stabilité et concessions

La stabilité d'une partition dépend du nombre de *concessions sur* $\succeq^{LSC}$ qu'elle implique.

Définition 6 (Vecteur de concessions)**.** *Soit un HG2P et* Π *une partition de* N. *Le vecteur de* concessions *de* Π *est le vecteur* $\vec{c}(\Pi)$ *où :*

$$c_i(\Pi) = \begin{cases} r(LSC_i^*) & \text{si } \exists\, LSC_i \in \succeq^{LSC}_i \\ & \text{tel que } \Pi \in LSC_i \\ \infty & \text{sinon} \end{cases}$$

avec $LSC_i^* = \underset{LCS_i \in \succeq^{LSC}_i : \Pi \in LSC_i}{\operatorname{argmin}} r(LSC_i)$.

Intuitivement, le vecteur de concessions d'une partition représente combien de fois chaque agent doit choisir un concept de solution local moins préféré afin que la partition Π soit localement stable. Remarquons que si $\exists a_i \in N$ tel que $c_i(\Pi) = \infty$ alors Π n'appartient à aucun des concepts de solution considérés par l'agent a_i et donc ne peut pas être stable. Dans la suite, nous nommons concession de l'agent a_i la i-ème composante du vecteur de concessions.

À partir des vecteurs de concessions, nous proposons un nouveau concept de solution *global* : la stabilité au sens de la leximin-concession. Ce concept est inspiré de la règle de sélection leximax définie par [11, 5] et du dernier cœur dans les jeux à utilité transférable [13]. Il s'agit ici de chercher les partitions qui minimisent le nombre de concessions de l'agent ayant la concession maximale parmi tous les agents, puis la concession du second agent ayant la plus importante valeur, et ainsi de suite. Pour cela, nous ordonnons les vecteurs de concessions par ordre croissant de composantes et les comparons deux-à-deux par ordre lexicographique inverse. Toute partition dont le vecteur de concessions n'est pas dominé est considérée comme stable au sens de la *leximin-concession*, ou plus simplement appelée *partition leximin-stable*.

Définition 7 (Leximin-concession)**.** *Soit un HG2P et* Π *une partition de* N. Π *satisfait la* leximin-concession *si :*

(1) $\nexists\, a_i \in N$ *tel que* $c_i(\Pi) = \infty$,

(2) $\nexists\, \Pi'$ telle que, pour $\{x_1, \ldots, x_n\}$ (resp. $\{y_1, \ldots, y_n\}$) l'ensemble ordonné des composantes de $\vec{c}(\Pi)$ (resp. $\vec{c}(\Pi')$), il existe $k \in [1, n]$ vérifiant $x_k > y_k$ et $\forall i \in [1, k-1], x_i = y_i$

Exemple 4. *Reprenons l'Exemple 1 en y ajoutant les profils de préférence sur les concepts de solution locaux donnés en Table 5. La Table 6 indique les vecteurs de concessions de chaque partition. Comme montré dans l'Exemple 1, les partitions Pi_1 et Pi_4 ne peuvent pas être stable puisqu'elles n'appartiennent à aucun des concepts de solution locaux à l'agent a_1. Parmi les 3 partitions restantes, la partition $\Pi_2 = \{\{a_1\}, \{a_2, a_3\}\}$ est celle qui est stable au sens de la leximin-concession.*

Remarquons que, comme tout jeu de coalitions hédonique canonique peut être modélisé par un HG2P, un HG2P ne possède pas nécessairement de partition stable au sens de la leximin-concession. Cependant, comme certains concepts de solution canoniques dans un HG garantissent l'existence d'une solution, des conditions simples permettent d'assurer l'existence d'au moins une partition leximin-stable.

Propriété 5. *Il existe au moins une solution Leximin-concédée dans un HG2P si et seulement si il existe un concept de solution global SC tel que (1) $\forall HG$, $SC \neq \emptyset$, et (2) $\forall\, a_i \in N$, le concept de solution local LSC_i associé à SC est exprimé dans $\succ_i^{LSC}$.*

Rappelons que, parmi les concepts canoniques que nous considérons, la rationalité, la Pareto-optimalité et la stabilité individuelle contractuelle garantissent l'existence d'une partition stable [2, 3, 14].

Démonstration. Montrons tout d'abord que si au moins un agent n'exprime pas une préférence sur un concept de solution local LCS^* tel que le concept global associé SC^* garantisse l'existence d'une solution pour tout HG alors il n'existe pas nécessairement de partition leximin-stable. Fixons un jeu $HG2P = \langle \{a_1, a_2\}, LSC, \succeq^C, \succeq^{LSC} \rangle$ tel que :

$$\{a_1, a_2\} \succ_1^C \{a_1\} \text{ et } \{a_2\} \succ_2^C \{a_1, a_2\}$$

$$LPO_1 \succ_1^{LSC} LNS_1 \text{ et } LNS_2 \succ_2^{LSC} LR_2$$

Ici, l'agent a_1 définit ses préférences sur deux concepts de solution locaux dont les correspondants globaux ne garantissent pas l'existence d'une solution. Or, comme illustré sur la Table 7, il n'existe pas de partition Π telle que $\forall a_i \in N, c_i(\Pi) \neq \infty$. Ce jeu n'a donc pas de partition leximin-stable.

Montrons maintenant que si tous les agents expriment leurs préférences sur un concept de solution local LCS^* tel que le concept global associé SC^* garantisse l'existence d'une solution pour tout HG alors il existe nécessairement une partition leximin-stable. Soit un HG2P tel que $\forall a_i \in N$, $LSC^* \in \succ_i^{LSC}$. Soit le jeu hédonique classique $HG = \langle N, \succ \rangle$ tel que les agents aient les même préférences vis-à-vis des coalitions dans HG et dans HG2P. Par définition, $SC^* \neq \emptyset$ dans HG. Fixons une partition de HG2P telle que $\Pi^* \in SC^*$. Par la Propriété 1, nous avons nécessairement $\forall i \in N, \Pi^* \in LSC_i^*$. Ainsi, $\forall\ a_i$, $c_i(\Pi^*) \neq \infty$. Selon la Définition 7, soit Π^* est une partition Leximin-concédée, soit $\exists\ \Pi$ telle que pour $\{x_1, \ldots, x_n\}$ (resp. $\{y_1, \ldots, y_n\}$) l'ensemble ordonné des composantes de $\vec{c}(\Pi^*)$ (resp. $\vec{c}(\Pi)$), il existe $k \in [1, n]$ vérifiant $x_k > y_k$ et $\forall i \in [1, k-1], x_i = y_i$. Si une telle partition Π existe, soit elle est elle-même leximin-stable, soit par induction il existe une autre partition Π' leximin-stable. Dans les trois cas, il existe une partition leximin-stable dans HG2P. □

Si cette condition peut sembler restrictive, il est raisonable en pratique que chaque agent a_i exprime au moins $LR_i \in \succ_i^{LCS}$ en dernier rang de ses préférences. En effet, ceci représente le fait que si les agents n'arrivent pas à trouver une solution les satisfaisant tous alors ils ne coopéreront pas et formeront leurs coalitions singletons.

4.3 Analyse empirique des concession

Reprenons le protocole expérimental de la Section 3.3, la Figure 4 présente la proportion des vecteurs de concessions pour 5 agents sur 10000 jeux aléatoires. Nous ne considérons ici que des vecteurs anonymes : le vecteur de concessions $[3, 2, 1, 2, 1]$ est équivalent au vecteur $[1, 1, 2, 2, 3]$ par exemple. Avec environ 60% de jeux sans concession, nous retrouvons la même proportion de partitions stables que dans les MHG – comme illustré sur la Figure 3 – et 30% des jeux ne nécessite qu'une seul concession de la part d'un unique agent. De manière globale, un peu plus de 98% des jeux nécessitent aux plus que 2 agents concèdent une fois leurs préférences. Les autres vecteurs de concessions sont quant à eux des cas anecdotiques. Par exemple, seul 1 jeu sur 10000 néces-

N	$\{a_1, a_2, a_3\}$
LSC	$\{LR, LNS, LIS, LICS, LNCS, LCS, LPO, LO\}$
$\succeq_1^C$	$\{a_1, a_2\} \succ \{a_1\} \succ \{a_1, a_2, a_3\} \succ \{a_1, a_3\}$
$\succeq_2^C$	$\{a_2, a_3\} \succ \{a_1, a_2, a_3\} \succ \{a_1, a_2\} \succ \{a_2\}$
$\succeq_3^C$	$\{a_2, a_3\} \succ \{a_1, a_2, a_3\} \succ \{a_3\} \succ \{a_1, a_3\}$
$\succeq_1^{LSC}$	$LO \succ LNS \succ LCS \succ LIS \succ LNCS \succ LR \succ LICS \succ LPO$
$\succeq_2^{LSC}$	$LO \succ LNS \succ LIS \succ LCS \succ LR \succ LNCS \succ LICS \succ LPO$
$\succeq_3^{LSC}$	$LO \succ LPO \succ LNS \succ LIS \succ LNCS \succ LICS \succ LCS \succ LR$

TAB. 5 – Préférences des agents sur les concepts de solution locaux

Π	$\vec{c}(\Pi)$
$\Pi_1 = \{\{a_1, a_2, a_3\}\}$	$[\infty, 2, 3]$
$\Pi_2 = \{\{a_1\}, \{a_2, a_3\}\}$	$[2, 1, 1]$
$\Pi_3 = \{\{a_1, a_3\}, \{a_2\}\}$	$[\infty, 5, \infty]$
$\Pi_4 = \{\{a_1, a_2\}, \{a_3\}\}$	$[1, 5, 2]$
$\Pi_5 = \{\{a_1\}, \{a_2\}, \{a_3\}\}$	$[6, 5, 8]$

TAB. 6 – Vecteur de concessions des partitions

	$\{\{a_1\}, \{a_2\}\}$	$\{\{a_1, a_2\}\}$
LPO_1		✓
LNS_1		✓
LNS_2	✓	
LR_2	✓	
$\vec{c}(\Pi)$	$[\infty, 1]$	$[1, \infty]$

TAB. 7 – Satisfaction des concepts locaux

site que 2 agents concèdent 2 fois leurs préférences pour trouver une partition leximin-stable.

La Figure 5 présente la concession moyenne des agents et leur écart-type sur 1000 jeux. Bien qu'augmentant légèrement avec le nombre d'agents, elles varient entre $0,08$ et $0,11$ entre 3 et 7 agents. Ainsi, de manière générale, un agent n'est contraint à concéder sur ses préférences que dans 1 jeu sur 10. L'augmentation de l'écart-type indique que plus les agents sont nombreux, plus ils doivent concéder sur leurs préférences pour obtenir une partition leximin-stable. Toutefois, 98% des jeux avec 5 agents n'ont au plus que deux concessions. Ce chiffre tombe à 89% sur les jeux à 7 agents.

Ainsi, s'il est possible de trouver une partition leximin-stable où tous les agents concèdent au maximum leurs préférences, il apparaît que ces cas sont extrêmement rares [2] et que, dans leur grande majorité, les jeux de coalitions hédoniques à double profil sont stables avec un très faible nombre de concessions.

2. Nous n'avons jamais observé ce cas dans nos expérimentations.

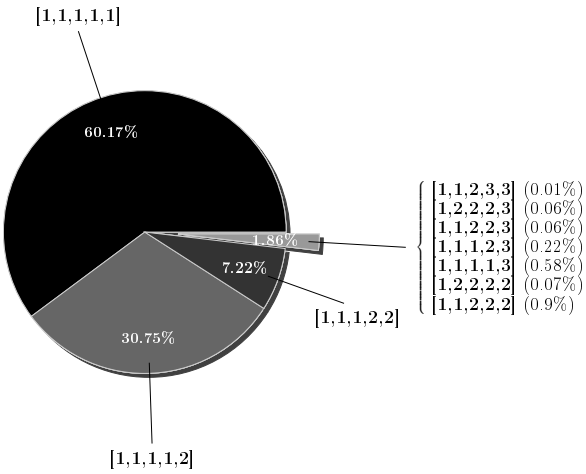


FIG. 4 – Proportions des concessions

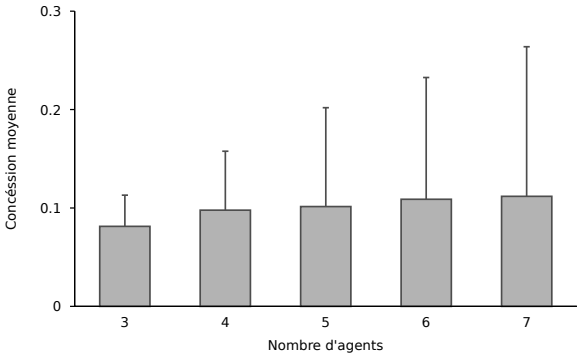


FIG. 5 – Concessions moyennes par agent

5 Conclusion et perspectives

Dans cet article, nous considérons un problème de la formation de coalitions hédoniques où des agents hétérogènes expriment des concepts de solution différents. Pour ce faire, nous proposons les *jeux de coalitions hédoniques à*

concepts de solution multiples (MHG) qui utilisent des *concepts de solution locaux* et les *jeux de coalitions hédoniques à double profil* (HG2P) où les agents expriment des préférences entre ces concepts locaux. Trouver une partition stable revient alors respectivement à trouver une partition qui satisfait les concepts de solution locaux de chaque agent au sens de la *leximin-concession*. Nous avons montré que les concepts de solution locaux satisfont les mêmes propriétés que leurs équivalents globaux, en particulier les propriétés d'inclusion, et stabilisent des partitions qui n'appartiennent à aucun concept global canonique. De plus, nos expériences montrent que, si quelques rares jeux nécessitent plusieurs concessions pour avoir une solution, la majorité des jeux ont une solution où seulement un ou deux agents doivent concéder d'un rang dans leurs préférences.

Ce travail ouvre plusieurs perspectives sur les modèles en eux-mêmes et leur intégration dans un cadre plus large. Comme nos modèles permettent d'obtenir des partitions stables qui ne correspondent à aucun concept de solution canonique couramment étudié, il serait intéressant de caractériser ces partitions et d'en étudier les propriétés. De plus, puisque nos modèles s'appuient sur plusieurs concepts de solution, analyser leur complexité en moyenne plutôt qu'au pire cas semble pertinent. Enfin, nous travaillons actuellement sur un protocole de formation de coalitions décentralisé pour les HG2P. Dans ce contexte, il serait intéressant de se pencher sur les critères permettant à un agent de concéder et une analyse des conséquences des différentes concessions (sur les concepts de solution) vis-à-vis de la satisfaction (sur les coalitions) des agents est nécessaire. En termes de perspectives plus larges, ce travail s'inscrit au sein d'un projet portant sur la modélisation d'agents éthiques. Or, les concepts de solution peuvent être associés à des *valeurs humaines* [12] que les agents doivent respecter. Par exemple, dans les jeux de coalitions à utilité transférable, la valeur de Shapley peut être associée l'*équité* ou la valeur de Nowak à la *solidarité* [9]. Il serait donc intéressant de spécifier en termes de valeurs les préférences des agents et d'intégrer ce travail dans une architecture d'agent éthique, comme par exemple celle de Cointe *et al.* [4].

Remerciements

Ce travail a été réalisé dans le cadre du projet ANR ETHICAA (ANR-13-CORD-0006).

Références

[1] Haris Aziz, Felix Brandt, and Hans Georg Seedig. Computing desirable partitions in additively separable hedonic games. *Artificial Intelligence*, 195 :316–334, 2013.

[2] Coralio Ballester. NP-completeness in hedonic games. *Games and Economic Behavior*, 49(1) :1–30, 2004.

[3] Anna Bogomolnaia and Matthew O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2) :201–230, 2002.

[4] Nicolas Cointe, Grégory Bonnet, and Olivier Boissier. Ethical judgment of agents' behaviors in multi-agent systems. In *15th AAMAS*, pages 1106–1114, 2016.

[5] Fabien Delecroix, Maxime Morge, Thomas Nachtergaelle, and Jean-Christophe Routier. Multi-party negotiation with preferences rather than utilities. *Int. J. of Cloud Computing*, 12(2) :27, 2016.

[6] Jacques H. Dreze and Joseph Greenberg. Hedonic coalitions : Optimality and stability. *Econometrica*, pages 987–1003, 1980.

[7] Edith Elkind and Michael Wooldridge. Hedonic coalition nets. In *8th AAMAS*, pages 417–424, 2009.

[8] Ahmadreza Ghaffarizadeh and Vicki H. Allan. History based coalition formation in hedonic context using trust. *Int. J. of AI & Applications*, 4(4) :1–8, 2013.

[9] Andrzej Nowak and Tadeusz Radzik. A solidarity value for n-person transferable utility games. *Int. J. of Game Theory*, 23 :43–48, 1994.

[10] Dominik Peters and Edith Elkind. Simple causes of complexity in hedonic games. In *24th IJCAI*, pages 617–623, 2015.

[11] Jeffrey S. Rosenschein and Gilad Zlotkin. Designing conventions for automated negotiation. *AI magazine*, 15(3) :29, 1994.

[12] Shalom H. Schwartz. An overview of the Schwartz theory of basic values. *Online Readings in Psychology and Culture*, 2(1) :11, 2012.

[13] Lloyd S. Shapley and Martin Shubik. Quasi-cores in a monetary economy with nonconvex preferences. *Econometrica*, pages 805–827, 1966.

[14] Shao Chin Sung and Dinko Dimitrov. On myopic stability concepts for hedonic games. *Theory and Decision*, 62(1) :31–45, 2007.

Posters

Modélisation multi-agent de la cohésion sociale en situation de crise

Carole Adam carole.adam@imag.fr
Julie Dugdale julie.dugdale@imag.fr
Catherine Garbay catherine.garbay@imag.fr

Laboratoire d'Informatique de Grenoble,
Université Grenoble-Alpes, France

1 Introduction

La cohésion sociale est un phénomène de plus en plus important à comprendre dans la situation politique et sociale actuelle. Mieux appréhender sa dynamique sociale permettrait d'en soutenir les dimensions positives (solidarité) tout en prévenant ses aspects négatifs (communautarisme). Les situations de crise offrent des contextes particulièrement pertinents pour observer la cohésion sociale, sa dynamique et son impact. Ce sont des situations où ses fondements peuvent se trouver ébranlés, et où peuvent émerger des formes nouvelles et imprévues de cohésion, positives (*e.g.* altruisme et solidarité pendant les attentats avec le phénomène "porte ouverte") ou négatives (*e.g.* racisme, manque de communication et d'ouverture entre organisations impliquées). Ces situations mettent en jeu deux points de vue complémentaires : celui des victimes et celui des gestionnaires de crise.

La modélisation et la simulation de ces phénomènes ont encore été peu abordés dans la littérature. Nous proposons ici une première approche en argumentant que trois facteurs peuvent influencer la cohésion et sa dynamique : émotions, connaissances mutuelles, et règles institutionnelles. En effet, les règles institutionnelles définissent des comportements prescrits et peuvent faciliter la gestion des émotions difficiles (empathie du groupe), mais les émotions peuvent inciter les individus à s'en écarter ; la connaissance et la compréhension de situations dynamiques sont vitales pour décider comment agir et quelles règles appliquer, mais ne sont pas toujours partagées ; enfin les règles institutionnelles peuvent être mal connues, mal appropriées ou contradictoires. Ces facteurs ont déjà été étudiés indépendamment mais pas leur combinaison dans une dynamique de cohésion sociale. Nous proposons donc de les combiner dans un modèle d'agent unifié rendant compte de leur impact sur la dynamique des croyances et des décisions de l'agent, modèle destiné à être intégré dans des outils de simulation et de support à la décision pour les gestionnaires de crise.

2 Modèle de la cohésion

Dans la version complète de cet article [1] nous discutons en détail les interactions de ces 3 facteurs (émotions, connaissances mutuelles, normes) entre eux et avec la cohésion sociale.

Dans des travaux précédents, nous avons fourni une formalisation logique et une implémentation des émotions pour des agents BDI [2]. Nous avons aussi déjà analysé l'émergence et la diffusion de connaissance mutuelle d'un point de vue à la fois théorique et pratique [3] grâce à la logique BDO. Enfin nous avons conçu une plateforme de simulation participative pour la préparation à la gestion de risques naturels [5] facilitant la coordination de multiples organisations ayant des normes différentes, basé sur un modèle BDI des normes. La logique BDI permet donc de capturer ces trois éléments grâce à ses opérateurs de normes, de croyance mutuelle, et à l'expression des émotions en termes des d'autres attitudes mentales.

Nous proposons maintenant une extension de l'architecture BDI intégrant ces concepts et leurs interactions (Fig 1) afin d'analyser la cohésion sociale. Notre but est d'implémenter cette architecture dans un simulateur GAMA [4] pour explorer de manière réaliste la dynamique de la cohésion sociale en situation de crise.

Le processus de comportement d'un agent avec cette architecture suit donc un cycle standard de perception-décision-action, dans lequel nous mettons en valeur l'influence des croyances mutuelles, des normes, et des émotions. L'agent **perçoit** d'abord le monde (*percept*), y compris les actions des autres agents, et **met à jour** sa base de croyances (qui contient en particulier les croyances mutuelles, *Mbel*, concernant les autres agents) et ses désirs en conséquence. Les désirs sont ensuite **filtrés** selon les croyances (ce qui est possible dans le contexte) et les normes (ce qui est permis) pour **décider** des intentions de l'agent. En parallèle, les attitudes mentales de l'agent (croyances, désirs, intentions) ainsi

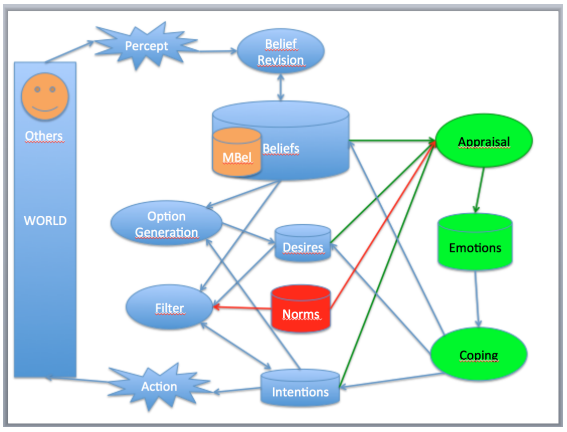


FIGURE 1 – Architecture d'un agent BDI étendu avec notre modèle

que les normes peuvent générer des **émotions** via un processus d'*appraisal*. Ces émotions influencent alors ces mêmes attitudes mentales via le processus de *coping*. Finalement, l'agent planifie sa prochaine **action** en fonction de son intention courante. Plus formellement, le cycle de contrôle de l'agent BDI suit donc l'algorithme figuré ci-dessous (B = Belief, D = Desire, I = Intention, E = Emotion N = Normes).

```
B = B0, I = I0, N = N0
while (true) do
  get next percept p
  B = belief-revision(B,p)
  D = options(B,I)
  E = appraisal(B,D,N)
  I = filter(B,D,I,N)
  B,D,I = coping(E,B,D,N)
  pi = plan(B,I,N)
  execute(pi)
end
```

3 Scénario d'exemple

Dans la version longue de ce travail [1] nous avons modélisé un scénario de réponse à une inondation en termes de cet algorithme de raisonnement. Ce scénario illustre la dynamique de la cohésion en lien avec les attitudes mentales et émotions et fournit des pistes pour son analyse. Il sert donc de preuve de concept avant l'implémentation de notre architecture d'agent, que nous laissons pour des travaux futurs.

Au cours de ce scénario, on voit par exemple comment la cohésion évolue au sein des différents groupes (pompiers, mairie...) et entre les groupes, en fonction des différents facteurs modélisés ici. La panique de certains habitants et le non respect des normes entraîne un manque de cohésion au sein de la population conduisant à des embouteillages. Le manque de connaissance mutuelle et le non respect de certaines normes diminue la cohésion entre pompiers et mairie ; finalement la concertation entre eux permet de résoudre ces problèmes et rétablir la cohésion nécessaire à une réponse coordonnée.

4 Conclusion

Nous avons défendu dans cet article une vision de la cohésion sociale comme une dynamique résultant de l'entrelacement entre plusieurs facteurs : émotions, connaissances mutuelles, et règles institutionnelles. Nous avons proposé un modèle d'agent BDI permettant d'intégrer ces 3 facteurs, en vue d'une implémentation dans la plateforme GAMA. Notre but à long terme est en effet de développer un jeu sérieux simulant une situation de crise afin de tester l'influence de différents aspects (qualité des plans de gestion et réglementations, coordination des gestionnaires) sur l'émergence des phénomènes de cohésion. Un travail important reste à faire au plan de la modélisation, en particulier pour prendre en compte la dynamique des normes, leur caractère émergent, ou pour intégrer d'autres facteurs de cohésion. Comment analyser la cohésion, son évolution, comment évaluer son influence comme négative ou positive, demeurent des champs de recherche ouverts.

Remerciements

Ce travail est soutenu par le Laboratoire d'Informatique de Grenoble via le projet SNICS de l'appel Emergence 2017.

Références

[1] C. Adam, J. Dugdale, and C. Garbay. Modélisation multi-agent de la cohésion sociale en situation de crise. Technical Report RR-53, Laboratoire d'Informatique de Grenoble, 2017.

[2] C. Adam, A. Herzig, and D. Longin. A logical formalization of the OCC theory of emotions. *Synthese*, 168(2) :201–248, 2009.

[3] J. Dugdale, B. Pavard, and J. L. Soubie. A pragmatic development of a computer simulation of an emergency call centre. In Rose Dieng et al., editor, *Designing Cooperative Systems. Frontiers in Artificial Intelligence and Applications*. IOS Press, 2000.

[4] P. Taillandier, M. Bourgais, P. Caillou, C. Adam, and B. Gaudou. A situated BDI agent architecture for the gama modelling and simulation platform. In *MABS @ AAMAS*. MABS, 2016.

[5] L. Thévin, J. Dugdale, O. Boissier, and C. Garbay. Evaluating plans and human response using a normative multi-agent system. In *ISCRAM*, 2016.

Délégation de tâches sur la plateforme multi-agents embarquée MERMAID

T. Inguère[a b] tifaine.inguere.etu@univ-lemans.fr
V. Renault[a] valerie.renault@univ-lemans.fr
F. Carlier[a] florent.carlier@univ-lemans.fr

[a]Laboratoire Centre de Recherche en Éducation de Nantes,
Université du Maine - Le Mans, France

[b]Entreprise STMicroelectronics,
Le Mans, France

Résumé

Une allocation des tâches pertinente est essentielle afin d'optimiser au mieux les ressources limitées de calcul, de mémoire ou la consommation d'énergie. Nous proposons d'apporter les algorithmes multi-agents au cœur de ces systèmes embarqués pour permettre la négociation entre les unités de ressources matérielles. Nos travaux reposent sur la conception d'une plateforme appelée MERMAID, dédiée à un environnement matériel.

1 Introduction

Dans les systèmes embarqués (SE), par souci de fiabilité, les fabricants déterminent lors de la phase de conception les tâches auxquelles les ressources sont dédiées. Nous proposons d'effectuer une optimisation des ressources existantes en ré-affectant des tâches en temps-réel à des composants non alloués. Nous définissons la gestion d'une ressource par l'allocation de certaines tâches à cette dernière à un instant donné. La délégation d'une tâche à une ressource est définie par la ré-affectation de cette tâche à une cible différente. Nous proposons la possibilité d'effectuer cette délégation au terme d'une négociation entre les ressources concernées. Pour effectuer cette délégation, nous proposons d'intégrer le protocole de négociation CNIP [2] des systèmes multi-agents (SMA) au cœur des systèmes embarqués. Nous nous concentrons sur une intégration à un SE existant. Notre perspective de recherche apporte des contraintes embarqués strictes aux systèmes multi-agents. Pour se faire, nous présentons la plateforme MERMAID (*Managed Embedded Resources by Multi-Agents applied to Integrated and Distributed systems*) et des expérimentations pour effectuer des délégations de tâches entre agents au sein des systèmes embarqués.

2 MERMAID : une plateforme agent embarquée

La plateforme MERMAID peut se situer dans une couche matérielle au plus bas niveau possible de différents systèmes embarqués. Nous avons adopté une architecture en plusieurs couches afin d'assurer sa portabilité. Cette architecture s'axe sur deux parties principales. La première correspond au protocole de communication. Nous utilisons D-Bus. Il permet les interactions entre les processus agents. Il permet également les interactions avec les processus non-agents du système. La deuxième représente la couche administrative de notre SMA permettant la gestion des agents. MERMAID se base sur la norme FIPA (*Foundation for Intelligent Physical Agents*) [1].

2.1 Expérimentation : délégation de tâches

Les agents sont définis en s'inspirant du modèle IODA [3]. Le point qui nous intéresse dans ce modèle est la formalisation de nos agents à partir de leurs interactions, indépendemment de leur réalisation. Un "service" est défini comme la capacité d'un agent à réaliser une tâche donnée. Pour expérimenter notre méthode de délégation de tâche, nous considérerons des agents proposant un service de traitement d'image. La tâche observée est l'affichage d'une image et la possibilité de zoomer sur celle-ci. Nous évaluons le coût administratif que représente l'application du protocole CNIP. Nous déterminons que le nombre de messages impliqué correspond dans le cas le plus coûteux à 3 fois le nombre d'agents proposant le même service. Nous établissons notre expérimentation pour un nombre d'agents concernés variant de 2 à 100, soit de 6 à 300 messages générés. Le temps de réponse induit par la négociation relative à ce nombre d'agents est relevé. Nous considérons des temps

limites à ne pas dépasser correspondant aux standards du domaine du traitement d'image.

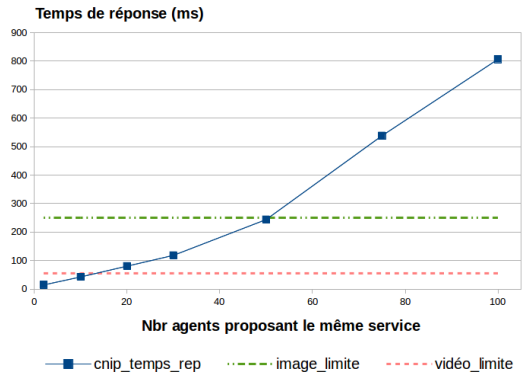


FIGURE 1 – Temps de réponse de la résolution du protocole de négociation CNIP en fonction du nombre d'agents impliqués.

2.2 Expérimentation : Capacité de modularité

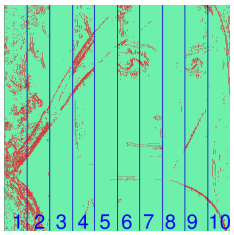
Nous posons l'hypothèse que la répartition du travail entre les agents doit pouvoir permettre une modularité au niveau du choix de la méthode de calcul pour une même image. Nous expérimentons notre hypothèse en proposant deux méthodes d'interpolation différentes. L'une représente une consommation CPU forte, supérieure à 50%, l'autre une consommation CPU faible, inférieure à 10%. Les agents appliquent l'une ou l'autre en fonction d'une valeur seuil. Dans le cadre d'un CPU déjà utilisé à plus de 60%, nous augmentons notre seuil. Le résultat est moins précis, mais l'agent s'adapte ainsi à la disponibilité de son environnement matériel en réduisant au maximum l'utilisation de la méthode la plus coûteuse.

Pour illustrer nos résultats, nous faisons ressortir visuellement la méthode d'interpolation qui aura été utilisée : un pixel vert pour la faible consommation CPU et un pixel rouge pour la forte consommation. En FIGURE 2, nous montrons les résultats obtenus en effectuant le zoom avec 10 agents, et simulons une surcharge CPU pour trois d'entre eux.

3 Conclusion

Nous avons présenté une possibilité de déléguer des tâches entre différents processus et la capacité de modularité de notre SMA intégré à

Lena, image d'origine



zoom par 10 agents (Agents n°3, 5 et 8 avec CPU à plus de 60%)

FIGURE 2 – Résultats d'un zoom partagé entre plusieurs agents Pixel sur l'image standard "Lena"

un contexte embarqué. L'utilisation d'un protocole de négociation agent permet une délégation de tâches flexible et dynamique qui amène bien un gain en terme du nombre de services traités, tout en respectant un coût administratif moindre pour notre SE. La capacité des agents à s'adapter à leur environnement a démontré une augmentation de la modularité sans augmenter la complexité. Pour la suite de nos recherches, nous étendrons ces processus de négociations et de modularité à différentes cartes embarquées, permettant ainsi un partage de ressources de différentes unités matérielles.

Les travaux menés dans cet article bénéficient d'un financement de thèse CIFRE en partenariat entre le laboratoire Centre de Recherche en Éducation de Nantes (CREN) et l'entreprise ST-Microelectronics Le Mans avec la collaboration de Stéphane Henry, Directeur Innovation et de Emmanuel Grandin, Architecte Système.

Références

[1] FIPA consortium. *FIPA Communicative Act Library, Specification and FIPA ACL Message Structure Specification*. Technical report, 2003.

[2] El Hassan Et-Tolba, Mohammed Ouassaid, and Mohamed Maaroufiand. An implementation of fipa contract net interaction protocol adapted for smart home agents simulation. *Renewable and Sustainable Energy Conference (IRSEC)*, December 10-13 2015.

[3] Yoann Kubera, Philippe Mathieu, and Sébastien Picault. Ioda : An interaction-oriented approach for multi-agent based simulations. *Journal of Autonomous Agents and Multi-Agent Systems*, 23(3) :303–343, 2011.

Modélisation d'une tâche collaborative sous forme d'engagements sociaux

Jean-Baptiste Louvet[a] jeanbaptiste.louvet@insa-rouen.fr
Nathalie Chaignaud [a] nathalie.chaignaud@insa-rouen.fr
Laurent Vercouter[a] laurent.vercouter@insa-rouen.fr
Guillaume Dubuisson Duplessis[b] gdubuisson@isir.upmc.fr
Jean-Philippe Kotowicz[c] j.kotowicz@insa.ueuromed.org

[a]Normandie Univ, INSA Rouen, UNIHAVRE, UNIROUEN, LITIS, 76000 Rouen, France

[b]Sorbonne Universités, UPMC Univ Paris 06, CNRS, ISIR, 75005 Paris, France

[c]UEMF, INSA Euro-Méditerranée, Fès, Maroc

Mots-clés : *Interaction humain-machine, recherche collaborative de documents, engagements sociaux*

1 Introduction

La recherche de documents, dans une base de données fermée indexant des documents présélectionnés à partir de sources fiables, est une tâche complexe pour des utilisateurs non experts. Pour trouver les documents pertinents à une recherche, les interfaces permettant la formulation de requêtes spécifiques sont peu utilisées car elles nécessitent de bien connaître la terminologie du domaine de recherche. Selon le besoin d'information de l'utilisateur, cette tâche peut nécessiter une assistance externe. Notre but est donc de concevoir un agent logiciel capable de collaborer avec un utilisateur pour l'aider dans sa recherche de documents.

Nous adoptons une démarche cognitive en étudiant un corpus d'interaction humain-humain (h-h) sur une tâche de recherche collaborative de documents impliquant un expert et un utilisateur.

Nous nous intéressons aux relations qui lient la tâche de recherche collaborative de documents et l'interaction avec l'utilisateur que notre agent assistant doit gérer. Nous montrons que le formalisme [1] utilisé pour modéliser les jeux de dialogue peut être enrichi pour décrire une tâche collaborative. Notre modèle fait le lien entre une structure de haut niveau (la tâche) et des interactions de bas niveau (jeux de dialogue). Chaque étape du scénario est décrite dans notre modèle et utilise la notion de *déclencheurs* qui permet de spécifier les processus délibératifs de notre agent assistant.

2 Scénario de recherche collaborative de documents

Pour comprendre les processus collaboratifs de la recherche de document d'un utilisateur assisté par un expert humain, nous avons étudié une interaction h-h. Cette étude est fondée sur l'analyse du corpus collecté au cours du projet Cogni-CISMeF [2].

L'analyse du corpus a permis d'identifier et de caractériser les différentes phases des dialogues du corpus jouant un rôle dans l'avancement de la tâche [3]. Nous avons identifié cinq phases :

- la verbalisation : c'est l'établissement du sujet de la recherche entre l'utilisateur et l'expert. Elle commence par une formulation de la demande de l'utilisateur et peut être suivie par des précisions spontanées de la part de celui-ci.
- la construction de la requête : c'est l'alignement des termes de la verbalisation de l'utilisateur avec la terminologie du moteur de recherche pour remplir le formulaire de requête ;
- le lancement de la requête : c'est simplement l'exécution de la requête courante.
- l'évaluation des résultats : l'expert et l'utilisateur évaluent le résultats de la requête. S'ils sont satisfaisants, la recherche se termine, sinon la requête doit être modifiée ;
- la réparation de la requête : l'expert et l'utilisateur mettent en place des tactiques pour modifier la requête tout en respectant le besoin d'information.

3 Modèle de dialogue

Le dialogue peut être considéré comme une activité partagée et dynamique nécessitant à la fois un raisonnement délibératif de haut niveau et des actions réactives de bas niveau.

Dubuisson Duplessis [1] propose d'utiliser une architecture hybride réactive/délibérative d'agent collaboratif, dans laquelle les actions conjointes sont encapsulées dans des motifs d'interaction modélisés sous forme de jeux de dialogue. Ces jeux de dialogue sont formalisés à l'aide d'engagements sociaux enregistrés dans un tableau de conversation. Les engagements sociaux lient un émetteur à ses interlocuteurs [5]. Le tableau de conversation décrit l'état du dialogue entre deux interlocuteurs à un instant donné. Il montre la partie publique du contexte dialogique, supposé strictement partagé.

Un jeu de dialogue est une activité conjointe entre un initiateur et un partenaire. Les règles d'un jeu de dialogue spécifient les coups dialogiques attendus de la part de chaque participant, appelés à jouer leur rôle en exécutant les coups attendus selon l'état courant du jeu.

4 Modéliser une tâche collaborative à l'aide d'engagements sociaux

Nous considérons qu'une tâche peut être divisée en sous-tâches que nous appelons étapes. Chaque étape est décrite par une table, appelée *table d'étape* [4], composée de trois parties : le *nom de l'étape*, les *conditions d'entrée* et une liste de *comportements attendus* de la part de chaque participant au dialogue. Un comportement attendu est défini par : un **jeu de dialogue attendu**, avec son émetteur et son contenu (action ou proposition) ; les **sorties** possibles de ce jeu de dialogue ; et un **déclencheur** (seulement si l'émetteur est l'agent logiciel) qui décrit les conditions à remplir pour jouer le jeu attendu.

Une table d'étape permet de définir ce qui est conventionnellement attendu de la part de chaque participant dans une étape. Cette table décrit les jeux de dialogue attendus dans cette étape et les modifications qu'ils apportent au tableau de conversation en termes d'engagements sociaux. Le comportement de l'agent peut être adapté en modifiant uniquement leur contenu.

5 Conclusion

Notre travail se fonde sur l'étude d'un corpus d'interaction h-h pour une tâche collaborative de RD. Nous avons construit un scénario à partir d'un corpus h-h, dont nous nous inspirons pour concevoir un agent assistant collaborant avec un utilisateur dans sa recherche de documents.

La notion de déclencheur dans notre modèle permet de décrire le comportement délibératif de notre agent. Son raisonnement peut être très réactif et de bas niveau (exprimé par des règles simples) ou au contraire plus sophistiqué en mettant en œuvre des processus de décision de plus haut niveau.

Références

[1] G. Dubuisson Duplessis, A. Pauchet, N. Chaignaud, and J-Ph. Kotowicz. A Conventional Dialogue Model Based on Dialogue Patterns. *International Journal on Artificial Intelligence Tools*, 26(1) :1–23, 2017.

[2] A. Loisel, G. Dubuisson Duplessis, N. Chaignaud, and J-Ph. Kotowicz. A conversational agent for information retrieval based on a study of human dialogues. In *International Conference on Agent and Artificial Intelligence (ICAART)*, pages 312–317, 2012.

[3] Jean-Baptiste. Louvet, Guillaume. Dubuisson Duplessis, Nathalie. Chaignaud, Jean-Philippe. Kotowicz, and Laurent. Vercouter. Recherche collaborative de documents : comparaison assistance humaine/automatique. In *Journées Francophones d'Ingénierie des Connaissances*, pages 161–166, 2016.

[4] Jean-Baptiste Louvet, Guillaume Dubuisson Duplessis, Nathalie Chaignaud, Laurent Vercouter, and Jean-Philippe Kotowicz. Modeling a collaborative task with social commitments. *Proceedings of the 21st International Conference on Knowledge Based and Intelligent Information and Engineering Systems (KES'17)*, September 2017 (à paraître).

[5] M. P. Singh. Social and psychological commitments in multiagent systems. In *AAAI Fall Symposium on Knowledge and Action at Social and Organizational Levels*, pages 104–106, 1991.

Un méta-modèle pour représenter les normes dans un contexte multi-institutionnel territorialisé

J.-P. Müller[a]
jean-pierre.muller@cirad.fr

S. Raharivelo[b]
sitraka_oliva@yahoo.fr

[a] UPR GREEN
CIRAD, Montpellier, France

[b] Département de Mathématiques et Informatique,
Université d'Antananarivo, Madagascar

1 Contexte et problématique

Le défi principal est de trouver un équilibre entre exploitation et conservation des ressources naturelles permettant un développement durable (donc écosystémique, économique et social) des populations locales malgaches. Ce défi doit être relevé dans un contexte de pluralisme juridique incluant les traditions locales et les lois modernes. Dans ce contexte, comprendre comment ces mécanismes existants (tradition, administration forestière, lois environnementales, etc.) ou futurs (p. ex. les paiements des services environnementaux) s'imbriquent et comment les populations y répondent est de la plus haute importance pour conseiller les porteurs d'initiatives locales et les décideurs politiques. Pour ce faire, nous utilisons les Systèmes Multi-Agents (SMA) pour modéliser à la fois les dynamiques biophysiques et sociales [1]. Ces dernières doivent reproduire les comportements des acteurs soumis à une multiplicité de systèmes de régulations qui permettent ou restreignent leurs activités sur les différentes ressources dans le temps et dans l'espace puisque la gestion est, en général, territorialisée.

2 Modéliser avec les SMA

Les SMA nous fournissent de nombreux concepts et outils permettant d'aborder cette question. Les SMA normatifs permettent de représenter les régulations en proposant diverses formalisations des normes [2] et la manière dont elles peuvent être prises en compte par les agents [3]. Les SMA organisationnels permettent de représenter les structures sociales en introduisant la notion de groupes dans lesquels les agents jouent différents rôles [4]. Les SMA institutionnels [5] permettent de représenter les systèmes de régulations en fusionnant ces deux approches, associant les normes aux rôles et aux groupes.
Si certaines formalisations des normes prennent en compte le temps, elles ne prennent pas en compte l'espace et elles ne sont pas suffisamment expressives pour expliciter les notions de rôle, de temps et d'espace. L'objectif de ce travail est donc d'intégrer ces notions dans une nouvelle formalisation des institutions, des organisations et des normes. Cette formalisation servira à implémenter un outil permettant de décrire facilement la multiplicité des systèmes régulations et de calculer en fonction de la situation temporelle, spatiale et sociale d'un acteur les normes qui s'appliquent sur lui et donc les conséquences des choix normatifs.

3 La mise en œuvre

Pour ce faire, nous avons décidé de définir un Langage Spécifique au Domaine (DSL en anglais), à savoir une syntaxe textuelle qui permet de décrire les éléments ci-dessus. Pour cela, nous avons mis en œuvre une approche d'Ingénierie Dirigée par les Modèles (IDM) qui consiste à définir les concepts à mettre en œuvre sous la forme d'un méta-modèle source, appelé aussi syntaxe abstraite, indépendant de toute plateforme existante. On peut ensuite définir une syntaxe concrète sous la forme soit d'une grammaire pour une syntaxe concrète textuelle, soit d'une représentation graphique.

Nous avons donc proposé un méta-modèle, et avons esquissé sa syntaxe textuelle.

4 Résultats obtenus

L'objectif est de mettre à disposition un outil permettant de décrire facilement les institutions, organisations et normes et de calculer en fonction de la situation temporelle, spatiale et sociale d'un acteur quelles sont les normes qui s'appliquent à lui. Nous avons présenté un langage spécifique au domaine (DSL en anglais), permettant de décrire ces différentes notions. Pour cela, nous avons présenté un méta-modèle permettant de faire la synthèse de toutes les notions nécessaires pour atteindre notre objectif. Cette synthèse a nécessité de faire des choix sur la sémantique des différentes notions nécessaires, de les situer et de les argumenter par rapport à la littérature. Par ailleurs, nous avons dû ajouter le raisonnement spatial pour pouvoir traiter de notre objet, à savoir la gestion territorialisée. Le méta-modèle, vu comme une syntaxe abstraite, a ensuite été utilisé pour définir un langage textuel de création des institutions, organisations, normes, etc. dont nous avons esquissé quelques éléments.

L'étape suivante est de mettre en œuvre la vérification des conditions des normes afin de permettre aux utilisateurs de visualiser les conséquences de leur choix normatifs sur les possibilités d'actions offertes aux acteurs en fonction des lieux, des périodes et de leur contexte social, c'est-à-dire des organisations auxquelles ils appartiennent.
Pour mettre en œuvre un SMA, chaque agent aura ainsi le répertoire de ses obligations, interdictions et permissions en chaque lieu et à chaque moment. Il restera à exploiter ce répertoire pour planifier les activités.

Remerciements

Ce travail a été partiellement financé par une Bourse du Gouvernement Français (BGF).

Références

[1] S. Aubert et J. P. Müller, « Incorporating institutions, norms and territories in a generic model to simulate the management of renewable resources », *Artificial Intelligence and Law*, vol. 21, p. 47‑48, 2013.

[2] G. Boella, L. Torre, et H. Verhagen, « Introduction to normative multiagent systems », *Computational and Mathematical Organization Theory*, vol. 12, nº 2‑3, p. 71–79, oct. 2006.

[3] F. L. y Lopez, M. Luck, et M. d'Inverno, « A normative framework for agent-based systems », *Computational and Mathematical Organization Theory*, vol. 12, nº 2‑3, p. 227–250, oct. 2006.

[4] J. Ferber et O. Gutknecht, « A meta-model for the analysis and design of organizations in multi-agent systems », in *International Conference on Multi-Agent Systems*, 1998, p. 128–135.

[5] J.-P. Müller et S. Aubert, « Formaliser les rôles et les territoires par les systèmes multi-agents institutionnels. », in *Systèmes multi-agents : ouverture, autonomie et co-évolution*, Honfleur, France, 2012, p. 13‑22.

SmartGov : architecture générique exploitant des données des smart cities pour co-concevoir des politiques crédibles

S. Pageaud[a,b]
simon.pageaud@liris.cnrs.fr

V. Deslandres[a]
veronique.deslandres@liris.cnrs.fr

S. Hassas[a]
salima.hassas@liris.cnrs.fr

V. Lehoux[b]
vassilissa.lehoux@xrce.xerox.com

[a]Laboratoire LIRIS, Université Claude Bernard Lyon 1, Lyon, France

[b]XEROX Research Centre Europe, Meylan, France

1 Introduction

La mise à disposition de données fournies de la ville intelligente (*smart city*) est croissante et va dans le futur représenter un vivier d'informations incontournable pour la prise de décision politique, avec notamment un accès simplifié aux données des usagers [3]. La question est de savoir comment les décideurs politiques vont pouvoir utiliser ces données pour construire des politiques. Dans le cadre des villes intelligentes, une solution peut être de simuler différentes politiques urbaines dans un cadre réaliste, et observer leur impact sur un horizon de temps donné. Plusieurs problèmes doivent alors être considérés : l'exploitation des données [3], la modélisation pertinente des populations d'usagers pour déterminer leur impact sur la simulation [4] et l'échange entre le décideur politique et l'outil d'aide à la décision [5]. Ce travail propose de répondre à ces contraintes en introduisant SmartGov, un modèle générique pour la co-conception de politiques utilisant une simulation multi-agent crédible et participative. Les objectifs sont les suivants : d'une part introduire des méthodes pour créer une simulation crédible et réaliste à partir des données qui seront disponibles avec la ville intelligente de demain ; d'autre part proposer un formalisme pour manipuler et évaluer les politiques dans le cadre de l'aide à la décision et enfin mettre en place une approche réflexive pour permettre une co-conception de politiques auto-adaptatives.

2 Simulation crédible du monde

L'apprentissage avec la fouille de données permet de produire des agents crédibles [1], nécessaire à la pertinence de la production du simulateur. Remondino et Correndo [4] proposent deux approches pour qualifier la fouille de données (Fig. 1). D'une part l'approche endogène : fouille de données pour enrichir le comportement de l'agent et ainsi proposer un agent plus crédible. D'autre part l'approche exogène : fouille de données sur les résultats de la simulation pour dégager des motifs qui vont enrichir la simulation. Ces deux approches permettent à la simulation de s'enrichir au long de son fonctionnement par retour d'expériences.

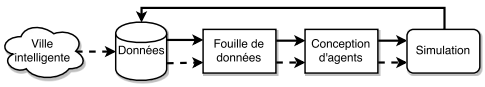


FIGURE 1 – Approche endogène (en pointillés) et exogène (trait plein)

Dans SmartGov, nous exploitons ces 2 approches. Les agents ont une architecture modulaire horizontale [2] et pour notre modélisation, ils comportent trois briques renforçant la crédibilité : un moteur de prise de décisions, une personnalité, et des indicateurs sur leur état (Fig. 2).

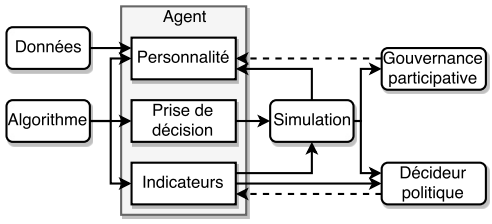


FIGURE 2 – Construction de l'agent par les données et l'algorithme (en gras) et le retour des usagers et du décideur politique (en pointillé)

3 Simulation crédible de politiques publiques

L'objectif est de co-construire des politiques publiques riches et intégrant le décideur dans la conception [5]. La co-construction est permise par couplage des interactions : les résultats des simulations sont présentés au décideur qui peut alors faire des retours pour affiner la politique et reproduire une simulation. Pour pouvoir traiter la politique comme une entité à par entière, nous proposons la définition suivante :

Définition (Politique) *Une* politique *désigne l'application d'un ensemble d'actions sur l'environnement pour un horizon de temps donné afin de satisfaire une ou plusieurs fonctions de coût.*

Le plan d'action représente une succession de politiques à court terme permettant de respecter les objectifs à moyen ou long terme. Des variantes sont proposées au décideur, pour permettre une ouverture vers des alternatives non envisagées initialement, ou pour explorer des voies nouvelles.

Le modèle repose sur des agents "gestionnaires de politiques" qui construisent leur représentation de l'environnement à partir d'une agrégation des indicateurs disponibles dans le monde étudié (Fig. 3). L'action effectuée par le simulateur change la structure de l'environnement et le comportement des agents s'adapte à ces modifications.

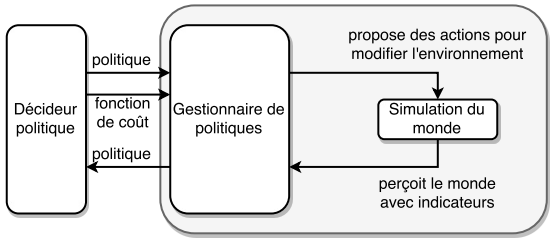


FIGURE 3 – Fonctionnement du gestionnaire de politiques

Le couplage entre les gestionnaires de politiques et la simulation du monde repose sur une représentation crédible du monde sur lequel le décideur politique peut co-concevoir des politiques publiques, qui seront à terme enrichies des retours des usagers concernés.

4 Conclusion

Afin de proposer un outil d'aide à la décision crédible et réaliste, nous nous sommes intéressés dans cet article à prendre position par rapport à la ville intelligente et son rôle dans le futur de la conception de politiques publiques. L'objectif est de présenter la démarche à adopter pour concevoir, étape par étape, un système permettant la co-conception de politiques publiques dans la ville connectée. Ce travail s'inscrit dans la ville intelligente et propose une manière d'établir une preuve de concept de la faisabilité de cette approche générique Développé avec Repast Sympony, notre prototype permet actuellement de tester la dynamique d'interactions entre le simulateur et le décideur pour des politiques de mobilité, sur des villes pour lesquelles on dispose d'informations standard (OSM).

Annexe

Ce projet est financé par la région Auvergne-Rhône-Alpes (financement ARC7). Ce projet est une collaboration entre le LIRIS (Laboratoire d'InfoRmatique en Image et Systèmes d'information) de Lyon et le XRCE (XEROX Research Centre Europe) de Grenoble.

Références

[1] Kevin Darty, Julien Saunier, and Nicolas Sabouret. Extraction de comportements pour l'étude de la crédibilité des agents. *Journées Francophones des Systèmes Multi-Agents*, 21(1) :10, 2013.

[2] Jacques Ferber. *Les systèmes Multi-Agents*, volume 1. 1995.

[3] Rob Kitchin. The real-time city ? big data and smart urbanism. *GeoJournal*, 79 :1–14, 2014.

[4] Marco Remondino and Gianluca Correndo. Data mining applied to agent based simulation. *Proceedings of the 19th European Conference on Modelling and Simulation*, 2005.

[5] Alexey Voinov, Nagesh Kolagani, Michael Keith McCall, Pierre Glynn, Marit Ellen Kragt, Frank Ostermann, Suzanne Alise Pierce, and Palaniappan Ramu. Modelling with stakeholders - next generation. *Environmental Modelling and Software*, 2016.

Intégration d'un simulateur multi-agent dans une plateforme de co-simulation DEVS

T. Paris[a,b] thomas.paris@loria.fr
L. Ciarletta[a,b] laurent.ciarletta@loria.fr
V. Chevrier[a] vincent.chevrier@loria.fr

[a]Laboratoire lOrrain de Recherche en Informatique et ses Applications UMR 7503, CNRS, Université de Lorraine, France

[b]Inria Grand-Est, France

Résumé

*Cet article présente une première réflexion sur l'intégration du simulateur multi-agent NetLogo dans la plateforme de co-simulation (*MECSYCO*).*

Mots-clés : *Système complexe, Système multi-agent, Co-simulation*

Abstract

*This article presents a preliminary study about the integration of a multi-agent simulator (NetLogo) into a co-simulation platform (*MECSYCO*).*

Keywords: *Complex system, Multi-agent system, Co-simulation*

1 Contexte

La Modélisation et Simulation de systèmes complexes est l'un des enjeux actuels majeurs de recherche. L'une des difficultés est de pouvoir combiner plusieurs perspectives d'un même système au sein d'une représentation cohérente (multi-modélisation). Cela implique la gestion de phénomènes à différents niveaux (micro et macro), à différentes échelles (temporelles comme spatiales), et selon différentes perspectives. La gestion de ces hétérogénéités rend nécessaire le développement de nouvelles approches et outils.

L'une des approches prometteuses pour faire face à ces défis est la co-simulation [Gomes et al., 2017]. Elle consiste à faire interagir plusieurs simulateurs au sein d'une même simulation en assurant la synchronisation et les échanges de données. Cela permet la réutilisation de simulateurs existants et donc des outils déjà utilisés dans des domaines spécifiques.

Par ailleurs, l'approche multi-agent est adaptée à la représentation de systèmes composés d'un grand nombre d'entités hétérogènes en interaction, ce qui correspond à la définition des systèmes complexes [Ramat, 2006]. De plus elle autorise une étude de ces systèmes à la fois au niveau des individus et au niveau collectif [Michel et al., 2009]. C'est une approche de choix pour modéliser les systèmes complexes.

La question qui nous intéresse ici est alors *comment peut-on modéliser et simuler un système complexe à partir de plusieurs modèles multi-agents* de ce système, chacun offrant une perspective complémentaire. Nous adopterons une approche de co-simulation, la question revient alors à comment faire interagir différents simulateurs multi-agents pour assurer les échanges d'informations entre eux et synchroniser leur exécution. Nous écartons les approches ad-hoc (qui ne serait pas pérennes et potentiellement sources d'erreurs) ainsi que la réécriture des simulateurs au sein d'un seul (source d'erreurs, de perte de temps, ...).

Nous décrivons ici l'intégration du simulateur NetLogo dans une plateforme de co-simulation basée sur DEVS (MECSYCO : Multi-agent Environment for Complex SYstems CO-simulation)) [1]. Nous nous restreignons ici au couplage spatial où un agent est présent dans un simulateur à la fois et où il n'y a pas d'interaction entre agents de simulateurs différents.

2 Principe

Nous utilisons le formalisme DEVS (Discrete EVent System specification) [Zeigler et al., 2000], qui grâce à sa propriété d'universalité, le place comme un formalisme pivot pour l'intégration de nouveaux formalismes [Vangheluwe, 2000].

L'intégration de simulateurs dans MECSYCO se fait par wrapping (encapsulation) qui consiste à

1. mecsyco.fr

créer une interface autour du formalisme cible de sorte à pouvoir le manipuler comme un modèle atomique DEVS, ce qui permet de réutiliser des modèles existants déjà implémentés.

La création d'un wrapper DEVS pour MECSYCO implique de créer une interface entre le simulateur et les cinq fonctions du protocole de simulation DEVS utilisé par MECSYCO et de spécifier les échanges d'informations (notion de ports DEVS et d'évènements). Ces informations sont regroupées dans un document d'interface (cf Figure 1). Il s'agit d'une description des paramètre initiaux, des ports d'entrée et de sortie ; et du code NetLogo correspondant aux traitements à effectuer pour chaque fonction du protocole de simulation DEVS. Ce document est utilisé par un wrapper (un morceau de code JAVA dans notre cas) dédié à la plateforme MECSYCO. Les questions de gestions du temps, de synchronisation, et d'échange d'information entre les simulateurs sont traitées par les 5 fonctions du protocole DEVS.

L'interopérabilité logicielle est assurée par l'API java permettant de lancer et d'interagir avec des modèles. Cette librairie permet d'exécuter des procédures de type *command* pour impacter le modèle et des procédures de type *report* pour récupérer des données du modèles.

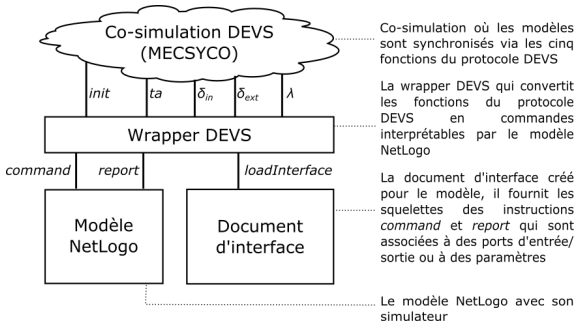


FIGURE 1 – Principe du wrapper NetLogo.

3 Expérimentations

Nous avons mené plusieurs expérimentations, notamment en reprenant les exemples tutoriels fournis avec MECSYCO. Ces exemples seront disponibles avec la prochaine mise à jour de la plate-forme.

Elles ont montré qu'il était alors possible d'intégrer des modèles décrits en NetLogo dans MECSYCO et de les faire interagir entre eux et avec d'autres simulateurs qu'eux sans avoir à développer spécifiquement du code hormis dans le document d'interface sous la forme de code NetLogo. Il devient alors possible à une personne ne connaissant que NetLogo de réaliser des co-simulations.

4 Bilan

Nous avons donc réalisé un couplage logiciel et formel permettant d'intégrer à MECSYCO une très grande quantité de modèles existants (ceux de NetLogo). Toutefois, notre proposition impose que le modèle soit conçu pour être exploité lors d'une co-simulation ce qui est loin d'être le cas de ceux de NetLogo, de plus elle ne permet pas les interactions entre agents présents dans deux simulateurs différents.

Nous envisageons de poursuivre la confrontation de notre approche à d'autres modèles NetLogo et de passer à d'autres simulateurs (qui outre ces même problèmes conceptuels poseront en plus des problèmes d'intégration logicielle) tels que MATLAB ou Janus par exemple.

Références

[Gomes et al., 2017] Gomes, C., Thule, C., Broman, D., Gorm Larsen, P., and Vangheluwe, H. (2017). Cosimulation : State of the art. *International Mediterranean Modeling Multiconference*.

[Michel et al., 2009] Michel, F., Ferber, J., Drogoul, A., et al. (2009). Multi-agent systems and simulation : a survey from the agents community's perspective. In Uhrmacher, A. and Weyns, D., editors, *Multi-Agent Systems : Simulation and Applications, Computational Analysis, Synthesis, and Design of Dynamic Systems*, pages 3–52. CRC Press - Taylor and Francis.

[Ramat, 2006] Ramat, E. (2006). Introduction à la simulation : principaux concepts. In *Modélisation et Simulation Multi-Agent : application pour les Sciences de l'Homme et de la Société*, pages 37–60.

[Vangheluwe, 2000] Vangheluwe, H. L. (2000). DEVS as a common denominator for multi-formalism hybrid systems modelling. In *Computer-Aided Control System Design, 2000. CACSD 2000. IEEE International Symposium on*, pages 129–134. IEEE.

[Zeigler et al., 2000] Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of modeling and simulation : integration discrete event and continuous complex dynamic systems*. Academic press.

Modèle relationnel pour la coordination des comportements des agents

I. Velontrasina
irene.velontrasina@univ-reunion.fr

D. Payet
denis.payet@univ-reunion.fr

R. Courdier
remy.courdier@univ-reunion.fr

Laboratoire d'Informatique et de Mathématiques,
Université de La Réunion, France

Résumé

Dans un environnement physique, l'hétérogénéité des composants limite souvent les interactions entre les agents. Notre objectif est de concevoir une architecture favorisant les échanges entre les agents. Nous proposons un système de gestion de l'ensemble des comportements basé sur le modèle relationnel des bases de données. Ce système de gestion inclut les modèles de description et de manipulation des comportements. Nous implémentons notre proposition sur notre plateforme d'interaction sociale qui permet de coordonner les comportements des agents distribués. Ces derniers collaborent alors plus efficacement.

Mots-clés : *Modèle relationnel de comportement, Environnement physique, Espace de coordination*

Abstract

Within physical environment, the heterogeneity of the components often limits the interaction between agents. Here we aim to design an architecture that furthers the exchange between agents. We propose a model to manage their behaviours. The model is based on the relational model used for database. Such a model includes the description and the manipulation of behaviours. We implement our model with our framework that enables social interaction. The framework provides support to coordinate behaviours from distributed agents. In this way agents collaborate efficiently.

Keywords: *Relational behaviour model, Real-world system, Space of coordination*

1 Introduction

Les caractères dynamique et ouvert de l'environnement réel rendent difficile la conception d'une architecture logiciel adéquate [3]. En effet, dans l'environnement physique, des évènements imprévisibles peuvent perturber le bon fonctionnement d'un système informatique. Par exemple, l'insuffisance de ressources matérielles suite à une avarie, ou l'interruption des flux de communication. Une autre caractéristique des systèmes opérant dans un environnement physique est l'hétérogénéité des composants qui complexifie les inter-liaisons au point de limiter leurs capacités d'interagir entre eux. Cela implique qu'à tout moment, et pour une durée plus ou moins importante, tout ce qui est produit par un composant (données ou services) reste inaccessible et inutilisable par les autres. Nous proposons ici une formalisation des comportements des agents pour leur permettre de coordonner leurs comportements respectifs. L'objectif est de favoriser leurs interactions et gagner ainsi en efficacité.

2 Modélisation des comportements

Notre approche dite pluri-comportementale consiste à définir plusieurs comportements pour un agent et de n'activer que les comportements opportuns. En analysant les comportements nous avons identifié des similarités avec les données dans une base de données [1]. Chaque comportement possède différentes propriétés comme une donnée ayant quelques attributs. En incluant l'agent qui le porte, chaque comportement est distinct des uns des autres. Chaque comportement peut être en relation avec d'autres. Les comportements peuvent donc former un ensemble tout comme les données. Nous nous inspirons donc du modèle relationnel de données pour représenter l'ensemble des comportements existants. Cela permet de faciliter l'analyse des compétences disponibles, tout en contextualisant leurs possibilités de mobilisation.

2.1 Modèle de description

Le modèle relationnel permet de représenter les comportements avec des concepts simples : attributs, clé, schéma relationnel. Un exemple de schéma relationnel est représenté ci-après. Le

tuple *Comportement* est alors défini comme une instance valuée de ce schéma relationnel.

COMPORTEMENT : (***#Alias*** : ChaîneDeCaractère, ***#Skill*** : ChaîneDeCaractère, ***Activity*** : Booléen, ***Duration*** : UnitéDeTemps, ***Frequency*** :UnitéDeTemps , ***Range*** : ChaîneDeCaractère, ***Precondition*** : ChaîneDeCaractère, ***Post-condition*** : ChaîneDeCaractère).

2.2 Le langage relationnel

Le modèle relationnel permet la manipulation des comportements avec des opérations simples comme la sélection (σ), et la projection (π) qui constituent un langage de requête. La sélection (σ) est pour la recherche d'un comportement particulier. La projection (π) est pour la sélection des attributs d'un ensemble de tuples. Cette opération est principalement utilisée pour identifier par exemple les agents ou les comportements actifs. Nous utilisons les opérateurs ensemblistes, l'union ($\cup$) et la différence ($/$), pour faire des combinaisons.

3 Implémentation

3.1 Architecture

Nous utilisons une architecture orientée comportement [2]. Notre architecture s'appuie sur un framework appelé *Ubiquity*. C'est une librairie Java qui permet de générer des espaces d'interaction sociale pour un ensemble de sur des machines connectées en réseau. Nous utilisons cet espace d'interaction comme support de coordination des comportements. Un espace de coordination correspond à une thématique particulière, par exemple : la surveillance, la sécurité,... Les agents déclarent chacun de leurs comportements sous forme de tuple dans les espaces correspondants. Ils y envoient également des requêtes pour solliciter un comportement auprès d'autres agents. La Figure 1 suivante illustre l'architecture pour la coordination des comportements.

3.2 Exemple de cas d'utilisation

Un exemple d'utilisation de notre modèle est l'animation d'une flotte de 3 robots dont la mission est d'extraire des débris métalliques enfouis sous la terre. Chacun de robots est représenté sous forme d'un agent. Une fois connecté dans l'espace de coordination via *Ubiquity*, il déclare ses comportements sous forme de tuple. Ensuite en fonction de ses besoins il envoie

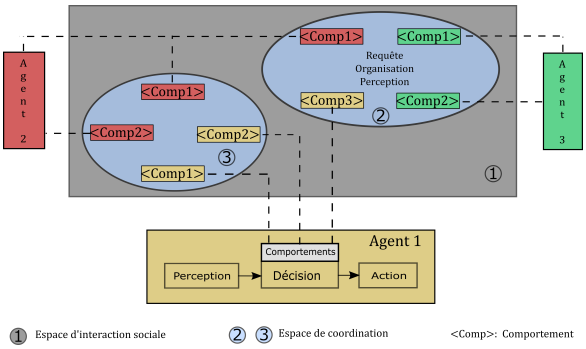


FIGURE 1 – Espace de coordination de comportements

des requêtes pour trouver un comportement. Les syntaxes utilisées reprennent celles du langage SQL (Insert, Select, Update,...). Une fois que l'agent obtient le résultat de ses requêtes, il négocie directement avec l'agent qui propose le comportement.

4 Conclusion

Le modèle relationnel permet de représenter et d'analyser de façon flexible l'ensemble des comportements pour des agents distribués sur un réseau de machines. En utilisant la représentation et les opérateurs fournis par le modèle relationnel, nous apportons une solution pour coordonner les comportements pour les systèmes destinés à opérer dans un environnement physique. Ce modèle favorise l'interaction entre les agents et surtout la mise en disposition des comportements, donc le partage de services entre agent.

Remerciement

Ce projet a reçu le soutien financier de la Région Réunion et de L'Union Européenne.

Références

[1] E. F. Codd. *The Relational Model for Database Management, Version 2*. Addison-Wesley, 1990.

[2] Paolo Pirjanian. Behavior coordination mechanisms-state-of-the-art. Technical report, Institute for Robotics and Intelligent Systems, University of Southern California, 1999.

[3] Franco Zambonelli, Andrea Omicini, Bernhard Anzengruber, Gabriella Castelli, Francesco L De Angelis, Giovanna Di Marzo Serugendo, Simon Dobson, Jose Luis Fernandez-Marquez, Alois Ferscha, Marco Mamei, et al. Developing pervasive multi-agent systems with nature-inspired coordination. *Pervasive and Mobile Computing*, 17 :236–252, 2015.

Démonstrations

BESSICA : un outil permettant de tester le comportement d'agents cognitifs

J-P. Barthès
barthes@utc.fr

CNRS UMR 7253 Heudiasyc, Sorbonne Universités,
Université de Technologie de Compiègne, France

Résumé

BESSICA est un outil prototype permettant de développer des simulations d'activités collaboratives de groupes mixtes d'agents artificiels et d'humains. Il n'y a pas de système à notre connaissance permettant de prendre en compte simultanément et facilement des paramètres physiques, psychologiques et sociaux. Le principe de le prise de décision pour un agent repose sur un calcul de priorité affecté à chaque tâche. Comme dans toute approche d'IA, BESSICA fournit des mécanismes par défaut.
Les humains sont pris en compte à travers des agents assistants personnels avec lesquels ils peuvent communiquer en langage libre.

Mots-clés : *Simulation d'équipes mixtes agents-humains*

Abstract

BESSICA, a tool for simulating the work of mixed teams of artificial agents and humans from rules combining the parameters according to the chosen model of behavior. It allows taking into account physical, psychological and social parameters. Humans are integrated via personal assistant agents with which they can conduct dialogs using natural language.
BESSICA was designed during a thesis about using trust for reaching collective decisions. It currently uses the ANR project VICTEAMS for further developments.

Keywords: *Simulation of mixed teams of artificial agents and humans*

1 Introduction

On rencontre de plus en plus de situations où l'on souhaite prendre en compte des interactions entre agents artificiels et humains lors de simulations. Pour cela il est nécessaire de doter les agents de capacités de dialogue et de comportements qui nous paraissent plausibles. Les sciences cognitives fournissent un grand nombre de modèles de comportement (humains) en essayant de caractériser les états mentaux et de déterminer les paramètres intervenant dans ces états et gouvernant les transitions entre ceux-ci jusqu'à la prise de décision. Depuis SOAR [4] jusqu'à BDI/SID [3] en passant par ACT-R ou PRS de nombreuses propositions de plateformes ont été faites y compris pour des domaines ou applications particulières [1, 2]. Toutefois la mise en œuvre reste souvent difficile.

BESSICA n'est pas un n-ième modèle de comportement d'agents cognitifs, mais un outil permettant de tester des modèles tenant compte de paramètres physiques, psychologiques et sociaux en déchargeant l'utilisateur des aspects de gestion de la simulation.

2 Principe de fonctionnement

BESSICA repose sur un système multi-agents où chaque agent est multi-tâches. Chaque agent est indépendant et décide à tout moment de la tâche à exécuter en fonction de la valeur d'une *priorité* associée à cette tâche. Chaque agent a une représentation du monde tel qu'il le perçoit, une représentation des autres agents et une représentation de buts qui se décomposent en tâches qu'il doit effectuer. Les priorités sont recalculées en permanence à chaque pas de la simulation. Une tâche peut être interrompue si elle est interruptible, puis reprise plus tard. Le mécanisme de comportement est implanté sous forme de règles combinant les valeurs des différents paramètres se trouvant dans les différentes représentations. Un agent particulier représente l'état objectif du monde et règle les problèmes de tâches nécessitant une coopération.

Les humains sont interfacés à l'aide d'agents assistants personnels avec lesquels ils communiquent en langage naturel.

Le fonctionnement de tous les agents est asynchrone.

3 Fonctionnalités

BESSICA comporte un certain nombre de fonctionnalités :

tâches : gestion de tâches simples, ancillaires, complexes (AND, OR, SEQ), collaboratives (nécessitant plusieurs acteurs), simultanées (pour un même agent) ;

règles de combinaison permettant de modifier les valeurs de priorités des tâches ;

initialisation des paramètres et **affichage** des résultats ;

mécanismes par défaut : jeu de règles et mécanisme de combinaison des priorités ;

gestion des dialogues en langue naturelle ;

connexions à d'autres environnements (grâce à des agents de transfert).

4 Démonstration

Aménagement d'un labo : un exemple proposé par Lucile Callebert mettant en scène trois agents, Lucile, Margaux et Youyou, ayant à aménager un labo : nettoyer la pièce, monter un bureau, installer un ordinateur et acheter une cafetière. Interviennent les pauses café, déjeuner, le téléphone, les relations entre les agents.

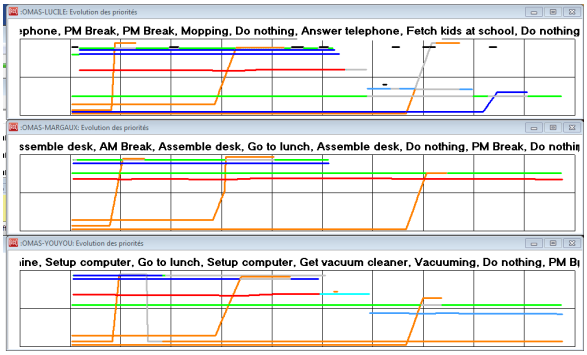


FIGURE 1 – Évolution des priorités des différentes tâches

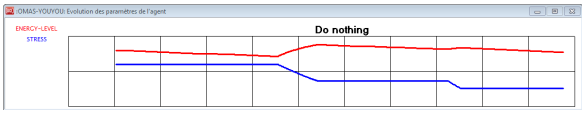


FIGURE 2 – Évolution du stress et de l'énergie de l'agent Youyou

On voit sur la Fig. 3 l'avancement de sous-tâches et l'action simultanée des agents Lucile et Youyou sur la deuxième sous-tâche.

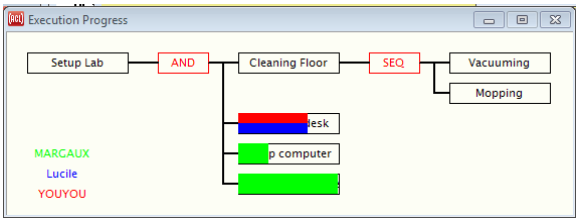


FIGURE 3 – Fenêtre montrant l'évolutions des tâches collectives

VICTEAMS : projet ANR impliquant un médecin leader (humain) des infirmiers et aides soignants dans un poste avancé de secours recueillant des blessés.

Remerciements

Ce travail a bénéficié en partie du projet VICTEAMS (ANR-14-CE24-0027, financé par l'ANR et par à la fois la Direction Générale de l'Armement (DGA) et le Labex MS2T à travers le programme "Investissements pour l'Avenir" géré par l'ANR (Reference ANR-11-IDEX-0004-02).

Je remercie en particulier Domitile Lourdeaux, Lucile Callebert, Lauriane Huguet, et Rémi Lacaze-Labadie pour les nombreuses discussions fructueuses que nous avons eues et pour leur aide sur les éléments pratiques concernant les applications.

Références

[1] N. Bécu, P. Bommel, C. Le Page, F. Bousquet, Cormas, une plate-forme multi-agent pour concevoir collectivement des modèles et interagir avec les simulations, *Systèmes Multi-Agents et simulation - Vingt-quatrièmes Journées francophones sur les systèmes multi-agents*, pp 97-106,, 2016.

[2] P. Chevaillier, T-H. Trinh, M. Barange, P. De Loor, F. Devillers, J. Soler, R. Querrec, Semantic modeling of Virtual Environments using MASCARET, *5th Workshop on Software Engineering and Architectures for Realtime Interactive Systems*, pp. 1-8, 2012.

[3] S. Mascarenhas, N. Degens, A. Paiva, R. Prada, G.J. Hofstede, A. Beulens, R. Aylett, Modelling culture in intelligent virtual agents, *Autonomous Agent and Multi-Agent Systems*, vol 30, pp 931-962, 2016.

[4] P. S. Rosenbloom, J. Laird, *The Soar papers : research on integrated intelligence,* MIT Press, 1993.

VICTEAMS : une équipe de personnages virtuels autonomes pour la formation au sauvetage de blessés

L. Huguet[a,b] lauriane.huguet@hds.utc.fr
D. Lourdeaux[a] domitile.lourdeaux@hds.utc.fr
N. Sabouret[b] nicolas.sabouret@limsi.fr

[a]Sorbonne universités, Université de technologie de Compiègne, CNRS, Heudiasyc UMR 7253
60203 Compiègne CEDEX, France

[b]LIMSI, CNRS, Univ. Paris-Sud, Université Paris Saclay, Bat 508, Campus Universitaire, 91405 Orsay, France

Résumé

Le projet VICTEAMS [1] a pour objectif de concevoir un environnement virtuel pour la formation de leader d'équipes médicales, centré sur les compétences non-techniques. L'équipe médicale est constituée d'agents autonomes capables de produire des comportements erronés. Nous décrivons ici les principaux éléments du système multi-agent que nous proposons de présenter lors d'une démonstration à JFSMA 2017.

Mots-clés : *SMA et agents virtuels*

Abstract

We propose a demonstration of the VICTEAMS platform[1], a virtual environment for training leaders of medical teams. The medical team is played by a multi-agent system in which autonomous agents can adopt erroneous behaviours.

Keywords: *MAS and virtual agents*

1 Introduction

Le projet ANR VICTEAMS[1] vise à développer un environnement virtuel pour entraîner un leader médical à gérer une situation de crise. Au delà de la formation aux compétences médicales (ou « techniques »), il est important de développer les compétences non-techniques comme le travail en équipe, le leadership, la communication ou encore la gestion du stress [3].

Dans cet objectif, nous utilisons des agents adaptatifs, capables de générer des comportements adaptés ou non, représentatifs de comportements humains. Notre modèle est implémenté dans la plate-forme logicielle multi-agent HUMANS, couplée avec un moteur d'animation réalisé par la société Reviatech [2]. Les principaux composants sont : l'agent SELDON, qui est responsable du moteur de scénarisation des situations d'apprentissages ; l'agent REPLICANTS qui est lui-même un système multi-agent dont les agents, appelés PVA, correspondent aux membres de l'équipe médicale (ils sont incarnés par des avatars dans l'environnement virtuel) ; l'agent WORLD-MANAGER qui est responsable de la gestion sémantique du monde en lien avec l'environnement virtuel ; et l'agent COMMUNICATOR qui est responsable de la centralisation et de la répartition des communications entre les différents modules.

1. https://victeams.hds.utc.fr/
2. http ://www.reviatech.com/

2 Le SMA REPLICANTS

Le SMA REPLICANTS contrôle le comportement des PVA dans l'environnement virtuel. Dans une majorité des systèmes existants utilisant des agents autonomes, l'objectif principal est d'obtenir une équipe performante, que ce soit en limitant l'autonomie des PVA via une gestion décentralisée du collectif (*e.g.* S-MOISE) ; ou par l'entretient de plan partagés (SharedPlans) entre les agents (*e.g.* STEAM). Au contraire, notre objectif est de générer une équipe virtuelle avec des comportements représentatifs de comportements humains en situation de crise et, de ce fait, qui peuvent être plus ou moins adaptés et qui peuvent conduire à des erreurs de l'équipe.

Pour atteindre cet objectif, une analyse terrain menée par des ergonomes du LIMSI et de l'IRBA a conduit à la conception du modèles de tâches de l'équipe qui a été ensuite traduit dans le langage de description ACTIVITY-DL [1], inspiré de langages ergonomiques (MAD, HTN). Les PVA disposent ainsi, via le modèle de tâches, d'une connaissance commune de l'activité du groupe. Cependant, ils vont l'interpréter différemment en fonction de leurs croyances et nous ne cherchons pas à réduire les différences d'interprétation entre les PVA. Ce sera à l'utilisateur (le leader de l'équipe virtuelle), de gérer les déviations en interagissant avec son équipe.

Modèle décisionnel du PVA. Chaque PVA décide de l'action à entreprendre en fonction de ses croyances sur le contexte (actions déjà effectuées, état du monde...), du modèle de tâches décrit en ACTIVITY-DL et de ses variables internes. Il peut décider de communiquer pour informer ou demander une information nécessaire à sa prise de décision. L'une des originalités de notre modèle décisionnel est qu'en fonction de ses caractéristiques personnelles et de ses croyances locales, le PVA interprète différemment le modèle de tâche.

Les croyances locales sont obtenues par observation de l'environnement et par des interactions entre les PVA, qui traduisent des communications entre les membres de l'équipe médicales. Ces communications peuvent être erronées (communication incomplète ou mécompréhension) ou s'appuyer sur des croyances locales qui ne sont plus à jour. Ainsi, les PVA peuvent maintenir des croyances erronées et prendre de mauvaises décisions, que le joueur doit détecter et corriger.

Les comportements collectifs émergent des actions des PVA. Ces comportements variés doivent rendre compte de la complexité des situations rencontrées sur le terrain. Pour cela, nous avons défini avec les experts du domaine un ensemble de caractéristiques des PVA. En particulier, nous avons identifié l'importance du niveau de qualification, qui détermine quelles tâches le PVA peut réaliser, et le style de communiquant, qui définit la propension du PVA à communiquer avec le reste de l'équipe. Nous illustrons ces deux caractéristiques dans notre démonstration.

Nous avons aussi identifié : le style de followership (le rapport du PVA au leader [2]) ; le style de mindfulness (résistance au stress du PVA) ; et un taux d'erreur lié au stress, à la fatigue, à la charge cognitive, etc.. Ceux-ci impactent directement le choix d'action et le comportement de communication.

3 Environnement virtuel

L'environnement virtuel dans lequel est immergé l'apprenant représente un poste médical avancé, c'est-à-dire la zone dans laquelle sont amenés les blessés lors de leur extraction de la zone de crise. La figure 1 montre un premier visuel de l'intérieur de ce poste médical.

L'apprenant dispose de menu d'actions et de communication pour interagir avec son équipe virtuelle et effectuer certaines actions lui-même.

FIGURE 1 – Aperçu de l'environnement virtuel

4 Conclusion

L'objectif du projet VICTEAMS n'est pas de former à l'expertise médicale, bien que les compétences requises pour utiliser le simulateur sont de cette nature, mais bien de contribuer à la constitution d'équipes expertes et adaptables, dans laquelle les membres de l'équipe médicale sont conscients de l'importance de la coordination et sont capables de faire face à des situations non anticipées.

5 Remerciements

Les auteurs remercient la Région Hauts-de-France et le FEDER 2014/2020 pour le cofinancement de ce travail. Celui-ci a été réalisé dans le cadre du projet VICTEAMS (ANR-14-CE24-0027), financé par l'ANR et la DGA. Il est labellisé par le Labex MS2T.

Références

[1] Camille Barot. *Scénarisation d'environnements virtuels.* PhD thesis, Université de Technologie de Compiègne, 2014.

[2] Guillaume Demary, Virginie Demulier, and Jean-Claude Martin. Leader/suiveur : quel impact du stress et des différences interindividuelles sur le processus de catégorisation du leader, lors d'une situation de crise. In *Journée Scientifique des Jeunes Chercheurs en Psychologie*, 2015.

[3] Rhona Flin, Lynne Martin, Klaus-Martin Goeters, HJ Hormann, René Amalberti, Claude Valot, and Herman Nijhuis. Development of the notechs (non-technical skills) system for assessing pilots' crm skills. *Human Factors and Aerospace Safety*, 3 :97–120, 2003.