Actes des 27es Journées Francophones sur les Systèmes Multi-Agents

Olivier Simonin

AfIA

Association française
pour l'Intelligence Artificielle

JFSMA

Journées Francophones sur les Systèmes Multi-Agents
— Articles acceptés —

PFIA 2019

Table des matières

Éditorial

Nourri de pluridisciplinarité, le paradigme multi-agent fournit un cadre conceptuel pour l'étude et la conception de systèmes dont la dynamique globale est le fruit d'entités autonomes – les agents – qui interagissent dans un environnement commun. Depuis 1993, les Journées Francophones sur les Systèmes Multi-Agents (JFSMA) réunissent chaque année, trois jours durant, des chercheurs qui étudient, utilisent et font évoluer ce paradigme pour adresser des problématiques issues de domaines liés à l'informatique (intelligence et vie artificielles, génie logiciel, robotique collective, etc.) et aux sciences humaines et naturelles (économie, sociologie, éthologie, etc.). Les JFSMA constituent ainsi un rendez-vous scientifique privilégié, placé sous le signe de l'échange et de l'ouverture. Cette année les JFSMA 2019 se tiennent à Toulouse, du 3 au 5 juillet 2019, dans le cadre de la Plate-Forme de l'Intelligence Artificielle (PFIA 2019).

Ce document contient l'ensemble des articles sélectionnés pour la vingt-septième édition des Journées Francophones sur les Systèmes Multi-Agents (JFSMA) dans leur version pré-édition. Les versions finales sont présentées dans les Actes JFSMA 2019 publiés aux éditions Cepadues.

Par tradition, chaque édition des JFSMA met en exergue une thématique spécifique que les auteurs sont invités à prendre en compte dans leurs contributions. Cette année, le thème des journées est « *systèmes distribués, embarqués et diffus* ». Compte tenu du développement actuel de la robotique mobile, des véhicules autonomes connectés, des robots ou assistants connectés aux objets, dans l'habitat ou plus généralement dans l'espace urbain, il apparaît que l'interaction entre ces objets et ces robots, dans un but de coopération et d'assistance à l'homme, reste un défi majeur de l'Intelligence Artificielle Distribuée et plus spécifiquement pour la communauté Systèmes Multi-Agents.

Cette année nous avons reçu 28 soumissions parmi lesquelles le comité de programme a sélectionné 10 présentations longues, 6 présentations courtes, et 3 démonstrations, pour nous proposer un programme scientifique riche et varié. Les verrous adressés dans plusieurs de ces articles sont ceux de l'optimisation, de la résilience et de la tolérance aux pannes. Les modèles et méthodes qui sont proposés, plaçant au centre les notions de distribution et de décentralisation des décisions des agents, abordent de manière renouvelée des techniques de résolution collective de problèmes, d'auto-organisation et de simulation individu-centrée.

Cette année est également marquée par la diversification des champs d'applications, mentionnons l'internet des objets, les flottes robotiques, la consommation d'énergie, les systèmes routiers, ou le domaine médical. La présence de démonstrations lors de ces journées témoignent également de la maturité technologique des contributions francophones.

Olivier SIMONIN

Comité de programme

Présidents

— Olivier Simonin, INSA Lyon, CITI lab. & Inria

Membres

— Flavien Balbo ISCOD / Henri Fayol Institute, MINES Saint-Etienne
— Valérie Camps University of Toulouse - IRIT, France
— Olivier Boissier Mines Saint-Etienne, Institut Henri Fayol, Laboratoire Hubert Curien UMR CNRS 5516
— Zahia Guessoum LIP6, Université de Paris 6 and CReSTIC, Université de Reims Champagne Ardenne
— Guillaume Hutzler Evry University,
— Aurélie Beynier University of Pierre and Marie Curie (Paris 6), LIP6
— Maxime Morge Université de Lille
— Yves Demazeau CNRS - LIG
— Philippe Mathieu University of Lille 1
— Stéphane Galland UBFC - UTBM
— Bruno Mermet GREYC-CNRS - Université du Havre
— Gildas Morvan LGI2A, Université d'Artois
— Nicolas Marilleau UMI UMMISCO - Institut de recherche pour le développement (IRD)
— Rene Mandiau LAMIH, Université de Valenciennes
— Julien Saunier LITIS, INSA-Rouen
— Michel Occello Université Grenoble Alpes - LCIS
— Frederic Migeon IRIT
— Emmanuelle Grislin LAMIH - Universite de Valenciennes
— Nicolas Sabouret LIMSI-CNRS
— Salima Hassas, LIRIS, université de Lyon
— Lilia Rejeb SMART LAB, ISG Tunis, Université de Tunis
— Vincent Chevrier Université de Lorraine, LORIA
— Elsy Kaddoum IRIT Université Toulouse III
— François Charpillet LORIA, Inria, Nancy
— Suzanne Pinson Université Paris-Dauphine
— Grégory Bonnet Université de Caen Normandie
— Jean-Pierre Muller CIRAD
— Jean-Paul Jamont LCIS, Université de Grenoble
— Mahdi Zargayouna IFSTTAR Marne-la-Vallée
— Denis Payet Université de la Réunion
— Christophe Lang FEMTO-ST Besançon
— Emmanuel Adam LAMIH, Université de Valenciennes

Stratégie situationnelle pour l'équilibrage de charge

Quentin Baert[a] quentin.baert@univ-lille.fr
Anne-Cécile Caron[a] anne-cecile.caron@univ-lille.fr
Maxime Morge[a] maxime.morge@univ-lille.fr
Jean-Christophe Routier[a] jean-christophe.routier@univ-lille.fr
Kostas Stathis[b] kostas.stathis@rhul.ac.uk

[a]Centre de Recherche en Informatique, Signal et Automatique de Lille,
Université de Lille, France

[b]Department of Computer Science,
Royal Holloway, University of London, Egham, United Kingdom

Résumé

Nous étudions une stratégie qui tient compte de la localité des ressources pour équilibrer les charges dans un système distribué. Cette stratégie permet aux agents coopératifs d'identifier une allocation non équilibrée, voire de déclencher des enchères concurrentes pour réallouer localement certaines des tâches. Les tâches sont réallouées en tenant compte de l'accessibilité des ressources pour les agents ; elles sont exécutées conformément aux capacités des nœuds de calcul sur lesquels se trouvent les agents. Ce processus de négociation dynamique et continu est concurrent à l'exécution des tâches, ce qui permet d'adapter l'allocation des tâches aux perturbations (exécution de tâche, chute de performance d'un nœud). Nous évaluons cette stratégie dans le cadre du déploiement multi-agents de MapReduce. Ce patron de conception permet le traitement distribué de données massives. Les résultats empiriques démontrent que notre stratégie améliore significativement le temps d'exécution du traitement d'un jeu de données.

Mots-clés : *Résolution collective de problème, Négociation, Modèles de comportement agent*

Abstract

We study a novel location-aware strategy for distributed systems where cooperating agents perform the load-balancing. The strategy allows agents to identify opportunities within a current unbalanced allocation, which in turn triggers concurrent and one-to-many negotiations amongst agents to locally reallocate some tasks. The tasks are reallocated according to the proximity of the resources and they are performed in accordance with the capabilities of the nodes in which agents are situated. This dynamic and ongoing negotiation process takes place concurrently with the task execution and so the task allocation process is adaptive to disruptions (task consumption, slowing down nodes). We evaluate the strategy in a multi-agent deployment of the MapReduce design pattern for processing large datasets. Empirical results demonstrate that our strategy significantly improves the overall runtime of the data processing.

Keywords: *Distributed cooperative problem solving, Negotiation, Agent behaviour*

1 Introduction

Le problème d'équilibrage des charges et d'allocation des tâches dans un système distribué apparait dans de nombreuses applications telles que l'informatique en nuage, les réseaux pair-à-pair, les réseaux sociaux et le traitement de données massives. Dans cet article, nous abordons les problèmes applicatifs où (a) certaines ressources, par exemple des données, nécessaires à l'exécution d'une tâche sont distribuées sur différents nœuds du réseau, et (b) certains des nœuds sont susceptibles de rencontrer des perturbations lors de l'exécution, par exemple une chute de performance ou des latences réseau. Comme plusieurs ressources sont requises pour exécuter une tâche, toute allocation implique inévitablement le transfert de ces ressources entre les nœuds de calculs, ce qui induit un délai supplémentaire lors du traitement de la tâche [15]. Dans cette classe de problèmes, l'allocation de tâches peut être remise en cause lors de l'exécution des tâches et tirer parti de la distribution des ressources dans le système.

Afin d'aborder le problème d'équilibrage des charges et d'allocation des tâches dans des applications telles que celles qui motivent ce travail, les technologies multi-agents ont fait l'objet d'une grande attention, particulièrement celles qui portent sur l'ordonnancement à l'aide d'enchères [27, 24, 26]. En plus de la décentralisation qui permet d'éviter les goulots d'étranglement en terme de performance, nous montrons ici qu'une approche multi-agents pour l'allocation située de tâches répond à deux autres exigences cruciales : (a) la co-occurence de la ré-

allocation des tâches et de leur exécution ; (b) l'adaptation de l'allocation, c'est-à-dire le déclenchement d'une réallocation quand une perturbation se produit. Nous supposons que les agents sont coopératifs. Ils partagent le même objectif : diminuer le temps d'exécution global des tâches. Nous supposons également que les agents ne partagent aucune connaissance, y compris l'état de l'allocation courante. Néanmoins, les agents possèdent un modèle de leurs pairs : ils sont ainsi capables de calculer le coût des tâches pour leurs pairs. Nous supposons également qu'une tâche peut être exécutée par n'importe quel agent seul, et ce sans préemption ni contrainte de précédence. Une tâche est indivisible, sans date butoir et non partageable, c'est-à-dire qu'une tâche n'appartient qu'à un agent à la fois.

Nous formalisons ici le problème d'allocation de tâches situées. En adoptant une approche fondée sur le marché, nous décentralisons totalement le processus d'équilibre des charges grâce auquel les agents coopératifs essaient de minimiser le temps d'exécution de la dernière tâche, i.e. le *makespan*. À cette fin, nous proposons une stratégie situationnelle afin de procéder à l'équilibrage des charges [1]. Quand les agents identifient des opportunités pour rééquilibrer l'allocation courante, ils initient des enchères de manière concurrente pour réallouer localement certaines tâches. Ces dernières sont réallouées en considérant la proximité entre les ressources nécessaires à leur exécution et le nœud sur lequel elles sont effectivement exécutées. Ce processus de réallocation de tâches est dynamique et continu. Il est concurrent à l'exécution des tâches, ce qui rend le système distribué adaptatif aux événements perturbateurs (consommation de tâche, chute de performance d'un nœud).

Notre application pratique est la distribution de MapReduce. Ce patron de conception, dont Hadoop est une implémentation, permet de traiter de larges volumes de données sur des grappes de calculs [9]. Plusieurs biais apparaissent sur des jeux de données réels et mènent à un déséquilibre des charges [17]. Un ordonnanceur centralisé ne peut pas être utilisé comme point de référence de par le grand nombre de tâches à traiter. Cependant, un processus de négociation multi-agents permet de réallouer les tâches lors de la phase de *reduce* pour réduire le temps d'exécution du *job*. Nos résultats empiriques préliminaires montrent que la stratégie situationnelle permet d'améliorer significativement le temps de traitement d'un jeu de données.

Cet article est structuré comme suit. Après une présentation des travaux connexes en section 2, nous formalisons le problème d'allocation de tâches situées à l'aide d'un système multi-agents en section 3. Cette section définit également la délégation socialement rationnelle de tâches, ce qui permet aux agents d'améliorer localement une allocation de tâches. La section 4 illustre le processus de négociations itérées qui a lieu en même temps que l'exécution des tâches. La section 5 décrit la stratégie situationnelle, c'est-à-dire comment les agents choisissent quelle tâche exécuter/négocier. Notre application pratique et notre validation empirique sont décrites dans la section 6. Enfin, la section 7 résume notre contribution et présente nos perspectives.

2 Travaux connexes

Les problèmes d'ordonnancement classiques ont fait l'objet d'un grand nombre d'études et de recherches. Ces dernières ont abouti à des ordonnanceurs hors-ligne pour des modèles simples [6]. Le problème de la minimisation du *makespan* (le temps auquel la dernière tâche est achevée) pour n tâches sur m machines hétérogènes (avec des capacités différentes), appelé $R||C_{max}$, est NP-difficile [13]. Les algorithmes pseudo-polynomiaux conçus pour ce problème incluent : l'heuristique *earliest completion time* [14] (ECT), les heuristiques de recherche locale [12], l'algorithme *branch and bound* [20] et les heuristiques biphasées basées sur la programmation linéaire [18]. Même si ECT est un algorithme d'approximation qui atteint des résultats acceptables avec un coût computationnel faible, les algorithmes centralisés ne peuvent pas être appliqués à notre scénario où les tâches sont nombreuses (par exemple $82,283$ clés dans la section 6). La stratégie situationnelle est une heuristique de recherche locale décentralisée.

L'ordonnancement multi-agents [15] a suscité beaucoup d'intérêt dans le cadre du problème d'équilibrage de charge pour les systèmes distribués. Ce problème est différent des problèmes d'ordonnancement classiques de par ses exigences :

— **passage à l'échelle.** Un contrôleur global constitue un goulot d'étranglement en terme de performance car il doit collecter les informations d'état de l'ensemble du système en temps réel. Au lieu de

1. Cet article est une version révisée, étendue et traduite de [5] où, dans la continuité de [4], la localité des ressources est prise en compte.

cela, la répartition des tâches peut être négociée par des agents représentants les nœuds [27, 1] ;
— **réactivité.** Les problèmes d'ordonnancement classiques sont statiques. L'estimation inexacte du temps d'exécution des tâches, aggravée par des perturbations (consommation de tâches, ralentissement des nœuds, etc.), peuvent nécessiter d'importantes modifications de l'allocation existante pour qu'elle reste optimale [26]. Plutôt que de recalculer en continu une allocation optimale, quelques modifications locales au cours de l'exécution des tâches peuvent améliorer l'équilibre de charge [3, 28, 16].

La plupart des travaux existants adoptent une approche fondée sur le marché [27, 11, 21] qui modélise l'équilibrage de charge comme un jeu non coopératif afin d'optimiser des métriques centrées sur l'utilisateur plutôt que des métriques centrées sur le système, telles que le temps d'achèvement global que nous utilisons dans cet article.

Contrairement à notre travail, [24] assignent les tâches à des coalitions d'agents car ils ne peuvent pas réaliser les tâches seuls. Dans cet article, nous supposons qu'une tâche, qui peut être exécutée par n'importe quel agent seul sans préemption ni contrainte de précédence, est indivisible, non partageable (i.e. une tâche n'appartient qu'à un seul agent à la fois) et sans date butoir. [8] ont étudié l'allocation multi-agents de ressources mais leur travail se limite à l'affectation d'une seule ressource par agent. Nous supposons ici qu'un lot de tâches peut être assigné à chaque agent. Dans la continuité de [2, 10], [22] proposent des méthodes potentiellement distribuables pour l'allocation de ressources qui s'appuient sur la négociation multi-agents. Même si la topologie du réseau n'entre pas dans le cadre de notre travail (nous partons du principe que les agents sont pleinement connectés), nous allons un peu plus loin en décentralisant effectivement la négociation et en remettant en jeu cette allocation pendant le traitement des tâches.

Il est bien connu que le traitement de jeux de données réels via le déploiement distribué du patron de conception MapReduce comporte souvent des biais de données qui peuvent conduire à un déséquilibre des charges [17]. En particulier, le biais de partitionnement se produit lorsqu'un *reducer* traite un plus grand nombre de clés que les autres. Puisque le *job* se termine lorsque toutes les tâches *reduce* sont terminées, le temps d'exécution du *job* est pénalisé par le *reducer* le plus chargé. Ce biais de données est abordé par [7, 19] à l'aide d'un paramétrage dépendant de connaissances préalables sur des données et sur l'environnent d'exécution. Nous abordons cette question à l'aide d'une réallocation de tâches dynamique et adaptative qui est concurrente à la consommation des tâches. Ainsi, la réallocation s'adapte au traitement des données. Cela nous permet d'aborder les problèmes réels suivants : (a) l'absence de connaissance préalable sur les données, (b) l'estimation inexacte du temps d'exécution des tâches, et (c) les aléas d'exécutions (chute de performance d'un nœud, latence réseau). À notre connaissance, aucune autre proposition n'est, comme la nôtre, réactive et capable de passer à l'échelle.

3 Allocation de tâches situées

Nous formalisons maintenant le problème de l'allocation multi-agents de tâches situées (MASTA). Ici les tâches ont des coûts différents selon les agents, en fonction de la localité des ressources.

Définition 1 (MASTA)**.** *Un problème d'allocation multi-agents de tâches situées de taille* (k, m, n) *avec* $k \geq 1$*,* $m \geq 2$ *et* $n \geq 1$ *est un n-uplet* $MASTA = \langle Node, \mathcal{A}, \mathcal{T}, l, d, c \rangle$ *tel que :*

— $Node = \{node_1, \ldots, node_k\}$ *est un ensemble de* k *nœuds ;*
— $\mathcal{A} = \{1, \ldots, m\}$ *est un ensemble de* m *agents ;*
— $\mathcal{T} = \{\tau_1, \ldots, \tau_n\}$ *est un ensemble de* n *tâches à traiter ;*
— $l : \mathcal{A} \mapsto Node$ *donne la localisation d'un agent ;*
— $d : \mathcal{T} \times Node \mapsto \mathbb{N}^+$ *donne le nombre de ressources d'une tâches* τ *situées sur un nœud* x *;*
— $c : \mathcal{T} \times Node \mapsto \mathbb{R}_+^*$ *donne le coût d'une tâche* τ *lorsqu'elle est exécutée sur un nœud. Plus une tâche est locale, moins elle coûte chère :*

$$\forall i, j \in \mathcal{A}, d(\tau, l(i)) > d(\tau, l(j)) \Rightarrow c_i(\tau, l(i)) \leq c(\tau, l(j)) \quad (1)$$

Dans la suite de l'article, on écrira l_i (resp. $c_i(\tau)$, $d_i(\tau)$) à la place de $l(i)$ (resp. de $c(\tau, l_i)$, $d(\tau, l_i)$). De la même façon, on écrira d_τ pour $\sum_{node \in Node} d(\tau, node)$. On dit que τ est locale (resp. semi-locale, distante) pour l'agent i si $d_i(\tau) = d_\tau$ (resp. $d_i(\tau) < d_\tau$, $d_i(\tau) = 0$).

Étant donné un problème MASTA particulier, nous évaluons d'un point de vue collectif une

allocation de tâches répondant à ce problème, en considérant le temps nécessaire pour achever la dernière tâche, i.e. le *makespan*.

Définition 2 (Allocation de tâches/Charge de travail/Makespan)**.** *Une allocation de tâches P est une partition des tâches parmi les agents, i.e un ensemble de m lots de tâches $\{P(1), \ldots, P(m)\}$ tel que :*

$$\cup_{i \in \mathcal{A}} P(i) = \mathcal{T} \quad (2)$$

$$\forall i \in \mathcal{A}, \forall j \in \mathcal{A} \setminus \{i\}, P(i) \cap P(j) = \emptyset \quad (3)$$

La charge de travail de l'agent $i \in \mathcal{A}$ pour l'allocation P est définie par :

$$w_i(P) = \sum_{\tau \in P(i)} c_i(\tau) \quad (4)$$

Le makespan de P est défini par :

$$C_{max}(P) = \max\{w_i(P) \mid i \in \mathcal{A}\} \quad (5)$$

Les agents réalisent des délégations de tâches qui sont socialement rationnelles afin d'améliorer l'allocation.

Définition 3 (Délégation socialement rationnelle)**.** *Soit P une allocation de tâches. La délégation δ de la tâche τ par l'agent i à l'agent j permet d'obtenir l'allocation $\delta(P) = \{P'(1), \ldots, P'(m)\}$ telle que :*

$$\forall k \in \mathcal{A} \setminus \{i, j\}, P'(k) = P(k) \quad (6)$$

$$P'(i) = P(i) \setminus \{\tau\} \wedge P'(j) = P(j) \cup \{\tau\} \quad (7)$$

La délégation est socialement rationnelle si et seulement si :

$$w_j(P) + c_j(\tau) < w_i(P) \quad (8)$$

Comme la délégation socialement rationnelle δ décroît strictement le *makespan* local entre deux agents, elle ne peut pas augmenter le makespan global ($C_{max}\big(\delta(P)\big) \leq C_{max}(P)$).

Nous pouvons maintenant noter $\Gamma_i(P)$ l'ensemble des délégations socialement rationnelles pour l'agent i :

$$\Gamma_i(P) = \{\tau \in P(i) \mid w_j(P) + c_j(\tau) < w_i(P)\} \quad (9)$$

Une allocation de tâches P est dite stable si aucun agent ne peut réaliser de délégation socialement rationnelle, i.e. $\forall i \in \mathcal{A}, \Gamma_i(P) = \emptyset$.

Propriété 1. *On peut toujours aboutir à une allocation stable à partir d'une allocation non stable en utilisant un nombre fini de délégations socialement rationnelles.*

Preuve 1. *Soit P une allocation de tâches et $W_P =< w_{i_1}, \ldots, w_{i_m} >$ le vecteur des charges par ordre décroissant où w_{i_r} dénote la r^{ieme} charge la plus élevée. Si P n'est pas stable, il existe une délégation socialement rationnelle δ qui mène à $P' = \delta(P)$. Formellement,*

$$\exists i, j \in \mathcal{A}, max\big(w_i(P'), w_j(P')\big) < max\big(w_i(P), w_j(P)\big) \wedge \forall k \in \mathcal{A} \setminus \{i, j\}, w_k(P) = w_k(P') \quad (10)$$

Par conséquent, $W_{P'} < W_P$ selon l'ordre lexicographique. Formellement,

$$\exists r \in [1, m]\ \forall r' < r, W_{P'}(r') = W_P(r') \wedge W_{P'}(r) < W_P(r) \quad (11)$$

Puisqu'il y a un nombre fini d'allocations et que l'ordre est strict, il y a un nombre fini de délégations socialement rationnelles.

Voici un exemple pour illustrer les différentes définitions.

Exemple 1. *Considérons le problème $MASTA^{ex} =< Node, \mathcal{A}, \mathcal{T}, l, d, c >$ de taille $(2, 2, 7)$, où $Node = \{node_1, node_2\}$, $\mathcal{A} = \{1, 2\}$ et $\mathcal{T} = \{\tau_1, \ldots, \tau_7\}$ avec $l(1) = node_1$, $l(2) = node_2$. Les localisations et coûts des tâches sont donnés dans le tableau 1. Soient les allocations Pmks et P telles que*

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
$d_1(\tau_k)$	1	0	3	6	1	6	0
$d_2(\tau_k)$	0	4	6	12	4	1	7
$c_1(\tau_k)$	1	8	15	30	9	8	14
$c_2(\tau_k)$	2	4	12	24	6	13	7

TABLE 1 – Localisations et coûts des tâches

Pmks $= \{\{\tau_1, \tau_3, \tau_5, \tau_6\}, \{\tau_2, \tau_4, \tau_7\}\}$ et $P = \{\{\tau_2, \tau_4, \tau_6\}, \{\tau_1, \tau_3, \tau_5, \tau_7\}\}$. Pmks est optimale puisque C_{max}(Pmks) $= 35$ (w_1(Pmks) $= 33$ et w_2(Pmks) $= 35$). Ce n'est pas le cas de P puisque $w_1(P) = 46$, $w_2(P) = 27$, et donc $C_{max}(P) = 46$. De plus, $\Gamma_1(P) = \{\tau_2, \tau_6\}$ et $\Gamma_2(P) = \emptyset$.

Soit $P' = \delta_1(P)$ l'allocation obtenue par la délégation δ_1 de la tâche τ_6 par l'agent 1 à l'agent 2. Comme $w_1(P') = 38$ et $w_2(P') = 40$, δ_1 améliore le makespan (i.e. $C_{max}(P') = 40$). Cependant, P' n'est pas stable car $\Gamma_2(P') = \{\tau_1\}$.

Soit $P'' = \delta_2(P')$ l'allocation obtenue via la délégation δ_2 de la tâche τ_1 par l'agent 2 à l'agent 1. Même si P'' n'est pas optimale ($C_{max}(P'') > C_{max}$(Pmks)), P'' est stable car $\Gamma_1(P'') = \Gamma_2(P'') = \emptyset$.

4 Processus de négociation

Cette section présente le processus de négociation qui est itéré et concurrent à la consommation des tâches. Quand une tâche est exécutée, elle est retirée de l'ensemble des tâches et le système multi-agents cherche à minimiser le *makespan* d'un nouveau problème MASTA. En effet, l'accomplissement d'une tâche est un évènement disruptif qui modifie les paramètres du problème et l'allocation des tâches.

Définition 4 (Consommation de tâche)**.** *Soit P l'allocation de tâches courante pour le problème $MASTA = \langle Node, \mathcal{A}, \mathcal{T}, l, d, c\rangle$. La consommation γ de la tâche τ par l'agent i aboutit à l'allocation $P' = \gamma(P)$ pour le problème $MASTA' = \langle Node, \mathcal{A}, \mathcal{T}', l, d, c\rangle$ telle que :*

$$\mathcal{T}' = \mathcal{T} \setminus \{\tau\} \tag{12}$$

$$P'(i) = P(i) \setminus \{\tau\} \tag{13}$$

$$\forall j \in \mathcal{A} \setminus \{i\}, P'(j) = P(j) \tag{14}$$

À l'évidence, la consommation d'une tâche diminue le *makespan*. La séquence des consommations, qui retirent peu à peu toutes les tâches de l'allocation initiale jusqu'à l'allocation vide $\perp$, consiste en une itération de problèmes MASTA.

Processus décentralisé de délégations de tâches. Pour déléguer des tâches, les agents réalisent des négociations concurrentes. Chaque négociation est basée sur le protocole Contract Net [25]. Il y a 3 étapes de décision durant une négociation : (a) l'initiateur de l'enchère choisit une tâche selon une des stratégies décrites dans la section 5, (b) chaque agent propose/refuse de prendre la tâche selon que la délégation est socialement rationnelle ou non, et (c) l'initiateur sélectionne le vainqueur de l'enchère, e.g. l'enchérisseur qui a la plus petite charge de travail. Plusieurs négociations impliquant différents groupes d'agents peuvent se dérouler simultanément et les agents peuvent participer à plusieurs enchères en même temps. La concurrence des délégations de tâches améliore la réactivité de la réallocation des tâches.

Comme il n'y a pas de partage de connaissances, un agent a des croyances partielles éventuellement erronées concernant l'allocation courante P. En effet, l'agent i connaît sa charge de travail $w_i(P)$ et dispose de sa propre base de croyances :

$$\mathcal{B}_i(P) = \langle w_1^i(P), \ldots, w_{i-1}^i(P), w_{i+1}^i(P), \ldots, w_m^i(P)\rangle \tag{15}$$

où $w_j^i(P)$ est ce que l'agent i croit connaître de la charge de travail de l'agent j dans l'allocation P.

L'ensemble des délégations qui sont potentiellement socialement rationnelles $\Gamma_i^{\mathcal{B}}(P)$ et que l'agent i peut initier à partir de l'allocation P s'appuie sur sa base de croyances $\mathcal{B}_i(P)$. Formellement,

$$\Gamma_i^{\mathcal{B}}(P) = \{\tau \in P(i) \mid w_j^i(P) + c_j(\tau) < w_i(P)\}. \tag{16}$$

Lorsqu'un agent souhaite initier une enchère, le calcul du *makespan* local s'appuie sur sa base de croyances, éventuellement erronée. C'est le prix à payer pour la décentralisation. Cependant, un agent informe ses pairs de sa charge de travail au début du traitement (i.e. lorsqu'il se fait connaître de ses pairs), lorsqu'il envoie des messages en tant qu'initiateur ou enchérisseur durant la négociation, et à chaque fois qu'il consomme une tâche. Ainsi la base de croyances d'un agent est mise à jour quand il reçoit un appel d'offre, et l'agent refusera une délégation de tâche qui n'est pas socialement rationnelle. Une négociation réussie aboutit nécessairement à une délégation de tâche socialement rationnelle qui ne peut pas détériorer le *makespan*. En adoptant une approche conservatrice [23], nous nous assurons qu'une négociation couronnée de succès améliore strictement l'allocation de tâches et donc que le processus de négociation converge.

Consommations et délégations concurrentes. Les délégations et consommations de tâches sont concurrentes et complémentaires, puisque la disparition d'une tâche peut permettre de nouvelles délégations socialement rationnelles. La figure 1 représente leur impact sur l'allocation, jusqu'à ce que toutes les tâches soient exécutées. À partir de l'allocation initiale P_0, les agents réalisent des délégations socialement rationnelles pour améliorer le *makespan* (e.g. le chemin de P_0 à P_k) jusqu'à la consommation d'une tâche (e.g. l'arc de P_k à P'_0), ce qui interrompt un chemin vers une allocation stable (e.g. le chemin en gris de P_k à P_{stable}). Une consommation de tâche peut aussi se produire après avoir atteint une allocation stable (e.g. après P'_{stable}).

5 Stratégie situationnelle

Au cours du processus, les négociations et les consommations de tâches sont concurrentes. La stratégie d'un agent consiste à choisir la prochaine délégation potentiellement socialement rationnelle et la prochaine tâche à exécu-

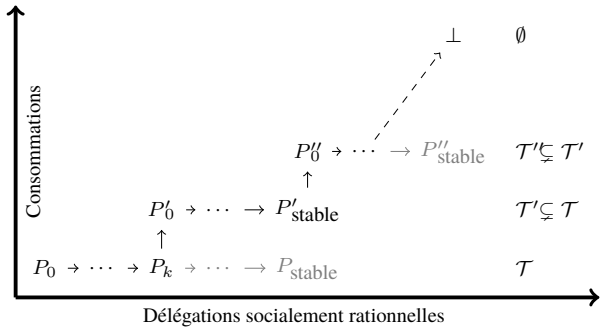


FIGURE 1 – Consommations de tâches (arcs verticaux) et délégations de tâches (arcs horizontaux) concurrentes.

ter.

Stratégie agnostique vis-à-vis de la localité. Dans une première approche, un agent i peut exécuter les tâches les plus grandes de son propre lot et négocier les plus petites parmi les délégations potentiellement socialement rationnelles. Formellement,

$$\overset{>}{\tau} = \underset{\tau \in P(i)}{\text{argmax}}\{c_i(\tau)\} \tag{17}$$

$$\overset{<}{\tau} = \underset{\tau \in \Gamma_i^{\mathcal{B}}(P)}{\text{argmin}}\{c_i(\tau)\} \tag{18}$$

où $\overset{>}{\tau}$ (resp. $\overset{<}{\tau}$) représente la prochaine tâche à exécuter (resp. à négocier). Cette stratégie nécessite que l'agent trie son lot de tâches selon leurs coûts. Cependant, acquérir une ressource consomme du temps et représente un coût supplémentaire lors de l'exécution.

Afin de mesurer la localité des tâches, on définit le ratio de possession d'une tâche par un agent i comme le ratio entre le nombre de ressources locales à l'agent pour cette tâche et le nombre total de ressources de la tâche :

$$o_i(\tau) = \frac{d_i(\tau)}{d_\tau} \tag{19}$$

Le ratio maximal de possession de τ est :

$$\hat{\text{o}}(\tau) = max_{i \in \mathcal{A}}\{o_i(\tau)\} \tag{20}$$

Stratégie situationnelle. Intuitivement, un agent devrait d'abord exécuter les tâches qui peuvent coûter plus pour les autres que pour lui-même, et déléguer les tâches qui peuvent coûter moins pour les autres. Selon cette stratégie, un agent exécute d'abord les grandes tâches locales et négocie d'abord les grandes tâches distantes, en fonction des ses connaissances et croyances locales, i.e. le coût de chaque tâche et son ratio de possession. Cette stratégie utilise une structure de données appelée lot par possession.

Le **lot par possession** de l'agent i (voir figure 2) est partagé en trois paquets selon le ratio de possession de l'agent pour ces tâches.

1. *Le paquet local* (noté MB) contient les tâches dont l'agent possède au moins une ressource et dont il est le plus gros propriétaire. Formellement,

   $$\forall \tau \in MB, o_i(\tau) \neq 0 \wedge o_i(\tau) = \hat{\text{o}}(\tau) \tag{21}$$

   Dans MB, les tâches sont triées par ordre décroissant de coût (cf. gauche de la figure 2).
2. *Le paquet semi-local* (noté IB) contient les tâches partiellement locales. Formellement,

   $$\forall \tau \in IB, 0 < o_i(\tau) < \hat{\text{o}}(\tau). \tag{22}$$

   Dans IB, les tâches sont triées par ordre décroissant de ratio de possession et les tâches ayant le même ratio de possession sont triées par ordre décroissant de coût (cf. centre de la figure 2).
3. *Le paquet distant* (noté DB) contient les tâches distantes. Formellement,

   $$\forall \tau \in DB, o_i(\tau) = 0. \tag{23}$$

   Dans DB, les tâches sont triées par ordre croissant de coût. (cf. droite de la figure 2).

Quand un agent recherche une tâche à exécuter, il commence par le début du paquet local, i.e. par la tâche locale la plus grande. Quand un agent cherche une tâche à négocier, il commence par la fin du paquet distant (i.e. la plus grande tâche distante) et il sélectionne la première qui constitue une délégation potentiellement socialement rationnelle selon ses croyances, $\mathcal{B}_i(P)$. Formellement,

— Un agent cherche une tâche à exécuter $\overset{>}{\tau} \in MB$ telle que

$$\forall \tau \in MB \setminus \{\overset{>}{\tau}\},\ c_i(\overset{>}{\tau}) \geq c_i(\tau). \tag{24}$$

Si $MB = \emptyset$, il cherche $\overset{>}{\tau} \in IB$ telle que

$$\forall \tau \in IB \setminus \{\overset{>}{\tau}\},$$
$$o_i(\overset{>}{\tau}) > o_i(\tau) \vee$$
$$\big(o_i(\overset{>}{\tau}) = o_i(\tau) \wedge c_i(\overset{>}{\tau}) \geq c_i(\tau)\big). \tag{25}$$

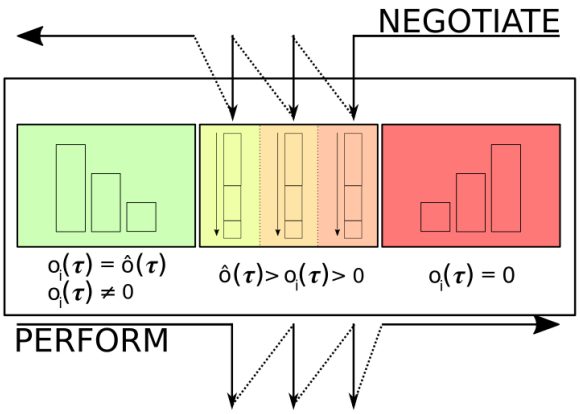


FIGURE 2 – Le lot par possession contient le paquet local (à gauche), le paquet semi-local (au centre) et le paquet distant (à droite). Les tâches sont représentées par des rectangles dont la taille est proportionnelle au coût de la tâche. Les flèches désignent l'ordre de parcours selon lequel un agent va chercher une tâche à exécuter/négocier.

Si $IB = \emptyset$, il cherche $\overset{>}{\tau} \in DB$ telle que

$$\forall \tau \in DB \setminus \{\overset{>}{\tau}\},\ c_i(\overset{>}{\tau}) \leq c_i(\tau). \quad (26)$$

Finalement, si $MB = IB = DB = \emptyset$, l'agent n'a aucune tâche à exécuter.

— Un agent cherche une tâche à négocier $\overset{<}{\tau} \in DB$ telle que

$$\forall \tau \in \big(DB \cap \Gamma_i^{\mathcal{B}}(P)\big) \setminus \{\overset{<}{\tau}\},\ c_i(\overset{<}{\tau}) \geq c_i(\tau). \quad (27)$$

Si $DB = \emptyset$, il cherche $\overset{<}{\tau} \in IB$ telle que

$$\forall \tau \in \big(IB \cap \Gamma_i^{\mathcal{B}}(P)\big) \setminus \{\overset{<}{\tau}\},\ o_i(\overset{<}{\tau}) < o_i(\tau) \vee \big(o_i(\overset{<}{\tau}) = o_i(\tau) \wedge c_i(\overset{<}{\tau}) \geq c_i(\tau)\big). \quad (28)$$

Si $IB = \emptyset$, il cherche $\overset{<}{\tau} \in MB$ telle que

$$\forall \tau \in \big(MB \cap \Gamma_i^{\mathcal{B}}(P)\big) \setminus \{\overset{<}{\tau}\},\ c_i(\overset{<}{\tau}) \leq c_i(\tau). \quad (29)$$

Finalement, si l'agent ne trouve pas une telle tâche, il arrête d'initier des enchères.

Il est important de remarquer que l'ordre leximax sur le paquet semi-local est conforme au principe selon lequel on exécute d'abord les grandes tâches locales et on négocie d'abord les grandes tâches distantes. Quand un agent explore le paquet semi-local, il commence par les tâches qui ont un ratio de possession élevé afin de les exécuter, et par celles qui ont un ratio de possession faible afin de les négocier. Quand un agent examine les tâches qui ont le même ratio de possession, il le fait toujours dans le même ordre : des plus coûteuses vers les moins coûteuses (cf. figure 2). Donc, quand un agent explore le paquet semi-local à la recherche d'une tâche à exécuter, il considère la plus locale et plus coûteuse ; et quand un agent explore le paquet semi-local à la recherche d'une tâche à négocier, il considère la moins locale et plus coûteuse.

	τ_1	τ_2	τ_3	τ_4	τ_5	τ_6	τ_7
$o_1(\tau_k)$	1	0	0.33	0.33	0.2	0.85	0
$o_2(\tau_k)$	0	1	0.66	0.66	0.8	0.14	1

TABLE 2 – Ratios de possession pour l'exemple 2

Exemple 2 (Lot par possession)**.** *Reprenons l'exemple 1 où nous supposons qu'une ressource distante est deux fois plus coûteuse qu'une ressource locale. Formellement,*

$$c_i(\tau) = d_i(\tau) + 2\big(d_\tau - d_i(\tau)\big) = 2d_\tau - d_i(\tau) \quad (30)$$

Le tableau 2 donne le ratio de possession de chaque agent pour chaque tâche. Considérons l'allocation où toutes les tâches sont affectées à l'agent 1*. Selon la stratégie agnostique, l'agent* 1 *ne fait pas de différence entre* τ_2 *et* τ_6 *pour une délégation puisque ces tâches ont le même coût pour lui. Selon la stratégie situationnelle, l'agent* 1 *dont le lot par possession est donné figure 3 considère :*

- *l'exécution des tâches* τ_6*,* τ_1*,* τ_4*,* τ_3*,* τ_5*,* τ_2*, puis* τ_7 *;*
- *la négociation des tâches* τ_7*,* τ_2*,* τ_5*,* τ_4*,* τ_3*,* τ_1*, puis* τ_6*.*

La stratégie situationnelle amène l'agent 1 *à exécuter* τ_6 *et à négocier* τ_7 *afin de diminuer le makespan, puisque* τ_6 *est en grande partie locale et* τ_7 *est distante. En particulier, l'agent* 1 *exécute* τ_6 *avant* τ_1 *car* τ_6 *est plus coûteuse (cf. table 1).*

Nous soulignons que les tâches dans le paquet semi-local sont triées d'abord selon le ratio de possession puis selon leur coût : que l'agent 1 *cherche à négocier ou à exécuter une tâche, il considère celles qui ont le même ratio maximal de possession et dans le même ordre (par ex.* τ_4*, puis* τ_3*).*

6 Application pratique

Comme application pratique, nous considérons le déploiement distribué du patron de conception MapReduce pour le traitement de jeux de

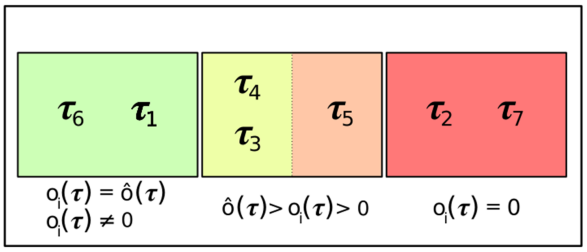


FIGURE 3 – Lot par possession de l'agent 1

données de grandes tailles sur une grappe de serveurs [9]. [17] identifient plusieurs biais qui pénalisent le temps nécessaire au traitement des données. En particulier, le biais de partitionnement se produit quand un *reducer* doit traiter un plus grand nombre de clés (et donc de tâches) que les autres. Ce biais produit un déséquilibre dans les charges de travail des *reducers*. La stratégie situationnelle permet aux agents d'équilibrer les charges et ainsi d'améliorer le temps d'exécution.

Nous avons déployé MapReduce à l'aide d'un système distribué multi-agents. Notre prototype a été implémenté en Scala avec la boîte à outils Akka adaptée aux applications orientées messages, fortement concurrentes, distribuées et robustes. Même si la tolérance aux pannes des nœuds est hors de la portée de cette étude, nous supposons que : (a) le délai de transmission des messages est arbitraire mais non négligeable, (b) l'ordre d'émission/réception des messages est identique par pair émetteur-récepteur et (c) les messages peuvent être perdus. Dans un tel système, les messages sont livrés 0 ou 1 fois. C'est la raison pour laquelle nous avons intégré un mécanisme d'interruptions temporelles dans le protocole d'interaction.

Nos expériences utilisent un jeu de données de 8 Gio ($82,283$ clés) qui a été généré de telle sorte que l'allocation initiale des tâches (cf. figure 4) soit déséquilibrée et de manière à pouvoir vérifier que la proximité entre les données et les nœuds de traitement a un impact sur le *makespan* et donc sur le temps d'exécution. L'allocation initiale des tâches est le résultat de la phase de partitionnement du processus MapReduce. En d'autres termes, quand il n'y a pas de négociation, notre système a le même comportement que Hadoop.

2. https://github.com/cristal-smac/mas4data
3. http://akka.io
4. Nos expériences ont été réalisées sur 16 PCs quadricœurs Intel(R) i7 avec 16GB RAM chacun.

Nous comparons, à partir de 10 exécutions pour chacune des (ré)allocations, les temps d'exécutions médians quand les agents adoptent une stratégie agnostique ou situationnelle. Nous observons que la stratégie situationnelle améliore significativement le temps d'exécution, d'environ -7.6%. Pour comparaison, le temps d'exécution sans remise en cause du partitionnement par défaut d'Hadoop est de $853s$ (environ $+100\%$). Nous en déduisons que le coût de la négociation peut être négligé au regard du gain obtenu par équilibrage des charges. De plus, la négociation permet de diminuer le temps d'exécution, tout particulièrement avec la stratégie situationnelle.

En raison du non-déterminisme de l'exécution distribuée, nous présentons un *run* typique. La figure 4 compare l'allocation de tâches quand toutes les tâches ont été traitées, qu'elles aient été négociées ou pas entre les 16 agents , selon les stratégies agnostique ou situationnelle. Les deux stratégies sont comparées *ex-post*. Nous remarquons que le *makespan* de l'allocation initiale vaut approximativement $3.3\ 10^8$, il est de $2.5\ 10^8$ (-24%) en cas de négociation avec la stratégie agnostique locale et de $2\ 10^8$ (-30.7%) avec la stratégie situationnelle. Nous en déduisons que la négociation, en particulier avec la stratégie situationnelle, permet d'améliorer l'équilibre des charges.

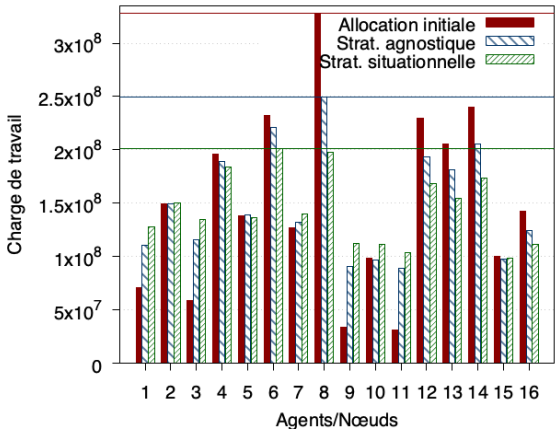


FIGURE 4 – Allocation initiale des tâches (Hadoop) et allocation *ex-post* avec les stratégies agnostique et situationnelle

La figure 5 présente le nombre de tâches t telles que $\alpha \leq o_i(t) < \alpha + 0.1$ quand l'agent i réalise t. L'allocation initiale de tâches est mal équilibrée

5. Une analyse empirique préliminaire montre qu'il n'y a pas de valeur ajoutée à avoir plus d'un *reducer* par nœud/disque.

car il y a plus de $4.6\ 10^4$ tâches pour lesquelles $o_i(\tau) = 0$. Nous observons que la stratégie situationnelle favorise le traitement des tâches qui sont les plus locales, i.e. $o_i(\tau) \geq 0.5$. Ce n'est pas le cas de la stratégie agnostique locale. Par exemple, il y a 171 tâches qui ont été traitées par un agent qui possède 60% des *chunks* (i.e. des ressources) avec la stratégie situationnelle, mais aucune avec la stratégie agnostique. Puisqu'elle favorise l'exécution des tâches par les nœuds qui sont les plus proches des chunks, la stratégie situationnelle améliore l'équilibre des charges et diminue le temps d'exécution.

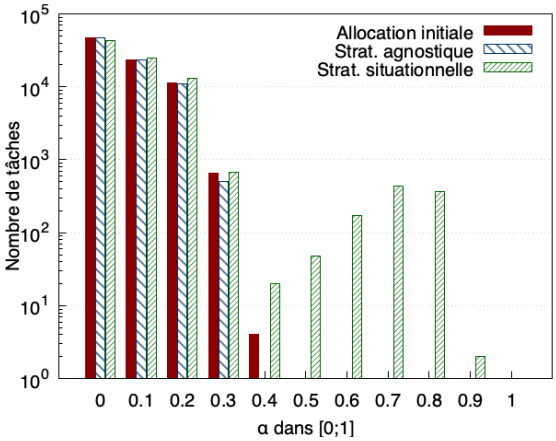


FIGURE 5 – Nombre de tâches par ratio de possession pour l'agent exécutant (échelle logarithmique)

7 Conclusion

Dans cet article, nous avons formalisé le problème de l'allocation multi-agents de tâches situées et nous avons proposé une stratégie situationnelle pour traiter le problème de l'équilibrage des charges dans les systèmes distribués. Cette stratégie adopte une approche fondée sur le marché qui nous permet de décentraliser l'équilibrage de charge pour minimiser le *makespan*, i.e. le temps nécessaire pour achever la dernière tâche. Au cours du processus, les négociations de tâches permettent aux agents qui utilisent la stratégie situationnelle de réallouer les tâches en même temps qu'ils en consomment. Cette stratégie permet également à un agent de déterminer la prochaine délégation socialement rationnelle et la prochaine tâche à traiter. Conformément à leur propre base de croyances et à leurs connaissances, les agents traitent en priorité les grandes tâches locales et négocient en priorité les grandes tâches distantes. Afin de valider notre approche, nous avons développé un prototype dans lequel les agents négocient les tâches de *reduce* du patron de conception MapReduce dans une configuration distribuée. Nos résultats expérimentaux montrent que la stratégie situationnelle proposée améliore l'équilibre de charge et donc le temps d'exécution pour de telles applications.

Actuellement, nous évaluons notre prototype dans différentes configurations matérielles, avec de nombreuses expérimentations qui se basent sur des jeux de données réels. Certaines configurations nous amènent à considérer que nous devons étendre notre stratégie pour négocier (a) des échanges de tâches afin d'améliorer le *makespan* des allocations stables, et (b) des lots de tâches pour accélérer les négociations.

Remerciements. Ce travail est soutenu par l'AAP ULille « Internationalisation Actions bilatérales ». Nous remercions le comité de programme des JFSMA qui, par ses remarques, nous a permis d'améliorer cet article.

Références

[1] Bo An, Victor Lesser, David Irwin, and Michael Zink. Automated negotiation with decommitment for dynamic resource allocation in cloud computing. In *Proc. of AAMAS*, pages 981–988, 2010.

[2] Martin R. Andersson and Tuomas W. Sandholm. Contract types for satisficing task allocation : Ii experimental results. In *Proc. of the AAAI spring symposium : Satisficing models*, pages 1–7, 1998.

[3] Gamal Attiya and Yskandar Hamam. Task allocation for maximizing reliability of distributed systems : A simulated annealing approach. *Journal of Parallel and Distributed Computing*, 66(10) :1259–1266, 2006.

[4] Quentin Baert, Anne-Cécile Caron, Maxime Morge, and Jean-Christophe Routier. Stratégie de découpe de tâche pour le traitement de données massives. In *Actes des JFSMA*, pages 65–74, 2017.

[5] Quentin Baert, Anne-Cécile Caron, Maxime Morge, Jean-Christophe Routier, and Stathis Kostas. Adaptive multi-agent system for situated task allocation. In *Proc. of AAMAS*, pages 1790–1791, 2019.

[6] Bo Chen, Chris N. Potts, and Gerhard J Woeginger. *Handbook of combinatorial optimization*, chapter A Review of Machine Scheduling : Complexity, Algorithms and Approximability, pages 1493–1641. Springer, 1998.

[7] Quan Chen, Daqiang Zhang, Minyi Guo, Qianni Deng, and Song Guo. SAMR : A self-adaptive MapReduce scheduling algorithm in heterogeneous environment. In *International Conference on Computer and Information Technology*, pages 2736–2743. IEEE, 2010.

[8] Anastasia Damamme, Aurélie Beynier, Yann Chevaleyre, and Nicolas Maudet. The Power of Swap Deals in Distributed Resource Allocation. In *Proc. of AAMAS*, pages 625–633, 2015.

[9] J. Dean and S. Ghemawat. MapReduce : Simplified data processing on large clusters. In *Symposium on Operating Systems Design and Implementation*, pages 137–150, 2004.

[10] Ulle Endriss, Nicolas Maudet, Fariba Sadri, and Francesca Toni. Negotiating socially optimal allocations of resources. *JAIR*, 25(1) :315–348, 2006.

[11] Saurabh Kumar Garg, Srikumar Venugopal, James Broberg, and Rajkumar Buyya. Double auction-inspired meta-scheduling of parallel applications on global grids. *Journal of Parallel and Distributed Computing*, 73(4) :450–464, 2013.

[12] A. M. A. Hariri and N. Potts, Chris. Heuristics for scheduling unrelated parallel machines. *Computers & operations research*, 18(3) :323–331, 1991.

[13] Ellis Horowitz and Sartaj Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *JACM*, 23(2) :317–327, 1976.

[14] Oscar H Ibarra and Chul E Kim. Heuristic algorithms for scheduling independent tasks on nonidentical processors. *JACM*, 24(2) :280–289, 1977.

[15] Yichuan Jiang. A survey of task allocation and load balancing in distributed systems. *IEEE Transactions on Parallel and Distributed Systems*, 27(2) :585–599, 2016.

[16] Qin-Ma Kang, Hong He, Hui-Min Song, and Rong Deng. Task allocation for maximizing reliability of distributed computing systems using honeybee mating optimization. *Journal of Systems and Software*, 83(11) :2165–2174, 2010.

[17] YongChul Kwon, Kai Ren, Magdalena Balazinska, and Bill Howe. Managing skew in Hadoop. *IEEE Data Eng. Bull.*, 36(1) :24–33, 2013.

[18] Jan Karel Lenstra, David B Shmoys, and Eva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical programming*, 46(1-3) :259–271, 1990.

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

[20] Silvano Martello, François Soumis, and Paolo Toth. Exact and approximation algorithms for makespan minimization on unrelated parallel machines. *Discrete applied mathematics*, 75(2) :169–188, 1997.

[21] Amro Najjar, Olivier Boissier, and Gauthier Picard. Négociation adaptative pour l'acceptabilité des services d'un fournisseur SaaS. In *Actes des JFSMA*, pages 85–94, 2017.

[22] Antoine Nongaillard and Philippe Mathieu. Reallocation problems in agent societies : A local mechanism to maximize social welfare. *JASSS*, 14(3) :21, 2011.

[23] Michael Schillo, Christian Kray, and Klaus Fischer. The eager bidder problem : a fundamental problem of DAI and selected solutions. In *Proc. of AAMAS*, pages 599–606. ACM, 2002.

[24] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artif. Intell.*, 101(1-2) :165–200, 1998.

[25] Reid G. Smith. The contract net protocol : High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, 29(12) :1104–1113, December 1980.

[26] Joanna Turner, Qinggang Meng, Gerald Schaefer, and Andrea Soltoggio. Distributed strategy adaptation with a prediction function in multi-agent task allocation. In *Proc. of AAMAS*, pages 739–747, 2018.

[27] William E Walsh and Michael P Wellman. A market protocol for decentralized task allocation. In *Proc. of ICMAS*, pages 325–332, 1998.

[28] Peng-Yeng Yin, Shiuh-Sheng Yu, Pei-Pei Wang, and Yi-Te Wang. Task allocation for maximizing reliability of a distributed system using hybrid particle swarm optimization. *Journal of Systems and Software*, 80(5) :724–735, 2007.

Approche Multi-agent pour l'analyse de journaux

F. Mouysset[a,b] F. Migeon[a] M.P. Gleizes[a] C. Bortolaso[b] M. Derras[b]

[a]Institut de Recherche en Informatique de Toulouse,
Université Toulouse 3, France
{firstname}.{surname}@irit.fr

[b]Berger-Levrault
Labège, France
{firstname}.{surname}@berger-levrault.com

Résumé

À mesure que les applications se complexifient, l'usage qui en découle dévie de leur conception. Il est alors intéressant de redécouvrir des modèles de ces processus métier a posteriori, notamment en analysant les journaux d'activité des utilisateurs. Cependant, ces journaux d'activité peuvent contenir des erreurs qui compliquent la découverte de modèles fiables et réalistes. Dans cet article, un système multi-agent (SMA) appelé SAMOTRACE est conçu et s'adresse à cette problématique. Sa mise en œuvre est basée sur des agents auto-organisés. Les expériences montrent que le système tend à converger vers une solution optimale, quels que soient le type et la quantité d'erreurs présentes dans les observations.

Mots-clés : *Système Auto-Organisé, Journaux d'Evènements, Système Multi-Agent*

1 Introduction

Comme de nombreux logiciels modernes, la complexité des logiciels destinés aux services publics augmente avec l'évolution constante des réglementations et des besoins utilisateurs. Ceci a des impacts majeurs sur la qualité des logiciels et les processus de maintenance. Pour l'éditeur, cela se caractérise par un nombre croissant de bogues [11] difficilement reproductibles. De plus, l'équipe projet conçoit les tests a priori, qui peuvent ne pas refléter les usages des utilisateurs. Ainsi, un écart apparaît entre l'utilisation prévue et la réalité, conduisant à des cas inattendus, non testés et donc possiblement erronés.

Enfin, chaque évolution logicielle tend à rendre plus difficiles son utilisation et sa compréhension [12]. L'expérience utilisateur s'en trouve détériorée et le risque d'erreur augmente en conséquence.

À la lumière de ces constats, il parait clair que la qualité du logiciel doit être améliorée. L'usage effectif doit guider les travaux d'amélioration. La compréhension des usages réels devient primordiale.

La plupart des logiciels modernes consignent les actions qui sont réalisées dans des journaux. Un journal contient la séquence des actions réalisées pour chaque utilisateur. Chaque action est perçue comme un évènement. A minima, un évènement est composé du nom de la tâche effectuée, d'un horodatage et d'un identifiant utilisateur. Cependant, un même utilisateur peut exécuter plusieurs instances de cas d'utilisation en concurrence. C'est par exemple le cas lorsqu'un utilisateur ouvre plusieurs instances d'un même logiciel. Un évènement doit alors indiquer dans quelle instance de cas d'utilisation se trouve l'utilisateur. Ces informations sont appelées ID d'instance de cas d'utilisation (ou ID d'instance de processus). Un journal contenant de telles données est « étiqueté ». Il n'est pas toujours facile de déterminer à quelle instance de processus est lié un évènement. Par conséquent, dans les logiciels complexes, il est assez courant de trouver des journaux composés d'évènements sans identifiant d'instance de cas d'utilisation. Ces journaux sont qualifiés de « non étiquetés ».

En plus d'être souvent non étiquetés, les journaux d'activité utilisateur peuvent contenir des erreurs comme des évènements

manquants ou désordonnés. Enfin, le fait de ne pas étiqueter les journaux induit un « entrelacement » entre les évènements appartenant à des instances de cas d'utilisations différents. L'analyse des journaux s'en trouve compliquée.

L'état de l'art montre que les approches actuelles, de découverte de modèle de processus à partir de journaux, sont inadaptées lorsqu'elles concernent des journaux d'évènements complexes, bruités et non étiquetés. Nous avons fait l'hypothèse qu'il était alors possible de concevoir un Système Multi-Agent (SMA) capable de corriger les erreurs dans les journaux tout en découvrant de tels modèles. Cependant, cet objectif est trop difficile pour être réalisé en une seule fois. Notre conception est alors décomposée en itérations et cet article décrit la première dans laquelle le modèle de processus est supposé connu. Ainsi, la section 3 présente un SMA, appelé SAMOTRACE, conçu pour détecter et corriger (de manière optimale) les journaux. La section 4 décrit les expériences menées et analyse les résultats. Enfin, certaines limites sont pointées et nos futures itérations sont présentées.

2 Contexte et travaux connexes

Un précédent article [10] établit une revue des domaines scientifiques s'intéressant à la découverte de modèle de processus. En particulier y est présenté le process mining [1]. Celui-ci peut être divisé en deux catégories d'approches : celles qui traitent des journaux non étiquetés et celles qui traitent des journaux étiquetés. Nous avons montré que la première catégorie n'est pas souhaitable pour les journaux logiciels, en raison du peu d'informations sur le comportement réel des utilisateurs. La seconde catégorie a été expérimentée avec une approche similaire à Astromskis [3] et a montré que la complexité des journaux ne peut pas être traitée par des approches de process mining.

La communauté scientifique s'entend sur le fait que les SMA contribuent à l'analyse des données et à la découverte des connaissances [4]. Il existe plusieurs approches ayant des objectifs différents.

Abdelkafi [2] propose un modèle SMA pour la découverte de l'organisation à partir des journaux de workflow [7]. Cette approche se limite aux workflows répartis entre plusieurs acteurs, ce qui n'est pas le cas pour beaucoup de logiciels.

Les SMA peuvent également être utilisés comme outil de simulation pour améliorer les modèles de processus existants [5]. Casalicchio propose une approche basée sur les agents pour modéliser et simuler des workflows dans l'administration publique. Par cette approche, les exigences non fonctionnelles et l'équilibre entre le dimensionnement des ressources et les modifications dans le workflow sont mieux contrôlés.

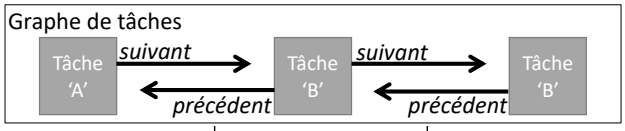
Récemment, Halaška [6] a conçu MAREA, un outil de modélisation et de simulation basé sur un SMA. L'outil est une source de données pour des activités l'analyse de process mining. De manière analogue, à Casalicchio, MAREA réalise des simulations réalistes des processus afin de tester des changements avant leur mise en œuvre. L'utilisation d'un simulateur orienté agent autorise un large éventail de comportements, et les simulations s'en trouvent plus réalistes. En particulier, les simulations ne se limitent pas aux comportements linéaires et stationnaires, mais peuvent recréer des comportements complexes et émergents.

Enfin, Clair et all [8] proposent AMAS4Opt un SMA pour l'optimisation de la fabrication des produits, plus précisément sur la planification du contrôle de fabrication et la conception de produits complexes. Cette méthode permet de résoudre des problèmes d'optimisation complexes dans un environnement dynamique. En ajoutant des comportements coopératifs entre agents, le système converge vers l'optimum.

Clair et Halaška montrent que les approches SMA sont recommandées pour les problèmes dynamiques et complexes. En outre, Halaška indique que « les processus d'entreprise peuvent également être considérés comme des systèmes très complexes ». Ainsi, les journaux produits par de tels logiciels reflètent cette complexité. Par conséquent, les SMA semblent pertinents pour le traitement des journaux logiciel, en particulier pour la découverte de modèles de processus.

2.1 Définition du problème

La résolution des erreurs contenues dans les journaux nécessite une réorganisation de l'ordre des évènements. La Fig. 1 décrit un modèle de processus et différentes traces produites par ce même modèle. Pour cet exemple, le modèle est réduit à 3 tâches en séquence, mais des modèles plus complexes sont admis (avec plusieurs tâches suivantes et précédentes, des boucles de différentes tailles…). Les journaux illustrent différentes situations possibles : journal n° 1 sans erreur, journal n° 2 un désordre local, journal n° 3 un évènement manquant. L'ordre dans lequel les évènements sont enregistrés est appelé « ordre chronologique ». Cet ordre peut être erroné, c'est pourquoi l'ordre « logique » est défini. L'ordre logique décrit des relations entre évènements qui sont conforment au graphe de tâches. Pour cela, des réorganisations peuvent être nécessaires par rapport à l'ordre chronologique, ces réorganisations définissent l'ordre logique. Par exemple, le journal n° 2 peut être corrigé en réorganisant l'ordre des évènements : l'évènement n° 4 devrait être avant l'évènement n° 3. Le journal n° 3, lui, nécessite une correction plus profonde en créant un évènement « de complétion » qui comble le manque d'évènement enregistré.
Cependant, plusieurs corrections peuvent être possibles pour un même problème. Dans le journal n° 2, deux réorganisations sont possibles, l'évènement d'index chronologique



Journal n° 1 'sans erreur'

Indice chrono-logique	Indice logique	Tâche réalisée
1	1	A
2	2	A
3	3	B
4	4	B
5	5	C

Journal n° 2 'désordonné'

Indice chrono-logique	Indice logique	Tâche réalisée
1	1	A
2	2	A
3	4	C
4	3	B
5	5	B

Journal n° 3 'évènement manquant'

Indice chrono-logique	Indice logique	Tâche réalisée
1	1	A
2	2	A
3	3	B
4	4	C
5	5	C

FIG. 1. EXEMPLE DE GRAPHE DE TACHES ET DE JOURNAUX (AVEC ET SANS ERREURS)

n° 4 est inséré soit avant l'évènement d'index chronologique n° 3, soit avant l'évènement d'index chronologique n° 5. Si l'on considère l'erreur avec le plus petit désordre (en termes de distance entre évènement) comme le plus probable alors la première réorganisation serait la plus appropriée (rasoir d'Occam). Celle-ci propose alors une solution qui minimise, au mieux, la distance entre les évènements. Si chaque évènement minimise cette distance alors elle l'est également globalement. Dans le reste de l'article, nous appelons cette métrique « distance chronologique ».

Dans cet article, les propriétés d'auto-organisation des systèmes multi-agents sont utilisées pour déterminer l'ordre logique des évènements qui optimisent cette mesure.

3 SAMOTRACE

Dans cette première version, l'objectif de SAMOTRACE est de détecter et corriger les erreurs contenues dans les journaux d'évènements. Le modèle de processus est supposé connu et correct. Ce SMA est basé sur la théorie des systèmes multi-agents adaptatifs [9]. Les propriétés d'auto-organisation sont utilisées pour explorer et converger vers l'organisation corrigeant au mieux les journaux.

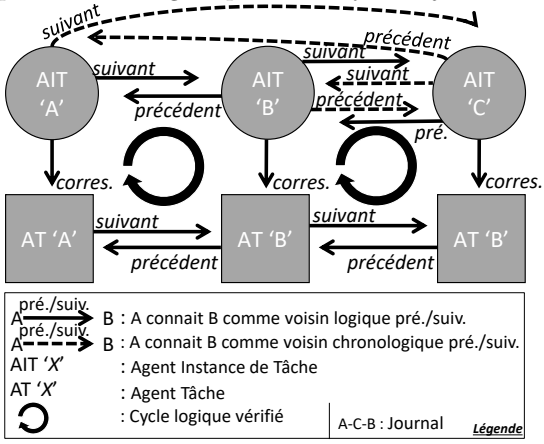


FIG. 2 – VOISINAGES ET CYCLES LOGIQUES

Les principaux concepts de la Fig. 1, sont agentifiés et représentés dans la Fig. 2. Ainsi dans SAMOTRACE on définit trois principaux types d'agents :

1) L'Agent de Tâches (AT) qui représente une tâche dans le graphe de tâches.

2) L'Agent Instance de Tâche (AIT) qui représente un évènement.

La figure 2 montre trois AT et trois évènements qui représentent, respectivement, la réalisation des tâches 'A', 'B' et 'C'. L'on dit que les AIT et AT correspondent entre eux.

3) L'Agent de Complétion (AC) qui est créé lorsqu'un évènement (ou une série d'évènements) est manquant.

Les agents sont dans un environnement social, toutes les accointances sont donc soit innées soit acquises par message. Les agents n'ont absolument pas « conscience » de l'existence du journal. Initialement, tous les AT sont créés à l'image du graphe de tâche et aucun AIT n'existe.

3.1 Agent Tâche (AT)

Une arête, dans le graphe de tâche, est une relation de voisinage donc une connaissance. Chaque AT connait ses voisins AT suivants et précédents. Les AT et leurs voisinages modélisent alors le modèle de processus. Pour l'instant, ces connaissances sont immuables et innées pour les AT, ainsi leurs relations de voisinage n'évoluent pas. La perception des AT est limitée à leurs voisins.

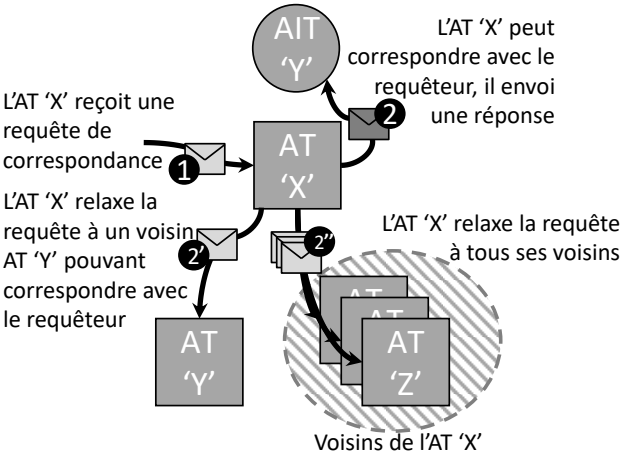


FIG. 3 – COMPORTEMENT DE L'AT

Le rôle de l'AT est d'aider les AIT à trouver l'AT avec lequel correspondre. Pour cela, les AIT envoient des requêtes spécifiques. La Fig. 3 détaille le comportement d'un AT. En ❶, l'AT 'A' reçoit une requête. Cette requête peut provenir d'un autre AT ou d'un AIT (voir 3.2). Trois possibilités :

- ❷ l'AT peut correspondre à l'AIT demandeur, il lui répond alors.
- ❷' l'AT connait un voisin AT qui peut correspondre avec le demandeur.
- ❷'' l'AT ne connait pas d'agents capables de satisfaire la requête, il la relaxe alors à tous ses voisins.

Le mécanisme de relaxation est limité pour éviter les problèmes d'inondation et d'auto-entretien de messages.

3.2 Agent Instance de Tâche (AIT)

L'AIT possède un voisinage dit « chronologique » qui reflète l'ordre chronologique des évènements dans le journal (Fig. 2 flèches pointillés). Ce voisinage est inné et immuable. À la lecture d'un évènement du journal, un AIT est créé et son voisinage chronologique est initialisé conformément à l'ordre du journal ; et il est mis en relation avec un AT candidat. L'objectif principal des AIT est de trouver et de corriger au mieux l'ordre chronologique c'est pourquoi on définit également le voisinage logique. Dans la Fig. 2 le voisinage logique décrit un ordre alternatif correct. Ce voisinage est initialisé avec le voisinage chronologique.

Pour atteindre son but l'AIT cherche à minimiser sa criticité qui représente la distance à son but. La criticité d'un AIT est un leximin de 3 critères : 1) connaissance d'un AT avec lequel correspondre, 2) connaissance d'AIT vérifiant les cycles logiques et 3) connaissances d'AIT optimisant la distance chronologique. Pour chaque critère, l'AIT dispose de stratégies basées sur la réorganisation de son voisinage logique. Le premier critère doit être vérifié pour passer aux suivants.

1) Trouver sa correspondance avec l'AT

À la mise en relation avec un AT, l'AIT vérifie sa correspondance. Soit l'AIT a trouvé son AT et met à jour sa connaissance de correspondance. Soit, l'AIT envoie une requête de correspondance à l'AT, pour diffusion aux autres AT. Cette partie du comportement est spécifiée sous forme de règles appelées « règles de correspondance ».

L'AIT est un agent coopératif, donc toute information pouvant aider ses voisins doit être partagée avec eux. C'est le cas lorsqu'un AIT trouve l'AT avec lequel correspondre. En effet, les voisins de cet AT peuvent correspondre avec les voisins de l'AIT. Par exemple, dans la Fig. 4 (gauche), en ❶, l'AIT 'A' reçoit une réponse de correspondante par l'AT 'A'. Il met donc à jour sa connaissance de correspondance ❷①. L'AIT 'A' peut alors aider son voisin l'AIT 'B'. Pour cela, l'AIT 'A' perçoit les voisins suivants de l'AT 'A', il dispose alors de toutes les informations pour déterminer s'il existe une correspondance entre l'AIT 'B' et un des voisins. Si l'AT 'B' correspond, l'AIT 'A' informe alors l'AIT 'B' en ❷②. Ainsi, l'AIT 'A' a aidé l'AIT 'B' à trouver sa correspondance. L'AIT 'B' met à jour ses connaissances ❸, il peut alors aider ses voisins à son tour.

2) Détection et correction d'erreurs dans le journal

Le deuxième objectif d'un AIT est de détecter et corriger les erreurs du journal. Pour cela, l'AIT vérifie ses deux cycles logiques. Un AIT vérifie un cycle logique si l'AT avec lequel il correspond a un AT voisin qui est l'AT de son AIT voisin. La Fig. 2 illustre le concept de cycles logiques. Notons qu'un AIT doit vérifier (au pire) deux cycles logiques, un précédent et un suivant.

Dans la Fig. 4 (gauche) le cycle logique est vérifié par l'AIT 'A' après sa correspondance avec AT 'A'. En revanche, dans la Fig. 4 (droite), le cycle logique est invalidé par l'AIT 'A'. Il invalide sa connaissance sur son voisinage logique suivant ❷③ et transmet cette invalidation à l'AIT 'C' ❷② (règle d'invalidation). À la réception de cette information, l'AIT 'C' invalide sa connaissance sur son voisinage logique précédent avec l'AIT 'A' ❸. Finalement, l'AIT 'A' et l'AIT 'B' ont chacun un voisin logique manquant. La stratégie pour retrouver un nouveau voisin logique dépend du type d'erreur.

Le premier type d'erreur est le désordre local. Ainsi, un AIT recherche localement un autre AIT pouvant vérifier le cycle logique avec lui. Pour cela, l'AIT envoie à ses voisins chronologiques une requête « de recherche d'un voisin logique ». Cette requête est traitée par les voisins qui déterminent s'ils peuvent devenir ce nouveau voisin. Si c'est le cas, l'AIT informe le demandeur et met à jour ses connaissances sur son voisinage logique. Sinon la requête est relaxée auprès de l'autre voisin qui évaluera la demande à son tour... Comme l'erreur est supposée être locale, la requête est limitée en nombre de relaxations. Le comportement décrit ici forme la « règle de recherche de voisin logique local ». Lorsqu'un requêteur envoie une requête, il définit le nombre de relaxations en fonction du délai de réponse estimé. Lorsque le délai

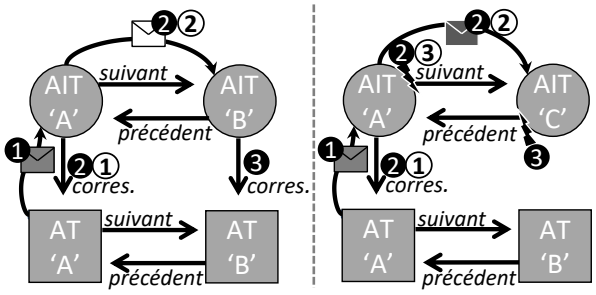


FIG. 4 – VALIDATION (GAUCHE) OU INVALIDATION (DROITE) DE CYCLE LOGIQUE

de réponse expire, l'agent considère le second type d'erreur.

Le deuxième type d'erreur est le désordre distant. Cela peut être dû à l'entrelacement d'évènements appartenant à des instances de cas d'utilisation différents. Par conséquent, certains AIT ont des voisins logiques manquants et ces AIT devraient être voisins logiques mutuellement. L'entité de liaison (EL) est une entité facilitatrice qui met en relation ces agents. Ainsi, un AIT contacte l'EL lorsqu'il ne parvient pas à résoudre une erreur locale. Lorsque l'EL reçoit cette demande, elle la mémorise. Également, elle retourne au demandeur une liste de toutes les requêtes compatibles précédemment mémorisées. Le demandeur reçoit alors tous les identifiants des agents susceptibles de vérifier le cycle logique avec lui, le meilleur est choisi (détaillé dans le paragraphe *3*). Ce comportement est régi par les « règles protocolaires avec l'entité de liaison ».

Le troisième type d'erreur est envisagé lorsqu'aucun nouveau voisin n'est trouvé. Dans ce cas, il ne s'agit plus de réorganiser les voisinages logiques, mais plutôt de créer un voisin logique manquant. Ainsi, l'AIT crée un agent de complétion (AC). Le rôle de l'AC est de compléter le journal lorsqu'il contient des évènements manquants. Dans la Fig. 5, l'évènement matérialisant la réalisation de la tâche 'B' est absent. Pour vérifier leurs cycles logiques les AIT 'A' et 'B' doivent créer un

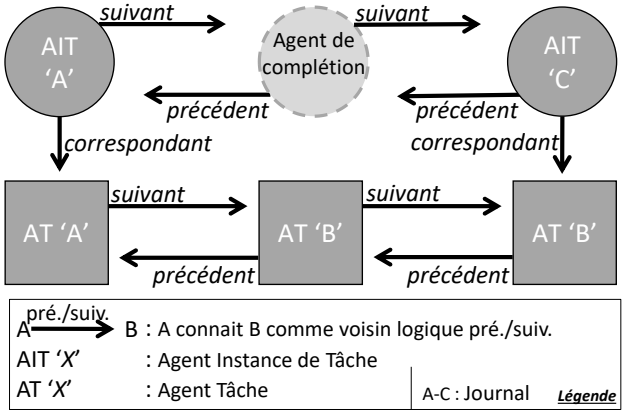


FIG. 5 – L'AGENT DE COMPLETION RESTAURE LA CONNEXITE DES EVENEMENTS

AC. Cependant, un AIT préfère toujours vérifier un cycle logique avec un autre AIT qu'avec un AC. Ainsi, si par la suite, l'AIT 'A' et/ou AIT 'C' trouvent (via l'entité de liaison) un AIT 'B', ils pourront se défaire de l'AC pour réorganiser leurs voisinages avec l'AIT 'B'. La règle de « création d'AC » est la dernière partie du comportement des AIT.

3) Optimisation de l'ordre logique

Le troisième et dernier objectif vise à éviter les minimums locaux en minimisant les distances chronologiques entre deux AIT. En effet, plusieurs AIT peuvent satisfaire le même cycle logique. Par exemple, Fig. 6, l'AIT 'B' reçoit une demande de recherche de voisin manquant ❶. Il a le choix entre deux AIT 'A', l'un avec un index chronologique de 1 et l'autre de 3. Pour déterminer quel AIT minimise la distance chronologique, l'AIT 'B' anticipe les conséquences du remplacement de son voisin logique courant par le candidat ❷. Il calcule les distances chronologiques pour chacun des agents candidats. La distance chronologique avec le voisin courant et AIT 'B' est de 4, tandis que la distance entre le candidat est de 2. La distance chronologique anticipée diminue par rapport à la distance actuelle. Le remplacement du voisin courant par le candidat est coopératif. Pour cela, AIT 'B' invalide sa connaissance de voisinage logique avec l'AIT 'A' 1 ❸① et lui envoie un message d'invalidation ❸②. Dans le même temps, il envoie un message de validation à AIT 'A' 3 ❸③. Ici, l'anticipation permet

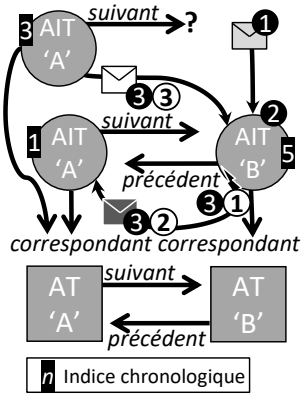


FIG. 6 – OPTIMISATION DU VOISINAGE LOGIQUE

d'estimer les effets du remplacement sur la distance chronologique. Mais ce n'est qu'une composante de l'anticipation. Tous les critères du leximin sont utilisés.

4 Expérimentations

Nous avons réalisé deux séries d'expériences pour valider le comportement du SMA. Chacune des expériences nécessite un journal d'évènements et un graphe de tâches comme entrées. La sortie est le journal réordonné et les criticités résiduelles. L'ensemble est comparé à un test de référence.

La première série d'expériences est constituée de tests fonctionnels. Chaque test aborde une partie spécifique du problème à résoudre. Comme l'indique le Tableau I, ces tests sont ad hoc et minimaux (graphes de tâches et journaux minimaux), mais ils demeurent représentatifs des erreurs possibles.

De plus, la simplicité des tests permet une approche de conception incrémentale du système. Chaque incrément se préoccupe d'une nouvelle fonctionnalité, c'est-à-dire d'un nouveau type d'erreur. L'ajout d'une fonctionnalité se traduit par l'évolution et l'ajout de règles. L'ensemble des tests est continuellement joué afin d'y détecter, au plus tôt, une éventuelle régression. Bien qu'incrémentale, cette conception est basée sur une approche de développement pilotée par les tests. Enfin, notons que l'utilisation de tests simples facilite l'explicabilité et le débogage du système (à une échelle réduite).

La deuxième catégorie d'expériences comprend les tests non fonctionnels de passage à l'échelle, ils impliquent des graphes de tâches et des journaux plus conséquents. Les données en entrée sont générées automatiquement selon différentes configurations. Par exemple, la synthèse des graphes de tâches est paramétrée par le nombre de tâches et leur connectivité moyenne. Puis, des marches aléatoires dans ces graphes produisent les journaux d'évènements. Cette étape est paramétrée par le nombre d'évènements souhaités. Certains journaux sont modifiés manuellement pour ajouter les différents types d'erreurs.

Chaque test est exécuté plusieurs fois avec différente configuration listée dans le tableau II. Par exemple, le test n° 1 a 12

TABLEAU II : TESTS DE PASSAGE A L'ECHELLE

#	**Objectif du test**	**Caractéristiques du graphe de tâches**	**Caractéristiques du journal**	**Résultat attendu**
1	Graphes de tâches de taille moyenne et pas d'erreur dans les journaux	Nombre de tâches : 50, 100, 500, 2500 Connectivité moyenne : 2 to 5	Nombre d'évènements : entre 50-70	1 solution optimale sans criticité résiduelle
2	Graphe de tâche important sans erreur	Nombre de tâches : 10.000 Connectivité moyenne : 2	Nombre d'évènements : 500-2.500	1 solution optimale sans criticité résiduelle
3	n° 1 et n° 2 avec erreur dans les journaux	n° 1 et n° 2	Différents types d'erreurs ajoutées manuellement	Des criticités résiduelles les plus petites possibles

TABLEAU I : TESTS FONCTIONNELS

#	**Objectif du test**	**Règles impliquées**	**Caractéristiques des entrées**	**Journal réordonné attendu**	**Criticité résiduelle attendue**
1	Cas parfait	Règles de correspondance et vérification de voisinage logique	Graphe de tâches tel que défini dans la Fig. 1 et journal n° 1 (Fig. 1)	*A-B-C*	Pas de criticité résiduelle
2	Auto-organisation pour la résolution de désordre local	Règles n° 1 + règle de vérification de voisinage logique + règle de recherche de voisin logique local	Graphe de tâches tel que défini dans la Fig. 1 et journal de 3 évènements avec un désordre local (B et C) : A-C-B	A-B-C	distance chronologique =6
3	Auto-organisation pour la résolution de désordre distant	Règles n° 2 + règles protocolaires avec l'*entité de liaison*	Graphe de tâches tel que défini dans la Fig. 1 et journal de 6 évènements avec un désordre distant : A-C-A-C-B-B	A-B-C A-B-C	distance chronologique =24
4	Évolution pour la résolution d'évènement manquant	Règles n° 3 + règle de création d'*agent de complétion*	Graphe de tâches tel que défini dans la Fig. 1 et journal de 2 évènements avec un évènement manquant : A-C	A – α – C (α *agent de complétion)*	1 *agent de complétion*

[a.] Pour créer un désordre distant, exceptionnellement pour ce cas de test, le nombre de relaxations est égal à 1.

configurations (3 x 4). Chaque graphe de tâches est utilisé pour générer 50 journaux de différentes tailles (fonction de la taille du graphe de tâches).

Chaque configuration de test est alors exécutée plusieurs fois. Contrairement à un agent qui possède un nombre limité d'états, un SMA, par la combinatoire de tous les agents, décrit un ensemble très important d'états. Tester tous ces états est impossible. Pire, à cause de la nature dynamique et complexe du système, il est impossible de prouver la convergence dans le cas général. Nous ne pouvons valider le système que par un grand nombre d'expérimentations.

Les tests sont effectués sur une machine standard (Intel® Core™ i7-7500U CPU @2.70GHz - RAM 16 Go). Le SMA est implémenté en Java (JDK 1.8).

Chaque test fonctionnel est exécuté 500 fois. Une exécution dure entre 900 ms et 1,5s et nécessite entre 60 et 120 cycles d'exécution.

4.1 Résultats

Tous les tests fonctionnels et de passage à l'échelle ont été validés.

1) Résultats des tests fonctionnels

L'expérimentation débute avec toutes les règles désactivées. L'ensemble minimal de règles est ajouté pour rendre le système fonctionnel. Des erreurs sont ensuite ajoutées afin de démontrer l'insuffisance des règles activées. De nouvelles règles sont alors ajoutées et ainsi de suite jusqu'à ce que toutes les typologies d'erreurs soient résolues.

L'exécution du test n° 1 sans aucune règle n'entraine aucune résolution. Le nombre d'AIT sans correspondance demeure élevé et inchangé (égal au nombre d'évènements). La Fig. 7 montre l'évolution des criticités moyennes et leurs enveloppes de variation avec l'activation des règles de correspondance (uniquement). Les 3 AIT trouvent leur correspondant AT. De plus, pour un journal sans erreur, aucune règle supplémentaire n'est nécessaire pour la vérification des voisinages

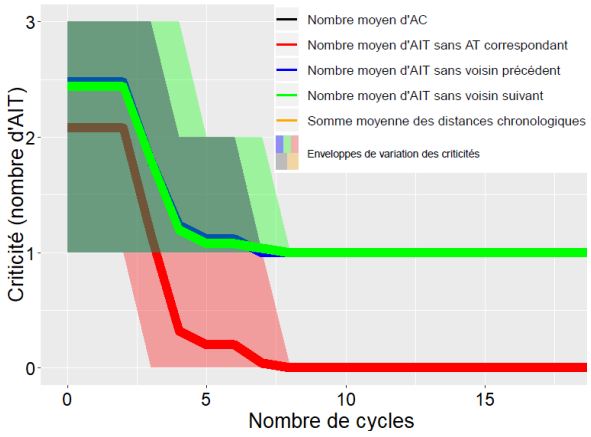


FIG. 7 – JOURNAL SANS ERREUR (TEST N°1, REGLES DE CORRESPONDANCE ACTIVEE)

logiques. Seuls, les deux voisinages logiques sont manquants aux extrémités du journal. AIT 'A' n'a pas de précédent et AIT 'C' n'a pas de suivant.

Puisque le système est fonctionnel, un désordre local est ajouté. Sur la Fig. 8 (haut), le nombre de voisins logiques manquants est de 6 (3 agents sans voisins précédents et suivants). En activant les règles associées au test n° 2, les désordres locaux sont résolus. Dans la Fig. 8 (en bas), le nombre de voisins logiques manquants est égal à 2 (toujours dû

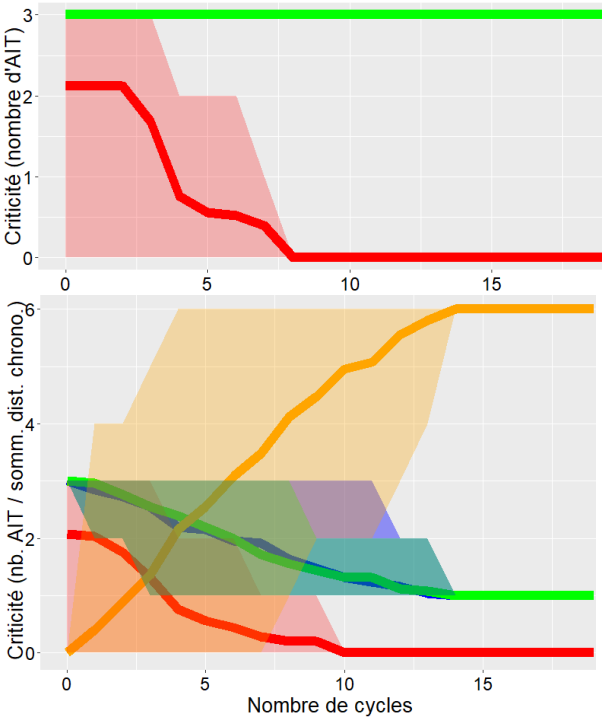


FIG. 8 – JOURNAL AVEC UN DESORDRE LOCAL (TEST N° 2). HAUT SANS LES REGLES, BAS AVEC LES REGLES

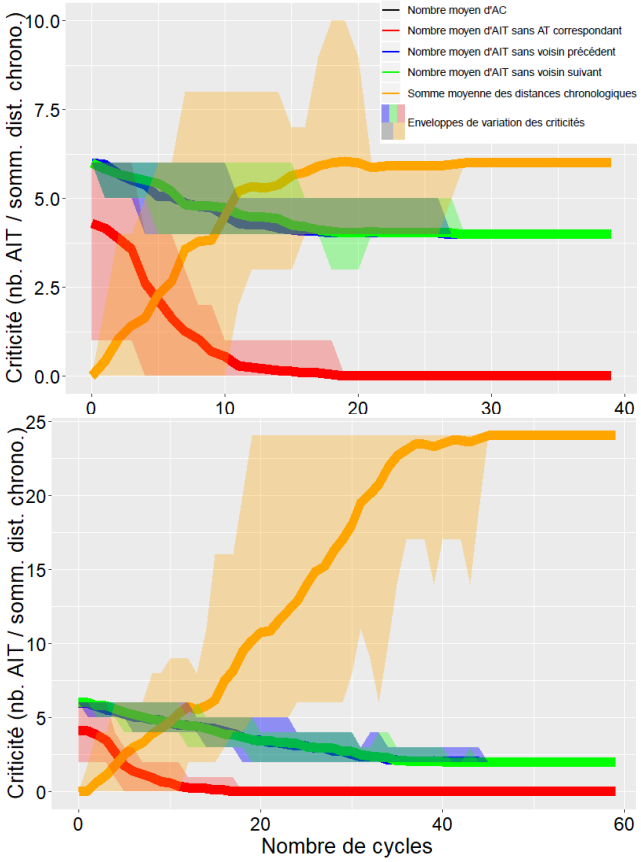


FIG. 9 – JOURNAL AVEC UN DESORDRE DISTANT (TEST N°3). HAUT SANS LES REGLES, BAS AVEC LES REGLES.

aux extrémités du journal). La somme des distances chronologiques augmente parce que les nouveaux voisinages logiques sont différents du voisinage chronologique.

Le test n° 3 débute par l'ajout d'un désordre distant, l'ensemble de règles est inchangé. Il en résulte des criticités résiduelles, Fig. 9 (haut), où 4 voisins logiques sont manquants. L'activation des règles du test n° 3, Fig. 9 (bas), résout le désordre distant. Mais, à nouveau, cela se traduit par une augmentation de la distance chronologique.

Enfin, dans le test n° 4 le journal contient un évènement manquant. Sans règle supplémentaire, aucun des voisinages logiques n'est vérifié. En revanche, l'activation de la règle de création d'AC résout ce problème. Dans la Fig. 10, nous observons que contrairement aux tests précédents, le nombre de voisins logiques manquants est de 0. En plus, le nombre d'AC n'est pas de 1, comme attendu, mais de 2. En fait, la possibilité de créer des AC a permis de relier les deux extrémités du journal via un AC. Le résultat du test est donc correct.

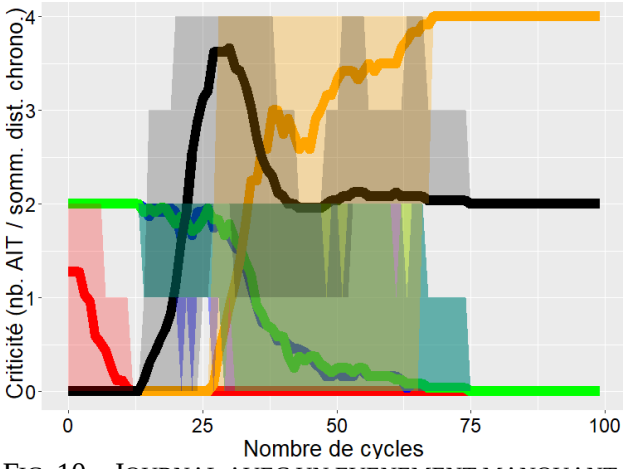


FIG. 10 – JOURNAL AVEC UN EVENEMENT MANQUANT (TEST N° 4 REGLES ACTIVEES).

2) *Résultats du passage à l'échelle*

Pour ces tests, les données en entrées sont générées et les erreurs sont ajoutées manuellement. Deux tests sont ici détaillés. Le premier est constitué d'un graphe de 10 tâches et d'une connectivité moyenne de 5. Le journal généré contient 1000 évènements et 10 erreurs de chaque type sont ajoutées. Le second test décrit une expérience avec 1000 tâches avec une connectivité moyenne de 5. Le journal est composé de 786 évènements, avec les mêmes types de bruit. Chaque test est exécuté 25 fois (avec les mêmes données en entrées). Pour ces 25 runs, l'évolution des criticités au cours de la résolution est retracée respectivement FIG. 11 et Fig. 12.

Dans les deux figures, le nombre d'AIT sans correspondant et le nombre de voisins logiques manquants diminuent ❶. Pour satisfaire les cycles logiques, des AC sont créés. Par conséquent, le nombre de voisins logiques manquants diminue à mesure que le

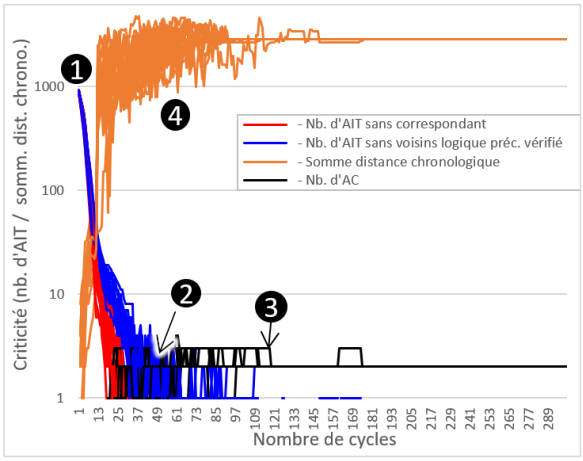


FIG. 11 – ÉVOLUTION DE LA CRITICITE – 10 TACHES ET JOURNAL AVEC ERREURS

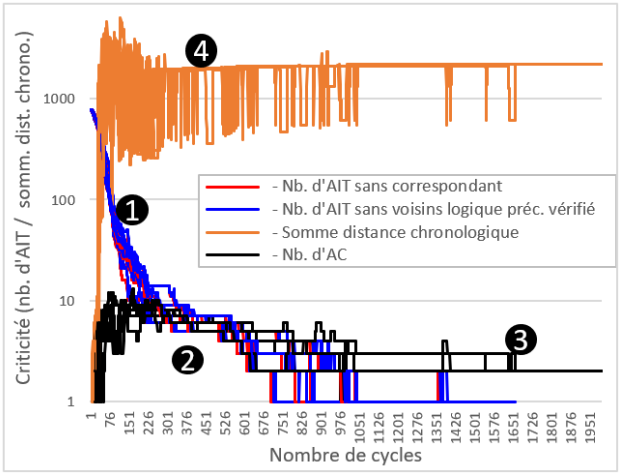


FIG. 12 – ÉVOLUTION DE LA CRITICITE – 1000 TACHES ET JOURNAL AVEC ERREURS

nombre d'AC augmente ❷. Après quelques réorganisations et contacts avec l'EL, plusieurs AIT se séparent de leur AC ❸ (diminution du nombre d'AC). Cependant, la distance chronologique augmente ❹. Enfin, le système se stabilise.

Pour toutes les exécutions, le système est parvenu à converger vers un résultat.

5 Conclusion et perspectives

Nous avons fait l'hypothèse que l'approche SMA pourrait être adaptée à l'analyse de journaux d'activité d'utilisateur sur des logiciels, en particulier pour la découverte de modèles de processus. Afin de valider cette hypothèse, nous avons conçu et mis en œuvre SAMOTRACE, un système multi-agent auto-organisé et adaptatif.

Cet article présente la première itération de notre travail : la résolution des erreurs contenues dans les journaux. Cette approche a été validée par deux types de tests. Les tests fonctionnels montrent la cohérence de notre base de règles. Tandis que les tests de passage à l'échelle montrent la viabilité de l'approche appliquée aux journaux de taille plus conséquents. Les tests ont démontré la viabilité de notre approche de correction des journaux logiciels.

Nos futurs travaux vont ajouter de nouvelles règles d'auto-organisation aux AT. Par exemple, certaines relations entre les AT seront supprimées, et les AT devront être en mesure de se réorganiser pour réparer les erreurs. Néanmoins, les règles devront être conçues de manière à maintenir la coopération entre les AIT et les AT. En effet, lorsque le TIA détecte une erreur, celle-ci peut provenir d'une erreur dans le journal ou dans le voisinage des TA.

Enfin, dans notre protocole expérimental certaines notions mériteraient d'être précisées, notamment la correspondance entre AIT et AT. Pour l'instant, une simple égalité Leibnizienne est utilisée. Pour finir, nous nous adressons le problème de la génération automatique de journaux bruités dans le prochain jalon.

References

[1] W. van der Aalst, "*Data Science in Action*", Springer Berlin Heidelberg, 2016.

[2] M. Abdelkafi, L. Bouzguenda, and F. Gargouri, "DiscopFlow: A new Tool for Discovering Organizational Structures and Interaction Protocols in WorkFlow", 2012.

[3] S. Astromskis, A. Janes, and M. Mairegger, "A process mining approach to measure how users interact with software: an industrial case study", Proceedings of the *International Conference on Software and System Process*, 2015.

[4] L. Cao, V. Gorodetsky, and P. A. Mitkas, "Agent Mining: The Synergy of Agents and Data Mining," *IEEE Intelligent Systems*, vol. 24, no. 3, pp. 64–72, 2009.

[5] E. Casalicchio and S. Tucci, "Public Administration Workflows Re-engineering : An Agent-Based & Approach", In : *Multiagent Systems and Applications*, Springer Berlin Heidelberg, 2013.

[6] M. Halaška and R. Šperka, "Advantages of application of process mining and agent-based systems in business domain", In *KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications*, pp. 177–186, 2018

[7] D. Hollingsworth, D. Hollingsworth, "The Workflow Reference Model: 10 Years On", In : *Fujitsu Services, UK; Technical Committee Chair of WfMC*, pp. 295–312, 2004.

[8] Gaël Clair, Elsy Kaddoum, Marie-Pierre Gleizes, Gauthier Picard. "Self-Regulation in Self-Organising Multi-Agent Systems for Adaptive and Intelligent Manufacturing Control", In *IEEE International Conference on Self-Adaptive and Self-Organizing Systems*, IEEE Computer Society, 2008.

[9] G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "*Self-organising software : from natural to artificial adaptation*". Springer, 2011.

[10] F. Mouysset et al., "Investigations of Process Mining Methods to discover Process Models on a Large Public Administration Software," *INFORSID*, 2019.

[11] R. Subramanyam and M. S. Krishnan, "Empirical analysis of CK metrics for object-oriented design complexity: implications for software defects", In *IEEE Transactions on software engineering*, vol. 29, no. 4, pp. 297–310, 2003.

[12] D. V. Thompson, R. W. Hamilton, and R. T. Rust, "Feature Fatigue: When Product Capabilities Become Too Much of a Good Thing", In *Journal of marketing research*, vol. 42, no. 4, pp. 431–442, 2005.

Résilience et auto-réparation de processus de décisions multi-agents

P. Rust[a] pierre.rust@orange.com
G. Picard[b] picard@emse.fr
F. Ramparany[a] fano.ramparany@orange.com

[a]Orange Labs, France
[b]Institut Henri Fayol, MINES Saint-Etienne, France

Résumé

Nous définissons la notion de k-résilience de graphes de calculs en support aux décisions d'agents opérées sur des systèmes dynamiques. Nous proposons une méthode d'auto-réparation de la distribution des calculs, DRPM[MGM-2], afin d'assurer la continuité des décisions collectives suite à la disparition d'agents, grâce au déploiement de réplicas de calculs. Nous nous intéressons ici à la réparation de processus d'optimisation sous contraintes, où les calculs sont des variables de décision ou des contraintes distribuées sur l'ensemble des agents. Nous évaluons expérimentalement les performances de DRPM[MGM-2] sur différentes topologies de systèmes opérant des algorithmes (Max-Sum ou A-DSA) pour résoudre des problèmes classiques (aléatoire, coloration de graphe, Ising) alors que des agents disparaissent.

Mots-clés : *DCOP, résilience, auto-réparation*

Abstract

We define the notion of k-resilience for computational graphs in support of agents' decisions on dynamic systems. We propose a method to self-repair the computation distribution, DRPM [MGM-2], as to ensure the continuity of collective decisions following the disappearance of agents, through the deployment of replicas calculations. We are interested here in constraint optimization process repair, where the computations are decision variables or distributed constraints. We experimentally evaluate the performance of DRPM[MGM-2] on different topologies of systems operating algorithms (Max-Sum or A-DSA) to solve classic problems (random, graph coloring, Ising) while agents disappear.

Keywords: *DCOP, resilience, self-repair*

1 Introduction

Nous examinons ici le problème de la répartition d'un ensemble de *calculs* étayant les décisions sur un ensemble d'agents incarnés dans des objets physiques (ou *nœuds*), comme des robots, des capteurs ou des véhicules autonomes.

Les décisions collectives et coordonnées sont organisées dans un graphe de calculs, où les sommets représentent des calculs et les arcs représentent une relation de dépendance entre les calculs. Une telle organisation est utilisée dans de nombreux modèles de décision, comme par exemple les graphes de facteurs et les graphes de contraintes utilisés lors de la résolution de problèmes d'optimisation distribuée sous contraintes (DCOP) [5], ou les algorithmes pour graphes de calculs tels que ceux adressés par Pregel ou d'autres frameworks basés sur BSP [8]. Bien que ces dernières structures ciblent généralement l'informatique en grappes hautes performances, nous considérons ici l'Internet des objets (IoT), des scénarios d'informatique embarquée ou des scénarios d'essaims de robots, où les calculs s'exécutent sur des nœuds distribués très hétérogènes et où une coordination centrale pourrait ne pas être souhaitable, voire même impossible [2].

Ici, les systèmes doivent pouvoir gérer les ajouts et les défaillances d'agents : lorsqu'un agent cesse de répondre, les autres agents du système doivent en assumer la responsabilité et exécuter les calculs partagés orphelins. De même, quand un nouvel agent est ajouté dans le système, il peut être utile de reconsidérer la distribution des calculs afin de tirer parti des capacités de calcul du nouveau venu, comme proposé dans [13]. Pour faire face à cette dynamique en maintenant les calculs et les décisions en cours alors que l'infrastructure évolue, une solution inspirée par les bases de données distribuées est la *réplication* [18, 17]. Nous proposons donc de répliquer les définitions de calculs plutôt que les données. Plus précisément, nous définissons la notion de *k-résilience*, qui caractérise les systèmes capables de fournir les mêmes fonctionnalités (ou prendre les mêmes décisions) même lorsque jusqu'à k noeuds disparaissent. L'adaptation de la distribution des calculs a été étudiée par [13], mais n'évalue aucune méthode distribuée en cours d'exécution. De plus, la disparition de plusieurs nœuds en même temps n'a pas non plus été prise en compte. [3] considère le problème distribution pour la résilience aux pannes, mais la méthode proposée s'appuie sur

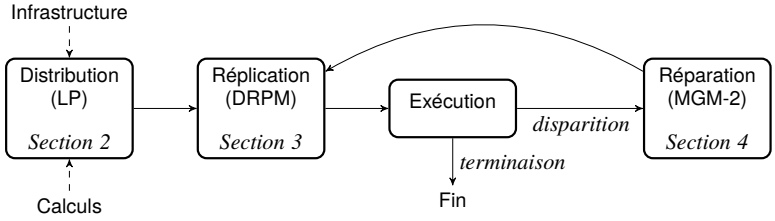


FIGURE 1 – Le cycle de vie de DRPM[MGM-2].

un agent *dispatcher* et reste donc partiellement centralisée. D'autre travaux, comme [6], ont étudié la réplication pour garantir tolérance aux pannes dans un SMA, mais répliquent en général des agents là ou nous proposons de répliquer les décisions, affectées aux agents.

L'article est structuré comme suit. La section 2 définit la notion de distribution optimale de graphes de calculs sur une infrastructure physique. La section 3 expose la notion de k-résilience et présente une méthode de recherche itérative distribuée, DRPM, pour déployer des réplicas afin d'assurer la k-résilience. Nous avons conçu une méthode de réparation distribuée, DRPM[MGM-2], basée sur un modèle DCOP utilisant la réplication pour adapter le déploiement de la décision à la suite de modifications apportées au système multi-agents physique (disparition des agents), dans la section 4. Nous nous intéresserons ensuite au cas particulier de la réparation des méthodes DCOP opérant sur des systèmes dynamiques, et proposons une méthode elle-même basée sur un DCOP pour réparer les solveurs DCOP. Le cadre général de notre approche est résumé dans la figure 1. Nous évaluons expérimentalement nos contributions algorithmiques sur différentes topologies de systèmes dont la fonctionnalité consiste à exécuter des processus de raisonnement sous contraintes distribuées pendant que les agents quittent le système, dans la section 5. Nous exécutons notamment Max-Sum et une version asynchrone de l'algorithme DSA (A-DSA) dans de telles configurations dynamiques. Enfin, nous concluons le document sur certaines perspectives.

2 Distribution des calculs

Le placement de calculs liés à des décisions sur des agents physiques peut voir un impact important sur les performances du système : certaines distributions peuvent améliorer le temps de réponse, d'autres favoriser la charge de communication entre les agents et d'autres peuvent améliorer d'autres critères tels que la qualité de service, les coûts d'exploitation, ou l'impact sur l'environnement.

Soit $G = \langle \mathbf{X}, D \rangle$ un graphe de calculs, où $\mathbf{X}$ est l'ensemble des calculs x_i, et D l'ensemble des arcs (i, j) représentant les dépendances entre calculs (qui impliquent l'envoi de messages). Soit $\mathbf{A}$ l'ensemble des agents pouvant héberger des calculs $x_i \in \mathbf{X}$. Notons $\mu : \mathbf{X} \rightarrow \mathbf{A}$ la fonction associant les calculs aux agents et $\mu^{-1}(a_m)$ l'ensemble des calculs hébergés par a_m. Un agent ne peut héberger qu'une quantité limitée de calculs, contrainte par la *capacité* $\mathbf{w_{max}}(a_m)$ de l'agent, et le *poids* du calcul, $\mathbf{w}(x_i)$. Notons x_i^m le booléen spécifiant si x_i est hébergé par a_m.

Définition 1. *Etant donnés un ensemble d'agents* $\mathbf{A}$ *et un ensemble de calculs* $\mathbf{X}$*, une* ***distribution*** *est une fonction* $\mu : \mathbf{X} \rightarrow \mathbf{A}$ *qui affecte chaque calcul à un agent exactement et qui satisfait les contraintes de capacité des agents.*

Trouver une distribution est un problème de satisfaction de contraintes avec :

$$\forall x_i \in \mathbf{X}, \quad \sum_{a_m \in \mathbf{A}} x_i^m = 1 \tag{1}$$

$$\forall a_m \in \mathbf{A}, \quad \sum_{x_i \in \mu^{-1}(a_m)} \mathbf{w}(x_i) \leq \mathbf{w_{max}}(a_m) \tag{2}$$

Lorsque les communications sont contraintes (e.g. IoT), la distribution devrait générer aussi peu de charge que possible et favoriser les liens les moins coûteux. Nous supposons que tous les agents peuvent communiquer entre eux, mais avec des coûts différents, représentés par une matrice : $\mathbf{route}(m,n)$ est le coût de communication entre a_m et a_n. Soit $\mathbf{msg}(i,j)$ la taille des messages entre x_i et x_j, le coût entre x_i sur a_m et x_j sur a_n est :

$$\forall x_i,x_j \in \mathbf{X}, \forall a_m,a_n \in \mathbf{A}, \mathbf{c_{com}}(i,j,m,n) = \begin{cases} \mathbf{msg}(i,j) \cdot \mathbf{route}(m,n) & \text{si } (i,j) \in D, m \neq n \\ 0 & \text{sinon} \end{cases} \tag{3}$$

où il n'y a aucun coût pour des calculs hébergés sur le même agent. Les coûts d'hébergement sur un agent sont modélisés par une fonction **host** affectant un coût pour chaque paire (a_m,x_j).

La qualité d'une distribution peut ainsi être évaluée par la fonction suivante, avec $\omega \in [0,1]$:

$$\omega \cdot \sum_{(i,j) \in D} \sum_{(m,n) \in \mathbf{A}^2} \mathbf{c_{com}}(i,j,m,n) \cdot x_i^m \cdot x_j^n \tag{4}$$

$$+ (1-\omega) \cdot \sum_{(x_i,a_m) \in \mathbf{X} \times \mathbf{A}} x_i^m \cdot \mathbf{c_{host}}(a_m, x_i) \quad (5)$$

Définition 2. *Une **distribution optimale** est une distribution μ qui minimise le coût des communications entre agents et le coût d'hébergement des calculs.*

Affecter les calculs aux agents de telle sorte peut se faire par la résolution d'un programme linéaire en linéarisant (4) et (5) comme des objectifs à minimiser, (2) comme des contraintes, et en ajoutant des contraintes afin que chaque calcul ne soit affecté qu'à un unique agent. Ce modèle est plus général que [13], dédié aux graphes de facteurs sans coût d'hébergement. Etant NP-difficile, il peut facilement mettre à mal les solveurs commerciaux modernes. Il peut cependant être utilisé pour initialiser le système pour calculer de manière centralisée la distribution initiale de petites instances. Dans nos expérimentations nous l'utilisons pour évaluer la qualité des distributions obtenues par nos techniques.

3 Résilience et réplication

Dans un contexte centralisé, lorsque certains agents échouent, on peut utiliser le programme linéaire discuté dans la section précédente pour recalculer une nouvelle distribution optimale. Cependant, accéder à un tel calcul centralisé peut ne pas être possible ou souhaitable, en fonction des exigences du scénario d'application. Ainsi, nous étudions des techniques décentralisées pour faire face à une telle dynamique dans l'infrastructure.

3.1 k-résilience

Nous définissons la notion de k-résilience comme la capacité d'un système à se réparer et à fonctionner correctement même lorsque jusqu'à k agents disparaissent. Cela signifie qu'après une période de récupération, tous les calculs doivent être actifs sur exactement un agent et communiquer les uns avec les autres comme spécifié par le graphe $\mathbf{G}$.

Définition 3. *Etant donnés les agents $\mathbf{A}$, les calculs $\mathbf{X}$, et une distribution μ, un système est k-**résilient** si pour tout $F \subset \mathbf{A}, |F| \leq k$, une nouvelle distribution $\mu' : X \rightarrow \mathbf{A} \setminus F$ existe.*

Une condition préalable à la k-résilience est de toujours avoir accès à la définition de chaque calcul après disparition. Une approche consiste à conserver k *réplicas* (copies de définitions) de chaque calcul actif sur différents agents. À condition que les k réplicas soient placés sur différents agents, il y aura toujours au moins un réplica restant après l'échec, comme en bases de données distribuées [8]. Ici, nous appliquons ces idées sauf que nous conservons des réplicas de définitions de calculs au lieu de données, ce qui implique que les calculs doivent être *stateless* ou que leur l'état doit être restaurable. La valeur maximale de k pour laquelle la k-résilience peut être atteinte dépend du système et surtout sur des capacités des agents. De plus, la k-résilience du système réparé devrait être restaurée, tant qu'il y a suffisamment de nœuds disponibles.

3.2 Placement des réplicas

Le problème de l'affectation de réplicas à des hôtes peut être considéré comme un problème d'optimisation, proche de la définition 2. Idéalement, nous devrions optimiser l'emplacement des réplicas pour les coûts de communication et d'hébergement. Cela garantirait que lorsque des agents échouent, des réplicas sont disponibles sur les bons agents candidats. Cependant, l'espace de recherche pour cette optimisation est extrêmement large. Dans un système k-résilient avec n agents, il y a $\sum_{0<i\leq k} \binom{n}{i}$ scénarios de défaillance potentiels (jusqu'à k agents sur n peuvent échouer simultanément). Avec m calculs, le nombre de *configurations de réplications* possibles est $m \cdot \binom{n}{k}$ Le problème de la distribution optimale des réplicas sur un ensemble d'agents ayant des coûts et des capacités différentes peut être transformé en un problème de sac à dos multiple quadratique (QMKP) [14], qui est NP-difficile.

Ensuite, pour chacune de ces configurations de réplicas, il existe m^k *configurations d'activations* (pour chaque calcul orphelin, activer l'un des k réplicas). En supposant que nous puissions calculer le coût de toutes ces configurations, il ne serait toujours pas évident de savoir quel placement de réplicas serait le meilleur : on pourrait envisager celui permettant la meilleure configuration d'activation, ou celui permettant, en moyenne, des configurations d'activation de bonne qualité ou même celle offrant les meilleures configurations d'activation sur l'ensemble des scénarios de défaillance possibles. Évidemment, la définition de l'optimalité pour le placement de réplicas dépend très fortement du problème. Par conséquent, étant donné cette complexité, nous optons pour une approche heuristique distribuée, décrite dans la section suivante.

3.3 Méthode distribuée de placement

Nous proposons ici une méthode distribuée, DRPM, pour déterminer les hôtes des k réplicas d'un calcul x_i donné. DRPM est une version distribuée de *iterative lengthening* (recherche de coût uniforme en fonction du coût des chemins) avec une mémorisation de chemins minimaux pour trouver les k meilleurs chemins. L'idée est d'hé-

berger des réplicas sur les voisins les plus proches en ce qui concerne les coûts de communication et d'hébergement et les contraintes de capacité, en effectuant une recherche dans un graphique induit par des dépendances de calcul. Il génère une distribution de k réplicas (et le coût des chemins d'accès vers leurs hôtes) avec des coûts minimum pour un ensemble d'agents interconnectés. S'il est impossible de placer les réplicas k, en raison de contraintes de mémoire, DRPM place autant de calculs que possible et génère le meilleur niveau de résilience possible. Un agent d'hébergement, appelé *initiateur*, demande *itérativement* à chacun de ses voisins les moins chers, par ordre de coût croissant, jusqu'à ce que tous les réplicas soient placés. Les hôtes candidats sont considérés de manière itérative par ordre croissant de coût, qui comprend à la fois le coût de la communication (tout au long du chemin entre le calcul initial et son réplica) et le coût d'hébergement de l'agent hébergeant le réplica.

Définissons tout d'abord le graphe modélisant les coût de communication qui sera développé lors de la recherche :

Définition 4. *Etant donné un graphe* $\langle \mathbf{X}, D \rangle$*, le* **route***–graphe est un graphe pondéré* $\langle \mathbf{A}, E, w \rangle$ *où* $\mathbf{A}$ *est l'ensemble des nœuds,* E *est l'ensemble des arcs avec* $E = \{(a_m, a_n) | \exists (x_i, x_j) \in D, \text{and } \mu(x_i) = a_m, \mu(x_j) = a_n\}$ *et* $w : E \to \mathbb{R}$ *est la fonction de pondération* $w(a_m, a_n) = \mathbf{route}(m, n)$.

Afin de prendre aussi en compte l'hébergement, étendons le **route**–graphe avec des nœuds supplémentaires attachés à chaque agent, excepté l'agent initiateur, par un arc pondéré par le coût d'hébergement, comme dans la figure 2.

Définition 5. *Étant donné le* **route***–graphe* $\langle \mathbf{A}, E, w \rangle$ *et un calcul* x_i*, le* **route+host***–graphe est un graphe pondéré* $\langle \mathbf{A}', E', \mathbf{cost} \rangle$ *où* $\mathbf{A}' = \mathbf{A} \cup \widetilde{\mathbf{A}}$ *est l'ensemble des nœuds,* $\widetilde{\mathbf{A}} = \{\tilde{a}_m | a_m \in \mathbf{A}, a_m \neq \mu(x_i)\}$ *est l'ensemble des nœuds supplémentaires (un pour chaque élément de* $\mathbf{A}$*),* $E' = E \cup \{(a_m, \tilde{a}_m) | \tilde{a}_m \in \mathbf{A}\}$ *est l'ensemble des arcs et* $\mathbf{cost} : E' \to \mathbb{R}$ *est la fonction de pondération t.q.* $\forall a_m, a_n \in \mathbf{A}$, $\mathbf{cost}(a_m, a_n) = w(a_m, a_n)$, $\forall \tilde{a}_m \in \widetilde{\mathbf{A}}$, $\mathbf{cost}(a_m, \tilde{a}_m) = \mathbf{c_{host}}(a_m, x_i)$.

Un **route+host**–graphe est un graphe de recherche, développé à l'exécution et exploré pour un calcul particulier x_i. Chaque agent exécute autant de DRPM que de calculs à répliquer sur plusieurs **route+host**–graphes. Pour un **route+host**–graphe donné, chaque agent peut encapsuler deux sommets (un dans $\mathbf{A}$ et son image dans $\widetilde{\mathbf{A}}$) et peut recevoir des messages concernant leurs deux sommets, et même des messages réflexifs. En outre,

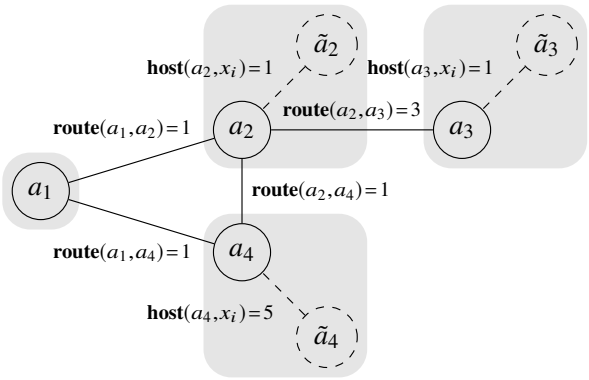


FIGURE 2 – Un **route+host**–graphe avec 4 agents

lors de l'évaluation de la possibilité pour un agent d'héberger un réplica pour x_i, nous nous assurons qu'il accepte uniquement s'il dispose d'une capacité suffisante pour activer tout sous-ensemble de taille k de ses réplicas. Bien sûr, cette contrainte est plus forte que ce qui pourrait être réellement nécessaire, ainsi, cette distribution n'est pas optimale en ce qui concerne les coûts d'hébergement, puisqu'un agent peut refuser d'héberger un calcul alors qu'il peut disposer de suffisamment de mémoire, au final. Comme de multiples instances de DPRM sont exécutées simultanément, une par calcul à répliquer, l'algorithme peut entraîner une distribution sous optimale, y compris en terme de communication. Cependant, si le placement de réplicas ne concerne qu'un seul calcul, la distribution est optimale puisque notre algorithme implémente une stratégie de recherche itérative par rallongement, ou *iterative lengthening*, avec mémorisation des chemins (pour éviter les boucles) [12, p.90].

Exemple 1. *La figure 2 représente un* **route+host***–graphe avec 4 agents (gris), où* a_1 *cherche à répliquer* x_i*. Pour* $k = 2$*, DRMP place des réplicas sur* a_2 *(coût de* $1+1=2$*) et sur* a_3 *(coût de* $1+3+1=5$*) s'il y a assez de capacité sur ces agents, comme le chemin minimal pour héberger sur* a_4 *est plus élevé (*$1+5=6$*).*

DRPM, étant une version distribuée de cet algorithme, utilise deux types de messages (`REQUEST` et `ANSWER`) avec les mêmes champs : (i) `current` : chemin de la requête, i.e. une liste contenant tous les sommets par lesquels les messages ont été transmis depuis l'initiateur jusqu'à celui qui reçoit le message actuel, (ii) `budget`, `spent` : budget restant pour l'exploration du graphe et budget déjà dépensé sur le chemin courant, (iii) `known` : tableau associatif affectant le coût aux chemins déjà découverts à des sommets non visités, qui comptabilise les chemins les moins chers jusqu'à présent, (iv) `visited` : liste des sommets déjà visités, (v) `k` : le nombre restant de réplicas à héberger, (vi) x_i : calcul à répliquer. Les messages

REQUEST descendent le long du graphe depuis l'initiateur, et correspondent au développement du graphe. Les messages ANSWER remontent les solutions (s'il y en a) jusqu'à l'initiateur.

Au début, l'agent nécessitant une réplication de calcul initialise known avec les chemins de ses voisins directs dans le **route**+**host**–graphe et envoie un message REQUEST avec un budget égal au chemin le moins cher connu. Ensuite, les agents traitent les messages comme expliqué dans le paragraphe suivant. Le protocole se termine lorsque toutes les réplicas possibles ont été placés (au plus k).

A la réception d'un message REQUEST, soit l'agent peut héberger un réplica et diminue ainsi le nombre de réplicas à placer, soit il transmet la demande à d'autres agents voisins. Dans le premier cas, si tous les réplicas ont été placés, l'agent répond à son prédécesseur avec un message ANSWER. Pour rechercher d'autres agents pour héberger des réplicas, s'il existe un chemin de coût minimum connu commençant par le chemin actuellement exploré qui est accessible avec le budget actuel, l'agent transmet la demande à son successeur dans ce chemin (avec un coût et un budget mis à jour). S'il n'y a pas un tel chemin, l'agent remplit le tableau known avec de nouveaux chemins menant à ses voisins dans le **route**+**host**–graphe, si ces derniers améliorent les chemins connus existants, et renvoie known avec un message ANSWER à son prédécesseur pour qu'il explore ces nouvelles possibilités.

A la réception d'un message ANSWER, le message peut soit notifier que tous les réplicas ont été placés, soit qu'il reste au moins un réplica à placer. Dans le premier cas, si l'agent est l'initiateur, il termine l'algorithme en plaçant tous les réplicas demandés, sinon il renvoie la réponse à son prédécesseur, jusqu'à ce qu'il atteigne l'initiateur. Dans ce dernier cas, si l'agent est l'initiateur, il augmente le budget et envoie une demande au voisin le plus proche le cas échéant ; sinon cela signifie qu'il n'y a plus de chemin à explorer et que tous les réplicas ne peuvent être placés : l'agent termine l'algorithme. Si l'agent n'est pas l'initiateur, mais qu'il existe un chemin accessible dans le budget actuel, il demande la réplication à son successeur dans le meilleur chemin connu, comme lors de la gestion des messages REQUEST. Enfin, s'il n'existe pas de tel chemin, il transmet simplement la réponse à son prédécesseur dans le chemin actuel.

Exemple 2. *Dans le cas de la figure 2, voici la séquence de messages qui sera générée. a_1, l'initiateur, peut envisager deux chemins $[a_1 \rightarrow a_2]$ et $[a_1 \rightarrow a_4]$ de coût identique, et initialise donc un budget de* 1. *a_1 envoie un message* REQUEST *à a_2, son premier successeur dans le premier chemin. a_2 répond avec un message* ANSWER *contenant deux nouveaux chemins $[a_1 \rightarrow a_2 \rightarrow \tilde{a}_2]$ de coût* 2, *et $[a_1 \rightarrow a_2 \rightarrow a_3]$ de coût* 4. *Le chemin $[a_1 \rightarrow a_2 \rightarrow a_4]$ de coût* 2 *n'est pas considéré car le meilleur chemin connu dans* known *vers a_4 est de coût* 1. *a_1, à la réception du message a_2 va explorer le meilleur chemin en envoyant un message* REQUEST *à a_4. a_4 rajoute un chemin à* known, *$[a_1 \rightarrow a_4 \rightarrow \tilde{a}_4]$ de coût* 6 *et répond à a_1. a_1 va ainsi envoyer un message* REQUEST *à a_2 pour explorer le meilleur chemin. a_2 peut héberger le réplica, car ce chemin même à une feuille $\tilde{a}_2$. a_2 peut même continuer l'exploration, car le prochain chemin dans* known *passe par lui et mène à a_3. a_2 envoie donc un message* REQUEST *à a_3, mais cette fois le nombre de réplicas à placer n'est plus de* 2 *mais de* 1. *a_3 peut rajouter le chemin $a_1 \rightarrow a_2 \rightarrow a_3 \rightarrow \tilde{a}_3$, qui est le meilleur actuel, et donc répondre qu'il peut héberger le dernier réplica. a_3 envoie donc un message* ANSWER *à a_2, qui le transmet à a_1. A la réception, comme tous les réplicas ont été placés, a_1 termine le placement.*

Globalement, chaque agent est chargé de placer k réplicas de tous les calculs actifs qu'il héberge actuellement et exécute donc DRPM une fois pour chacun de ses calculs actifs. Ces multiples exécutions de DRPM peuvent être faites de manière séquentielle ou simultanée mais leur résultat dépend de l'ordre de réception des messages. Notez cependant que même si vous exécutez plusieurs DRPM simultanément, un agent ne dispose que d'une seule file de messages et traite les messages entrants de manière séquentielle, ce qui l'empêche d'accepter des réplicas qui dépasseraient sa capacité.

Théorème 1. *DRPM se termine.*

Démonstration. Pour k = 1, étant donné que les coûts du DRPM sont additifs et monotones et qu'il enregistre les chemins d'accès vers les sommets non consultés, il se termine comme un algorithme *iterative lengthening* classique, avec le chemin de coût minimum ou le chemin vide s'il n'y a pas assez de mémoire dans les agents pour héberger le calcul x_i. Pour $k > 1$, DRPM tente de placer chaque réplica de manière séquentielle. Il recherche d'abord le meilleur chemin (comme pour $k = 1$), puis exécute le même processus pour un deuxième meilleur chemin, et ainsi de suite jusqu'à ce que (i) les k réplicas soient placés ou (ii) il n'y ait pas assez de mémoire pour héberger le $n^{\text{ième}}$ réplica . La mémorisation dans visited garantit que le même chemin ne sera pas pris en compte deux fois. Ainsi, des itérations de recherche consécutives génèrent des chemins différents avec des coûts de chemin augmentant. Ainsi, dans le cas (i), DRPM se termine lorsque k réplicas ont été placés sur les k meilleurs hôtes ; et

dans le cas (ii), il se termine lorsque $k' < k$ réplicas ont été placés, où k' est le nombre maximal de réplicas pouvant être placés. □

4 Méthode d'auto-réparation

Une fois la réplication abordée, nous modélisons maintenant le problème de réparation en tant que problème d'optimisation distribuée sous contraintes (DCOP) [7], à résoudre par les agents eux-mêmes, à la suite de d'apparitions ou disparitions dans le système, afin d'activer des réplicas ou d'améliorer la qualité de la distribution des calculs.

4.1 Formulation DCOP

On note X_c l'ensemble des calculs candidats x_i qui peuvent ou doivent être déplacés lorsque l'ensemble des agents change. Pour chacun de ces calculs, notons A_c^i l'ensemble des agents candidats pour accueillir x_i. L'ensemble de tous les agents candidats, indépendamment des calculs, est noté $A_c = \cup_{x_i \in X_c} A_c^i$ et X_c^m désigne l'ensemble des calculs que l'agent a_m pourrait héberger. Décider quel agent $a_m \in A_c$ héberge chaque calcul $x_i \in X_c$ peut être traduit en un problème d'optimisation similaire à celui présenté dans la section 2, limitée à A_c et X_c. contrainte Pour s'assurer que chaque calcul candidat est hébergé sur exactement un agent, nous réécrivons les contraintes (1) pour chaque $x_i \in X_c$:

$$\sum_{a_m \in A_c^i} x_i^m = 1 \tag{6}$$

De même, les contraintes de capacité (2) peuvent être reformulées comme suit :

$$\sum_{x_i \in X_c^m} \mathbf{w}(x_i) \cdot x_i^m + \sum_{x_j \in \mu^{-1}(a_m) \setminus X_c} \mathbf{w}(x_j) \leq \mathbf{w_{max}}(a_m) \tag{7}$$

L'objectif de coût d'hébergement dans (5) peut être formulé de manière similaire à l'aide d'une contrainte souple pour chaque agent candidat a_m :

$$\sum_{x_i \in X_c^m} \mathbf{c_{host}}(a_m, x_i) \cdot x_i^m \tag{8}$$

Enfin, les coûts de communication dans (4) sont représentés par un ensemble de contraintes souples. Pour un agent a_m, les frais de communication liés à l'hébergement d'un calcul x_i peut être formulé comme la somme des coût des arcs coupés (x_i, x_j) à partir du graphe de calculs $\langle \mathbf{X}, D \rangle$, (c'est-à-dire où $\mu^{-1}(x_j) \neq a_m$). Notons N_i les voisins de x_i dans le graphe de calcul. Quand un voisin x_n n'est pas un calcul candidat (c'est-à-dire qu'il ne sera peut-être pas déplacé et que $x_m \in N_i \setminus X_c$), le coût de communication de l'arc correspondant est simplement donné par $\mathbf{c_{com}}(i, j, m, \mu^{-1}(x_m))$. Pour les voisins qui pourraient être déplacés, le coût de la communication dépend de l'agent candidat choisi pour l'héberger, $\sum_{a_n \in A_c^j} x_j^n \cdot \mathbf{c_{com}}(i, j, m, n)$. Avec cela, nous pouvons écrire la contrainte souple de coût de communication pour l'agent a_m :

$$\sum_{(x_i, x_j) \in X_c^m \times N_i \setminus X_c} x_i^m \cdot \mathbf{c_{com}}(i, j, m, \mu^{-1}(x_j)) + \sum_{(x_i, x_j) \in X_c^m \times N_i \cap X_c} x_i^m \cdot \sum_{a_n \in A_c^j} x_j^n \cdot \mathbf{c_{com}}(i, j, m, n)) \tag{9}$$

Nous pouvons maintenant formuler le problème de réparation en tant que DCOP $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, C, \mu \rangle$ où $\mathcal{A}$ est l'ensemble des agents candidats A_c, $\mathcal{X}$ et $\mathcal{D}$ sont respectivement l'ensemble des variables de décision x_i^m et leur domaine $\{0,1\}$ et C est composé de contraintes (6), (7), (8) et (9) appliquées pour chaque agent $a_m \in A_c$. (6) et (7) entraînent des coûts infinis en cas de violation, alors que (8) et (9) définissent directement les coûts à minimiser. La fonction μ affecte chaque variable x_i^m à un agent a_m.

4.2 Réparer avec MGM-2

Maintenant que notre problème de réparation a été exprimé en tant que DCOP, nous discutons de sa résolution en utilisant un algorithme DCOP. Plusieurs méthodes existent, telles que les algorithmes de recherche [7, 9] et les algorithmes d'inférence [10, 16, 5], pour en citer quelques-uns. En bref, en utilisant ces protocoles d'envoi de messages (synchrones ou non), les agents se coordonnent pour attribuer des valeurs à leurs variables.

Dans notre cas, nous optons pour une méthode légère, rapide et itérative, à savoir MGM [7]. Chaque agent assigne d'abord des valeurs aléatoires à ses variables et envoie les informations à tous ses voisins. En utilisant toutes les valeurs des voisins, un agent calcule le gain maximal s'il modifie sa valeur et l'envoie à tous ses voisins. Ensuite, en utilisant les gains de tous les voisins, l'agent modifie sa valeur si son gain est le plus grand. Ce processus se répète jusqu'à ce qu'une condition de terminaison soit remplie. Dans notre cas, les décisions nécessitent une coordination entre deux agents : pour déplacer un calcul de l'agent a_m à a_p, la variable binaire x_i^m doit prendre la valeur 0, tandis que *simultanément*, x_i^p doit passer de 0 à 1. Ce besoin de modifications simultanées justifie l'utilisation de MGM-2 [7]. La propriété de monotonie de MGM-2 répond très bien à nos

besoins : une fois les contraintes dures (6) et (7) satisfaites, elles ne seront plus violées, tout en optimisant les contraintes souples (8) et (9).

En combinaison avec notre méthode de placement de réplicas, nous obtenons une méthode de réparation, appelée DRPM[MGM-2], qui peut être utilisée pour s'adapter au départ et à l'arrivée d'agents, en utilisant les définitions appropriées des ensembles A_c et X_c. Discutons le cas de départ d'agent, car c'est la situation la plus stressante. En supposant que le déploiement initial et le placement des réplicas aient été effectués au moment du démarrage du système, celui-ci exécutera le cycle de réparation suivant tout au long de son cycle de vie (voir la figure 1) : (a) Détecter départ / arrivée ; (b) Activer les réplicas des calculs manquants (avec MGM-2) ; (c) Placer les nouveaux réplicas pour les calculs manquants (à l'aide de DRPM) et poursuivre le fonctionnement nominal.

L'étape (a) suppose des mécanismes de découverte et de *keep alive* qui informent automatiquement certains agents de tout événement dans l'infrastructure. Ainsi, lorsqu'un agent a_m échoue ou est supprimé, nous considérons que tous les agents voisins de a_m dans le **route**–graphe sont informés du départ. L'étape (b) déplace les calculs hébergés par les agents disparus A_d à d'autres agents. Les calculs candidats sont les calculs orphelins hébergés sur ces agents : $X_c = \cup_{a_m \in A_d} \mu(a_m)$. Pour éviter tout délai supplémentaire et toute communication lors de la phase de réparation, ces calculs orphelins doivent être affectés à des agents disposant déjà des informations nécessaires pour exécuter le calcul. Cela signifie que l'agent candidat pour un calcul orphelin x_i affecte l'ensemble d'agents encore disponibles hébergeant une réplique pour ce calcul : $A_c^i = \rho(x_i) \backslash A_d$. Dans un système k-résilient, tant que $|A_d| \leq k$, nous sommes sûrs qu'il y aura toujours au moins un agent dans A_c^i. Ainsi, l'étape (b) donne une affectation de chacun des calculs orphelins à l'un des agents hébergeant son réplica. L'étape (c) maintient un bon niveau de résilience dans le système en réparant la distribution des réplicas en utilisant DRPM sur un problème plus petit, car de nombreux réplicas sont déjà placés.

5 Évaluation expérimentale

Analysons maintenant l'impact de la réparation sur les performances de processus de raisonnements sous contraintes – Max-Sum, un algorithme d'inférence approché [5] et A-DSA, une version asynchrone de l'algorithme de recherche locale DSA [19] – ainsi que la qualité des distributions obtenues après réparation.

5.1 Cadre expérimental

Nous exécutons les expériences en utilisant la plateforme multi-thread pyDCOP [1]. Pour évaluer notre cadre de réparation, nous générons des instances définies par trois composants : une définition de problème DCOP, une topologie multi-agents (l'infrastructure), et un scénario de perturbation.

Nous étudions 3 types de DCOPs : (i) coloration de graphes aléatoires avec densité $p = 0.3$, (ii) coloration de graphes *scale free* [1], et (iii) modèles d'Ising. Pour les colorations de graphes, chaque nœud est associé à une variable et chaque arc est associé à une contrainte souple dont le coût de chaque paire de valeurs possibles est tiré aléatoirement et uniformément dans $U[0-9]$. Le modèle d'Ising est une référence largement utilisée en physique statistique ; nous utilisons ici les mêmes paramètres que [15]. Nous générons une grille toroïdale régulière et associons chaque nœud à une variable binaire x_i et chaque arc à une contrainte binaire dont la fonction de coût r_{ij} est déterminée en échantillonnant d'abord une valeur k_{ij} à partir d'une distribution uniforme $U[-1{,}6{,}1{,}6]$, et ensuite assignons $r_{ij}(x_i,x_j) = -k_{ij}$ si $x_i = x_j$, k_{ij} sinon. De plus, chaque variable a une contrainte unaire dont la fonction de coût r_i est déterminée en échantillonnant k_i à partir d'une distribution uniforme $U[-0{,}05{,}0{,}05]$ puis en affectant $r_i(0) = k_i$ et $r_i(1) = -k_i$.

Selon le processus choisi (Max-Sum ou A-DSA), le DCOP est encodé dans un graphe de facteur (FG, pour Max-Sum) ou un graphe de contraintes (CG, pour A-DSA). Pour un même problème, $\langle \mathbf{A}, \mathcal{X}, \mathcal{D}, C, \mu \rangle$, il faut placer soit $|\mathbf{V}| = |\mathcal{X}| + |C|$ calculs pour un FG ou $|\mathbf{V}| = |\mathcal{X}|$ calculs pour un CG.

Une fois le graphe de calcul généré, nous générons deux infrastructures multi-agents différentes : uniforme et dépendante du problème. Une infrastructure est composée d'agents $|\mathbf{A}|$, chacun contenant une variable de décision ($|\mathbf{A}| = |\mathcal{X}|$), et est définie par $\mathbf{c_{host}}$, **route**, $\mathbf{w_{max}}$, $\mathbf{w}$ et **msg**. L'infrastructure uniforme considère les systèmes où les coûts de communication sont uniformes : $\forall a_m, a_n, \mathbf{route}(a_m, a_n) = 1$. Dans le cas dépendant du problème, les coûts $\mathbf{route}(a_m, a_n)$ sont définis de manière à respecter la structure du graphe de calcul : les agents ayant plusieurs voisins ont une faible communication tandis que les agents ayant peu de voisins ont un coût de communication plus élevé, comme dans de nombreuses infrastructures physiques (e.g. l'IoT). Plus précisément, $\mathbf{route}(a_m, a_n) = \frac{1+||N(a_m)|-|N(a_n)||}{|N(a_m)|+|N(a_n)|}$ où $|N(a_i)|$ est le nombre de voisins de a_i dans le graphe de calculs. Dans tous les cas, nous définissons

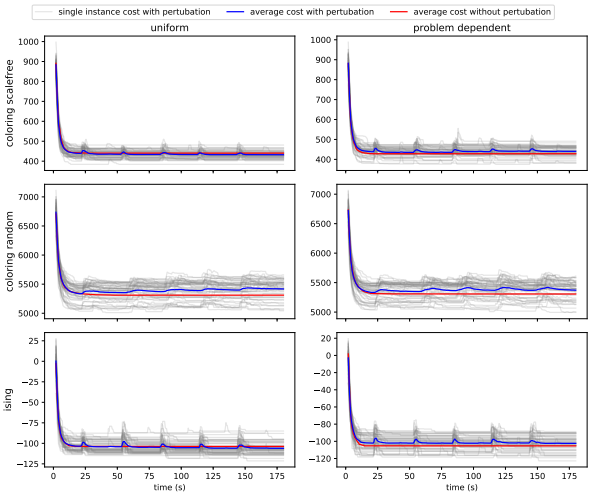
1. https://github.com/Orange-OpenSource/pyDcop

(i) $\mathbf{c_{host}}(a_m, x_j) = 0$ si le calcul x_j est initialement hébergé par l'agent a_m, $\mathbf{c_{host}}(a_m, x_j) = 10$ sinon ; (ii) la capacité de chaque agent dépend du poids de sa variable de décision et est mis à une grande valeur, pour que tous les réplicas puissent être hébergés et que la k-résilience soit possible, même après plusieurs réparations : $\mathbf{w_{max}}(a_i) = 100 * \mathbf{w}(x_i)$; (iii) finalement, $\mathbf{w}$ et $\mathbf{msg}$ dépendent du processus utilisé. Pour Max-Sum, la taille des messages est une fonction directe de la taille de domaine de la variable : $\mathbf{msg}(i,j) = |D_j| = 10$ et le poids des calculs de variables et de facteurs est respectivement proportionnel à la taille du domaine de la variable et la somme de la taille des domaines des variables liées. Pour A-DSA, $\mathbf{msg}(i,j) = 1$ et $\mathbf{w}(x_i) = |N(a_m)|$. Initialement, chaque variable de décision est affectée à un agent, et dans le cas de FG, les facteurs sont placés de manière optimale à l'aide du LP décrit dans la section 2. Nous utilisons $\omega = 0.5$ (les coûts d'hébergement et de communication sont également pris en compte).

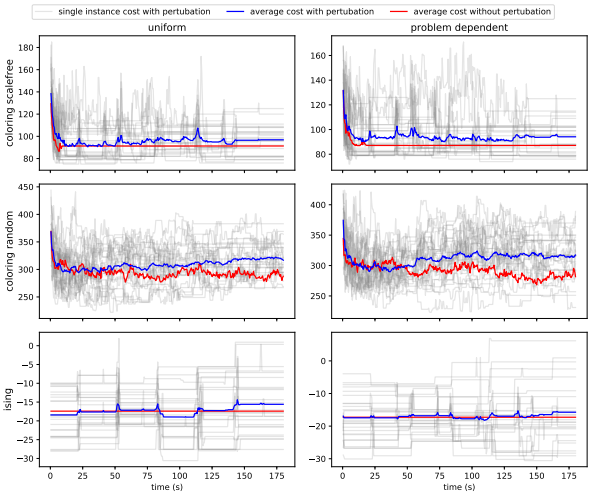
Nous générons des scénarios de perturbation sous forme de séquences d'événements toutes les 30 secondes, à partir de $t = 20s$. A chaque événement, k agents choisis au hasard disparaissent afin d'analyser l'impact sur les processus DCOP, et d'observer la k-résilience de notre système. Nous générons 20 instances (infrastructure et problème), et exécutons le scénario 5 fois pour chaque instance. De plus, nous résolvons les mêmes problèmes (5 exécutions pour chacune des 20 instances) sans aucune perturbation, afin d'évaluer l'impact de nos méthodes de réparation sur la qualité de la solution renvoyée par A-DSA ou Max-Sum. Dans les deux cas, les résultats sont moyennés sur toutes les instances. Les expériences sont effectuées sur un processeur Intel Core i7 avec 16 Go de RAM.

5.2 Impact de la réparation sur A-DSA

Regardons l'état d'exécution d'A-DSA actuel avec et sans perturbations, et analysons comment DRMP[MGM-2] modifie le fonctionnement du processus en cours d'exécution. Nous générons des problèmes avec $|\mathbf{A}| = |\mathcal{X}| = 100$ et utilisons $k = 3$. La figure 3a montre le coût de la solution trouvée par A-DSA au fil du temps. Le coût de chacune des 100 exécutions est affiché en gris transparent, et la forme globale illustre le fait que le comportement du système est cohérent dans les différentes instances. Nous pouvons voir que les solutions avec perturbations se dégradent lorsque les agents sont supprimés, mais rapidement améliorées à nouveau lorsque le système récupère. Ici, les réplicas activés par la réparation, par opposition aux calculs hébergés sur des agents supprimés, n'ont pas besoin des connaissances accumulées pour retrouver un état cohérent, grâce aux messages passés entre voisins, par nature dans A-DSA. En effet, les



(a) A-DSA



(b) Max-Sum

FIGURE 3 – Coût des solutions de Max-Sum (a) et A-DSA (b) en cours d'exécution, avec (bleu) et sans perturbations (rouge), sur des infrastructures uniformes (gauche) et dépendantes du problème (droite), et sur coloration *scale free* (haut), coloration aléatoire (milieu), et Ising (bas).

calculs sont ici `stateless`, comme l'exige notre approche de la k-résilience : ils collectent de nouvelles informations sur les coûts auprès de leurs voisins à chaque échange de messages.

La période de récupération est plus courte sur les modèles *scale free* et Ising. En effet, les problèmes de coloration des graphes sont plus denses et plus difficiles à résoudre, et leur réparation nécessite plus de messages et de temps. Pendant la même durée, les agents qui résolvent et réparent les processus de coloration des graphes doivent gérer davantage de messages spécifiques à la réparation (MGM-2 et DRPM). Ils gèrent donc moins de messages A-DSA pour améliorer le coût de la

solution. C'est la même raison pour laquelle le coût global est plus élevé dans une infrastructure dépendant du problème : les processus d'activation et de placement de réplicas nécessitent plus de messages. Nous avons évalué le temps moyen nécessaire pour réparer une distribution à moins de 3 secondes. Dans certains contextes, le coût moyen avec perturbation est inférieur au coût moyen sans perturbation. En fait, parfois, supprimer certains calculs et les transférer à d'autres agents, oubliant ainsi des informations sur le voisinage passé, peut extraire A-DSA de certains optima locaux. Nous pouvons également observer un décalage dans le temps pour la réparation du système. Cela est principalement dû à la latence de traitement des messages accumulés.

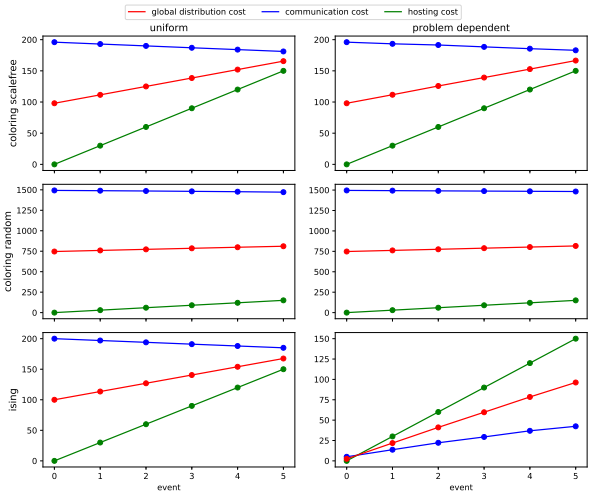
5.3 Impact de la réparation sur Max-Sum

Ici, nous considérons des problèmes plus petits, car Max-Sum fonctionne sur des graphes de facteurs nécessitant plus de calculs (un de plus par arête dans le graphe) à distribuer que les graphes de contraintes utilisés par A-DSA. Par conséquent, nous considérons $|\mathbf{A}| = |\mathcal{X}| = 25$, et $k = 2$. Néanmoins, sur un graphe aléatoire avec une densité de 0.3, de tels problèmes nécessitent en moyenne $25+0.3\frac{25\times24}{2} = 125$ calculs à gérer.

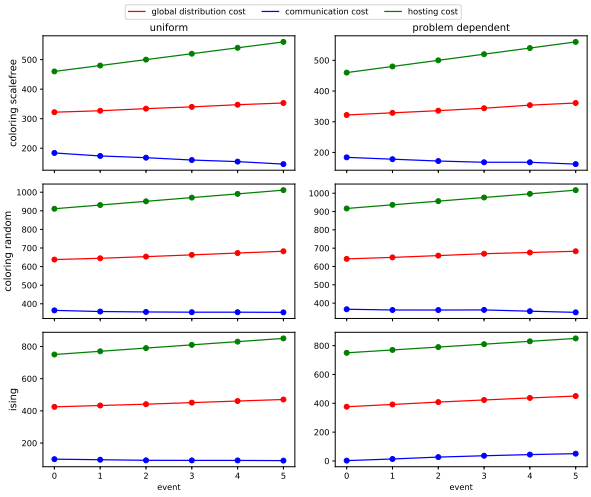
Dans la figure 3b nous pouvons voir que les solutions sur le système perturbé se dégradent lorsque les agents sont supprimés, mais s'améliorent à nouveau lorsque le système récupère, comme pour A-DSA. Cependant, le fonctionnement Max-Sum sur des problèmes très cycliques tels que la coloration aléatoire est connu pour être très bruitée, même en utilisant un facteur d'amortissement élevé (nous utilisons ici 0.8), comme proposé dans [4]. En outre, dans les algorithmes de propagation de croyances comme Max-Sum, les calculs ne sont pas vraiment sans état : ils accumulent des informations sur les contraintes et les préférences de leurs voisins. Lors de l'activation d'un réplica, le nouveau calcul actif recommence à nouveau et un nombre indéterminé de tours de messages sont nécessaires pour restaurer ces informations. Globalement, le fonctionnement Max-Sum est davantage impacté par la procédure de perturbations et de réparation qu'A-DSA.

5.4 Qualité des distributions réparées

Pour évaluer la qualité des distributions de calculs réparées, nous mesurons la dégradation de la distribution tout au long de la vie du système. A chaque événement, nous évaluons le coût de la distribution actuelle des graphes de contraintes (pour A-DSA) et des graphes de facteurs (pour Max-Sum) à l'aide des équations 4 et 5, par rapport au coût de distribution initial (qui est optimal,



(a) A-DSA



(b) Max-Sum

FIGURE 4 – Coût de la distribution des graphes de calculs pour A-DSA et Max-Sum, après chaque événement, sur des infrastructures uniformes (gauche) et dépendantes du problème (droite), et sur coloration *scale free* (haut), coloration aléatoire (milieu), et Ising (bas).

mais ne peut pas être calculé au moment de l'exécution). Les figures 4a et 4b montrent les coûts de distribution pour les 100 instances. Comme le coût de distribution global est constitué de coûts de communication et d'hébergement, nous traçons également ces deux coûts indépendamment. Dans tous les cas, le coût de l'hébergement augmente logiquement de $10 \cdot k$ à chaque événement de perturbation, les k calculs étant déplacés de leur agent initial à un autre (où le coût d'hébergement est de 10). Sur les modèles *scale free* et Ising, les coûts d'hébergement et de communication ont le même ordre de grandeur. Mais, pour les graphes aléatoires, une densité plus élevée implique qu'il y a plus d'arcs dans le graphique et, par conséquent,

le coût de communication global est plus élevé. En général, les coûts de communication diminuent à chaque réparation, sauf dans Ising avec une infrastructure dépendant du problème. Ici, tous les agents sont homogènes et les calculs sont nécessairement transférés vers des agents plus coûteux, en termes de communication (il n'existe aucun agent moins coûteux ni équivalent aux agents manquants) et les coûts de communication augmentent donc progressivement. Dans les autres cas, les calculs peuvent passer à un agent moins coûteux, ce qui entraîne une diminution des coûts de communication.

6 Conclusions

Nous avons étudié la distribution résiliente de calculs sur un ensemble d'agents dynamique, et deux algorithmes distribués ont été proposés : (i) un protocole de placement de réplicas (DRPM) et (ii) un protocole de réparation (DRPM[MGM-2]) basé sur les réplicas placés par DRPM et sur l'algorithme MGM-2. Nos contributions ont été évaluées expérimentalement en exécutant Max-Sum et A-DSA sur des systèmes dynamiques où les agents disparaissent lors du processus d'optimisation. Pour toutes les configurations étudiées, l'exécution de ces algorithmes a été peu impactée par notre méthode de réparation, et le système continue de fournir des solutions, ce qui démontre la résilience du système. Dans nos expérimentations, A-DSA s'est montré le moins impacté par la suppression d'agents et la réparation.

Nous nous sommes focalisés ici sur le pire scénario. Nous étudierons des scénarios moins critiques où d'autres agents peuvent (ré-)apparaître. Nous avons proposé d'utiliser MGM-2 comme algorithme au cœur de la réparation, mais d'autres algorithmes légers de résolution de DCOP pourrait être considérés, voire conçus spécifiquement. Nous visons également à appliquer nos contributions à un cadre plus large de graphes de calculs, comme la virtualisation de fonctions réseaux [11].

Références

[1] A. Barabási. Emergence of scaling in random networks. 286(5439) :509–512.

[2] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, pages 13–16, New York, NY, USA, 2012. ACM.

[3] Fadoua Chakchouk, Sylvain Piechowiak, René Mandiau, Julien Vion, Makram Soui, and Khaled Ghedira. Fault Tolerance in DisCSPs : Several Failures Case. In Fernando De La Prieta, Sigeru Omatu, and Antonio Fernández-Caballero, editors, *Distributed Computing and Artificial Intelligence, 15th International Conference*, volume 800, pages 204–212. Springer International Publishing.

[4] L. Cohen and R. Zivan. Max-sum revisited : The real power of damping. In Gita Sukthankar and Juan A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 111–124, Cham, 2017. Springer International Publishing.

[5] 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.

[6] Zahia Guessoum, Jean-Pierre Briot, and Nora Faci. Un mécanisme de réplication adaptative pour des sma tolérants aux pannes. In *JFSMA*, pages 135–148.

[7] R.T. Maheswaran, J.P. Pearce, and M. Tambe. Distributed algorithms for dcop : A graphical-game-based approach. In *Proc. of the 17th International Conference on Parallel and Distributed Computing Systems (PDCS)*, pages 432–439, 2004.

[8] G. Malewicz, M. H. Austern, A. J.C Bik, J. C. Dehnert, I. Horn, N. Leiser, and G. Czajkowski. Pregel : A system for large-scale graph processing. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 135–146. ACM, 2010.

[9] P.J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT : Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 2005.

[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] Q. Pham, K. Singh, A. Bradai, G. Picard, and R. Riggio. Single and multi-domain adaptive allocation algorithms for vnf forwarding graph embedding. *IEEE Transactions on Network and Service Management*, pages 1–1, 2018.

[12] S. Russel and P. Norvig. *Artificial Intelligence : a Modern Approach*. Prentice-Hall, 3rd edition, 2009.

[13] P. Rust, G. Picard, and F. Ramparany. On the deployment of factor graph elements to operate max-sum in dynamic ambient environments. In *8th International Workshop on Optimisation in Multi-Agent Systems (OPTMAS 2017)*, 2017.

[14] T. Saraç and A. Sipahioglu. Generalized quadratic multiple knapsack problem and two solution approaches. *Computers & Operations Research*, 43(Supplement C) :78 – 89, 2014.

[15] M. Vinyals, M. Pujol, J. A. Rodriguez-Aguilar, and J. Cerquides. Divide-and-coordinate : DCOPs by agreement. page 8.

[16] M. Vinyals, J. A. Rodriguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems*, 22(3) :439–464, 2010.

[17] R. Wattenhofer. *Principles of Distributed Computing*. 2015.

[18] O. Wolfson and A. Milo. The multicast policy and its relationship to replicated data placement. *ACM Trans. Database Syst.*, 16(1) :181–205, March 1991.

[19] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout : properties, comparison and applications to constraint optimization problems in sensor networks. 161(1) :55–87.

La cohésion comme outil pour le maintien de l'intégrité fonctionnelle d'un système multi-agents

Mickaël Bettinelli[a] mickael.bettinelli@lcis.grenoble-inp.fr
Damien Genthial[a] damien.genthial@univ-grenoble-alpes.fr
Michel Occello[a] michel.occello@univ-grenoble-alpes.fr

[a]Univ. Grenoble Alpes, Grenoble INP*, LCIS, 26000 Valence, France
*Institute of Engineering Univ. Grenoble Alpes

Résumé

Dans un contexte de systèmes ouverts, les agents d'un système peuvent être amenés à travailler avec d'autres agents dont on ne connaît pas le comportement. Ils doivent être capables d'adapter dynamiquement leur comportement afin de garantir le bon fonctionnement du système. Faire un rapprochement entre ces groupes ouverts d'agents et les groupes d'humains étudiés en Sciences Humaines et Sociales (SHS) ouvre de nouvelles perspectives dans la manière de maintenir l'intégrité fonctionnelle d'un système artificiel. Nous proposons de nous inspirer de mécanismes de cohésion issus des SHS dans le but d'améliorer la résilience de ces systèmes.

Mots-clés : *système multi-agents, intégrité fonctionnelle, cohésion*

Abstract

In the context of open systems, agents can work with other unknown agents. They must therefore be able to dynamically adapt their behavior to ensure that the system functions properly at all times. Bringing together these open groups of agents and the Human and Social Sciences (SHS) groups opens up new perspectives in how to maintain the functional integrity of an artificial system. We propose then to draw inspiration from mechanisms of cohesion resulting from SHS in order to improve the resilience of these systems.

Keywords: *multiagent system, functionnal integrity, cohesion*

1 Introduction

Aujourd'hui, une grande majorité des systèmes numériques sont hautement distribués et ouverts, impliquant des entités hétérogènes qui interagissent, dotées de capacités de décision avancées. Considérer ces systèmes artificiels comme des sociétés de systèmes ou d'objets intelligents ouvre de larges perspectives à travers l'analogie que nous pouvons faire avec les organisations sociales. Ces systèmes sont composés de groupes d'agents pouvant présenter le besoin de travailler en équipe afin d'atteindre un objectif commun. Ces groupes ont besoin de cohésion pour maintenir leurs agents unis et atteindre leurs objectifs. La cohésion est un concept qui porte sur les mécanismes qui relient un petit ou un grand ensemble d'unités, de manière plus ou moins étroite. La notion de cohésion est imbriquée dans la construction et le maintien en activité d'une société, d'une collection d'individus, de groupes, d'organisations. En informatique, la notion de cohésion est jusqu'ici principalement limitée à des aspects structurels de l'organisation des groupes se traduisant par le maintien de la connectivité des relations entre les individus (essaim de robots [17], composants logiciels [20], réseaux [23]). Dans un système artificiel décentralisé, le maintien de l'intégrité fonctionnelle représente la capacité d'un système multi-agents à réaliser son objectif [19]. Des facteurs comme le nombre d'agents dans le système et le nombre de communications inter-agents peuvent rendre un système inefficace voire défaillant, et donc menacer son intégrité fonctionnelle [15],[24]. Pour éviter la panne, un système multi-agents doit être capable de s'auto-organiser afin de maintenir ses membres unis et ainsi rétablir ses performances. Maintenir l'intégrité de ces sociétés artificielles se rapproche alors du maintien de la cohésion [6] dans les sociétés humaines et nous pouvons tirer parti des études menées dans les Sciences Humaines et Sociales (SHS). La notion a été abordée très récemment pour la modélisation dans des simulations comportementales au sein d'un SMA [1]. Pour l'intégrité fonctionnelle des SMA, si de nombreux travaux se penchent sur la tolérance aux pannes [3] peu font appel à des notions comportementales liées aux dynamiques de groupe. Des travaux abordent l'auto-évaluation d'un agent par rapport aux membres de son groupe à travers la notion de diagnostic social [14],[21]. Ils approchent ainsi la notion de cohésion mais uniquement du point de vue de son évaluation individuelle. Nous proposons

dans cet article un modèle d'agent s'inspirant des mécanismes de cohésion des SHS dans le but d'améliorer l'intégrité fonctionnelle des SMA.

Ce document est organisé de la manière suivante ; la partie 2 introduit la notion de cohésion de groupe telle qu'elle est vue en SHS ainsi qu'une liste de critères de la cohésion intégrables aux systèmes artificiels permettant une amélioration de leur résilience. La partie 3 présente un cas d'étude et décrit comment y intégrer les critères de cohésion. La partie 4 décrit le modèle d'agent cohésif que nous proposons. La partie 5 présente l'environnement d'expérimentation et spécifie les méthodes d'évaluation du système sur le cas d'étude ainsi que les résultats associés. Enfin, nous concluons dans la partie 6.

2 La cohésion

2.1 La cohésion de groupe

Définition. La cohésion est définie par Festinger comme la somme des forces qui agissent sur les membres dans le but de maintenir le groupe [10]. Ces forces dépendent de l'attraction et de la répulsion de plusieurs critères tels que le prestige du groupe, ses membres, ou les tâches sur lesquelles le groupe travaille. Plus tard, Carron [6] y ajoute la notion d'unité et décrit la cohésion comme un processus dynamique reflétant la tendance des membres d'un groupe à rester ensemble et à maintenir une unité dans la poursuite de buts communs.

Mesures de la cohésion. Mikalachki suggère que la cohésion peut être divisée en deux composantes [18] : la cohésion d'activité et la cohésion sociale. Il soutient que la cohésion d'activité apparaît lorsque les membres du groupe se rassemblent autour de la tâche qu'ils sont censé effectuer alors que la cohésion sociale apparaît lorsqu'ils se regroupent autour d'une fonction sociale. Plusieurs modèles ont été créés pour décomposer la cohésion de groupe.

Le premier présente les deux grandes catégories suggérées par Mikalachki, la cohésion d'activité et la cohésion sociale. La cohésion d'activité représente le degré d'implication des membres du groupe dans le travail d'équipe afin de réaliser des objectifs communs et la cohésion sociale représente l'attraction interpersonnelle des membres d'un groupe. Ce modèle à l'avantage de séparer les composantes sociales et d'activité et se veut intuitif.

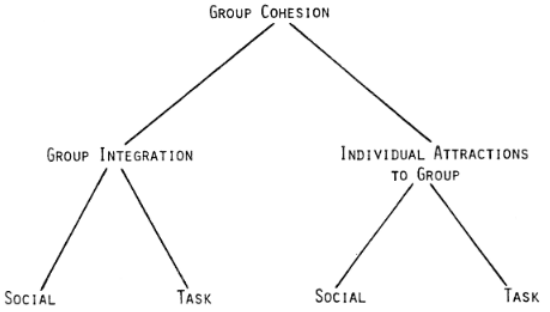


Figure 1 – Modèle proposé par Carron (1985)

Le second (Figure 1), présenté par Carron et al. [7], prend en compte l'attraction de l'individu au groupe ainsi que son intégration dans ce dernier. Il présente deux dimensions de la cohésion qui sont le Goup Integration (GI) et l'Individual Attraction To Group (ATG). La première (GI), représente la façon dont les individus perçoivent le groupe (similarité, proximité, etc.), la seconde (ATG), représente la satisfaction des attentes que le groupe apporte aux individus (interactions avec les autres, productivité, objectifs, etc.). Comme le montre la Figure 1, le modèle redécompose ces deux dimensions tel que l'avait fait Mikalachki afin de les diviser en composantes sociales et d'activités. Carron propose donc un modèle multi-dimensionnel où les critères de la cohésion se répartissent dans les dimensions GI-S, GI-T, ATG-S et ATG-T.

Heuzé et Fontayne cherchent à définir une mesure francophone de la cohésion [12]. Pour cela, ils examinent l'utilité du Group Environment Questionnaire (GEQ) [7] pour une mesure de la cohésion dans les équipes sportives françaises ainsi que la fiabilité du modèle de Carron. Leur étude permet de créer un questionnaire similaire au GEQ appelé Questionnaire sur l'Ambiance du Groupe (QAG). Ainsi, une trentaine de critères ont été réécrits sur le QAG, et répartis sur les quatre dimensions vues précédemment : GI-S, GI-T, ATG-S, ATG-T. Chacun d'eux affirme un ressenti sur le groupe sur lequel la personne interrogée peut être d'accord ou non, par exemple, *Je n'aime pas le style de jeu de mon équipe*, ou *J'ai quelques uns de mes meilleurs amis dans l'équipe*.

En plus des items du QAG, d'autres facteurs sont favorables à la création d'un climat d'équipe. Contrairement à ceux du QAG, ceux-ci ne sont pas centrés sur les perceptions des individus, mais sur l'état du groupe en lui-même. Carless et De Paola mènent une étude similaire à celle de

Carron et al. concernant la mesure de la cohésion dans laquelle ils se concentrent sur les équipes de travail [5]. Ainsi, en recherchant des corrélations entre les dimensions de la cohésion et les caractéristiques des groupes de travail, ils déterminent de nouveaux critères permettant d'évaluer la cohésion de groupes. Ceux-ci sont fortement corrélés avec la cohésion d'activité :

— interactions dans l'équipe ;
— efficacité de l'équipe ;
— présence d'aide sociale ;
— présence d'esprit d'équipe.

2.2 La cohésion artificielle

Dans l'objectif de transposer la cohésion vue en SHS aux systèmes artificiels, il est nécessaire d'extraire de la sociologie des critères quantifiables afin de pouvoir les réutiliser dans un modèle de décision basé sur la cohésion. Nous extrayons du QAG des critères de la cohésion et ajoutons ceux de Carless et Paola [5] (Table 1), nous retenons une liste de 18 items permettant l'apparition de la cohésion au sein d'un groupe. Pour donner quelques exemples des critères extraits : *Satisfaction de l'objectif de l'équipe* (ATG-T), *Présence d'affinités dans le groupe* (ATG-S), *Coopération dans l'équipe* (GI-T), *Interactions dans l'équipe* (GI-S), etc. Ces items représentent chacun une partie du QAG et évaluent aussi bien les dimensions sociale que d'activité du groupe. Chaque individu se construit alors une perception de son groupe qui lui permet d'avoir ou non un sentiment d'appartenance à celui-ci. Bien que le QAG comporte 31 items, cette liste n'en contient que 18. La principale raison est que de nombreux critères du QAG se ressemblent. Même s'ils sont différents du point de vue des sciences sociales, les items sont parfois trop ressemblants pour être intégrés à un système artificiel avec de réelles distinctions. Pour illustrer ceci, nous pouvons par exemple prendre les items ATG-S22 *J'aime l'ambiance qui règne au sein de mon équipe* et ATG-S26 *Je trouve qu'il règne une bonne ambiance entre les membres de mon équipe*. Les deux items sont sous la même dimension (ATG-S) et leur différence dans l'optique d'une intégration à un système artificiel peut paraître négligeable. De plus, certains items du QAG tel que ATG-S1 *Je n'aime pas participer aux activités extra-sportives de mon équipe* sont très difficilement intégrables à un système artificiel.

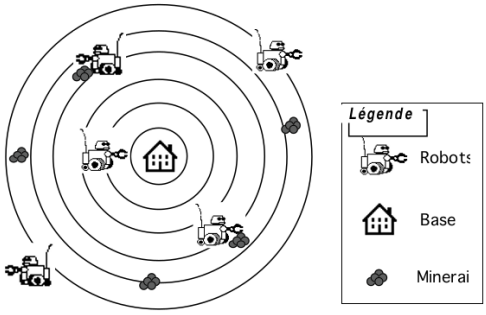


FIGURE 2 – Exploration de Mars avec des agents

3 Intégration des critères au système artificiel

3.1 Présentation du cas d'étude

Les mécanismes de cohésion doivent s'évaluer sur un cas d'étude dans lequel se forment des coalitions d'agents. Le cas d'étude choisi est celui de l'exploration de Mars, présenté dans le livre de Ferber [9] (Figure 2). L'objectif est de récolter des échantillons de minerais autour d'une base sur Mars à l'aide de robots représentant chacun un agent. Dans l'exemple de Ferber, il existe 3 types d'agents : 1. les détecteurs (qui explorent la planète à la recherche de minerais), 2. les foreurs (qui extraient le minerai du sol), 3. les transporteurs (qui rapportent le minerai à la base).

De par leur nature, les agents sont interdépendants, les transporteurs ne peuvent pas transporter de minerai si les foreurs n'en n'ont pas extrait, et les foreurs ne peuvent pas l'extraire si les détecteurs n'en n'ont pas trouvé. Chaque agent est contraint dans ses communications par une portée maximale d'émission. Dans ce cas pratique, une équipe d'agents valide est composée d'au moins 3 membres, un de chaque type disponible (détecteur, foreur, transporteur). Ce cas d'étude demande donc aux agents de coopérer pour atteindre leur objectif personnel. Ils doivent alors s'organiser en coalitions pour pouvoir s'entraider.

3.2 Intégration des critères au cas d'étude

Les avantages apportés par les critères de cohésion sont multiples :

— aide à l'auto-organisation : les agents ont tendance à s'auto-organiser en coalitions ;

— augmentation de la productivité : le système termine son travail plus vite en communiquant moins ;
— augmentation de la résilience : le système gère mieux les périodes de stress sans céder aux pannes ;
— détection de pannes : le système détecte mieux les dysfonctionnements bloquants durant la période d'exécution.

La table 1 illustre les avantages apportés par chaque critère de cohésion. La facilité d'implémentation est prise en compte afin d'aider les concepteurs de systèmes à choisir quels critères intégrer en priorité. L'augmentation de la résilience n'est pas vraiment utile dans ce tableau puisqu'elle est la conséquence plus ou moins directe de chaque critère de cohésion.

Assez peu de critères sont aisément intégrables aux systèmes artificiels. En effet, la plupart d'entre eux nécessitent une structure d'agent cognitif relativement complexe afin de fonctionner. Par exemple, l'aide sociale et la coopération inter-agents demandent aux agents de posséder une mémoire sur laquelle ils peuvent raisonner afin de déterminer qui doit être aidé, comment, quand, etc., mais aussi des protocoles de communication spécifiques pour agir efficacement. De la même manière, l'esprit d'équipe est un élément complexe à mettre en place puisqu'il nécessite que les agents prévoient des actions en fonction de leurs états internes. Au contraire, certains critères s'intègrent plus facilement comme la satisfaction de l'implication de l'agent dans son objectif individuel, ou encore la qualité des relations à travers les interactions des agents.

Afin d'intégrer les critères de la cohésion dans un système artificiel, nous recherchons prioritairement ceux qui sont facilement implémentables et qui couvrent un large éventail d'avantages parmi ceux présentés dans la table 1 (permettant l'auto-organisation, favorables à l'augmentation de la productivité et de la résilience, et permettant de détecter les pannes). Finalement, nous retenons tous les critères facilement implémentables : 1. satisfaction de l'implication du membre dans l'objectif, 2. interactions dans l'équipe, 3. présence d'affinités dans le groupe, 4. satisfaction du rôle joué dans le groupe, 5. satisfaction du rôle acquis dans le groupe. Les critères #4 et #5 portent tous les deux sur le rôle que jouent les agents. Or, comme nous l'avons vu, le cas d'étude présenté ne laisse pas le choix des rôles aux agents et rend donc ces deux critères inutilisables. Au contraire, si le cas d'étude utilisait des agents totipotents, les critères #4 et #5 prendraient une place plus importante et pourraient jouer un rôle dans l'auto-organisation.

Pour conclure, les critères de cohésions intégrés à notre cas d'étude sont : *satisfaction de l'implication du membre dans l'objectif*, *interactions dans l'équipe* ainsi que *présence d'affinités dans le groupe*.

4 Modèle d'agent cohésif

Dans l'optique de généraliser l'utilisation des critères de cohésion, nous proposons un modèle d'agent cohésif, qui, en fonction des critères précédents nécessite les caractéristiques suivantes :
— une représentation de l'environnement ;
— une représentation d'eux-mêmes ;
— des capacités sociales ;
— la capacité de raisonner sur leurs connaissances.

Ces propriétés peuvent être retrouvées dans plusieurs architectures cognitives connues telles que Soar [16], ACT-R [2], CLARION [22], LIDA [11], BDI [4], ou encore FORR [8]. Ces six architectures remplissent nos besoins de représentation de l'environnement, d'eux mêmes et de raisonnement sur des connaissances. Certaines, comme Soar, BDI, ACT-R et CLARION, permettent même de créer des agents sociaux. Beaucoup de ces architectures (Soar, LIDA, FORR) possèdent aussi des capacités d'apprentissage dont l'utilisation dépasse le cadre de cette étude. Quant à elles, les architectures ACT-R et CLARION sont très gourmandes en temps de calcul ce qui les rend difficilement utilisables sur un grand nombre d'agents et empêcherait un passage à l'échelle. Une architecture simple s'appuyant sur le modèle BDI semble donc la plus représentative de nos besoins parmi celles étudiées. Pour cette raison, nous la réutilisons et la modifions dans le but d'expérimenter l'effet des critères de cohésion dans les systèmes artificiels.

Les modifications apportées au modèle BDI classique font en sorte que l'agent puisse prendre en compte l'état de ses accointances pour trier les messages reçus ainsi que son état personnel pour prendre une décision.

Ainsi, le modèle BDI cohésif (figure 3) est modélisé sous la forme du tuple :

$A_{cohesif}$ = < Messages, S_M, filtre, Accointances, Implication, Désirs , plans, ctx, Intentions, S_I >

Ce modèle reprend le même cycle de raisonnement qu'un modèle BDI classique en y intégrant

Critères	Augmentation de la résilience	Facilité d'implémentation	Augmentation de la productivité	Aide à l'auto-organisation	Détection de pannes
Satisfaction de l'implication du membre dans l'objectif	X	X	X	X	
Interactions dans l'équipe	X	X			X
Présence d'esprit d'équipe	X		X		
Efficacité de l'équipe	X		X	X	
Présence d'affinités dans le groupe	X	X			
Satisfaction du rôle joué dans le groupe	X	X		X	
Satisfaction de la manière dont l'équipe performe son objectif	X		X	X	
Satisfaction de l'objectif de l'équipe	X			X	
Capacité d'évolution de l'individu dans le groupe	X		X	X	
Importance du groupe social pour l'individu	X				
Satisfaction du rôle acquis dans le groupe	X	X		X	
Coopération dans l'équipe	X		X		
Implication des membres du groupe dans l'activité	X			X	X
Ambiance appréciée au sein du groupe	X				X
Présence d'aide sociale	X		X		
Satisfaction des priorités de l'équipe	X		X	X	
Préférence des activités des autres groupes	X		X	X	
Compatibilité des objectifs individuels pour l'objectif commun	X		X	X	

TABLE 1 – Avantages apportés aux systèmes artificiels par les critères de cohésion

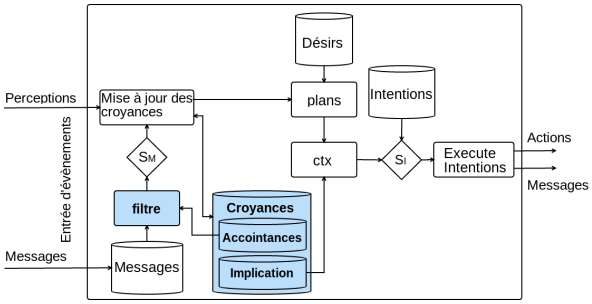


FIGURE 3 – Cycle de raisonnement du modèle BDI cohésif proposé

quelques modifications (en couleur sur la figure 3).

Gestion des messages (Messages, S_M). Le module de Messages est un tampon dans lequel se trouvent tous les messages reçus par l'agent. Ces messages sont stockés le temps d'être pris en charge par l'agent. La fonction de sélection de message permet de choisir quel message sera traité par l'agent. Dans ce cas d'étude, les messages sont traités dans leur ordre de réception (FIFO) mais il est possible de concevoir une heuristique afin d'adapter le comportement de l'agent aux besoins.

La fonction de mise à jour des croyances. La fonction de mise à jour des croyances apporte à l'agent de nouvelles informations construites par ses perceptions ou par les messages reçus des autres agents. La mise à jour des croyances impacte le module plans ce qui va permettre de modifier le comportement de l'agent en fonction de ses connaissances.

Implication, Accointances. L'état des croyances de l'agent est augmenté par l'ajout de modules d'implication et d'accointances. Le premier est une auto-estimation par l'agent de la qualité de son implication dans son travail. L'implication permet aux agents cohésifs de remarquer un manque d'activité et de réagir en conséquence via le module ctx. Ce premier mécanisme offre aux agents un moyen de s'auto-diagnostiquer un dysfonctionnement ce qui permet d'améliorer leur résilience individuelle, et donc celle du système.

Le module Accointances associe un score relationnel à chaque accointance d'un agent ce qui permet d'estimer la qualité de chaque relation. Puisque le module Accointances à un effet sur la fonction de filtre de message, ce score est une analogie de la confiance. Par la quantification de la fiabilité des accointances d'un agent, ce second mécanisme permet à un agent de détecter les agents dysfonctionnants avec qui il est en relation. La reconnaissance de ces agents permet alors au système de les isoler et donc d'améliorer sa résilience.

La fonction de filtre. La fonction de filtre permet de filtrer les messages par rapport au module d'accointances de l'agent. Les scores relationnels des accointances sont comparés à un seuil servant à déterminer si une relation est trop mauvaise ou non afin d'ignorer les messages de l'expéditeur. La fonction de filtre joue donc un rôle important dans la reconnaissance des agents dysfonctionnants et par conséquent dans le maintien de l'intégrité fonctionnelle des agents du système.

Désirs, Intentions. Les désirs sont les motivations de l'agent. Ils représentent les objectifs ou les situations que l'agent aimerait voir s'accomplir. Les intentions représentent ce que l'agent veut faire. Contrairement aux désirs, où l'agent ne vérifie pas la faisabilité, les intentions sont réalisables, certaines pouvant être partiellement engagées.

Génération de plans (plans, ctx). Afin d'agir, l'agent génère plusieurs plans en lien avec ses désirs. La fonction de contexte hiérarchise les plans selon la situation en évaluant leur utilité et ne retient que ceux qui sont applicables. À la fin de cette étape, l'agent a donc un aperçu des plans réalisables et de leur utilité.

Gestion des intentions (S_I, Exécute intentions). À ce niveau, tous les plans réalisables sont retenus. La sélection de l'intention va donc permettre de ne garder qu'un seul plan et de choisir une action à exécuter. Le plan sélectionné est le plan qui a le plus grand score d'utilité.

Pour illustrer le comportement d'un des trois types d'agents, la figure 4 montre le diagramme d'activité d'un agent foreur. Ce diagramme représente les mécanismes d'implication et d'évaluation des accointances expliqués précédemment. Le premier mécanisme de cohésion intervient sur la sélection du message permettant d'isoler socialement les agents qui ne réalisent

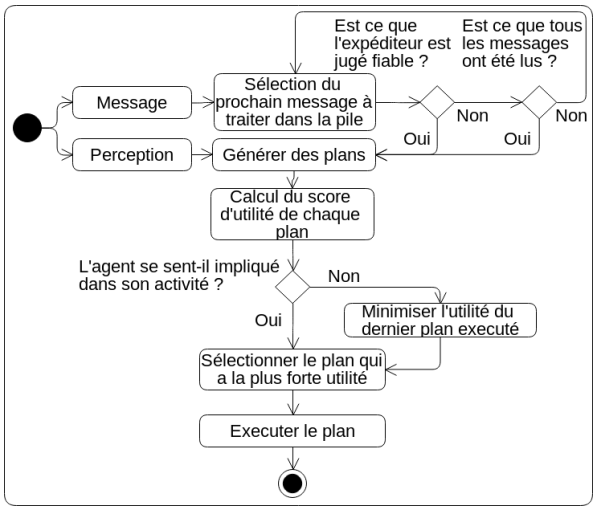


FIGURE 4 – Diagramme d'activités d'un agent cohésif

pas correctement leur travail. Le second, sur le choix d'un plan, permet à l'agent de changer son comportement en cas d'insatisfaction du travail réalisé précédemment. Ces deux mécanismes, complémentaires, participent tous deux à l'amélioration de la résilience du système en rendant possible l'auto-diagnostic d'un dysfonctionnement ou en isolant les agents du système présentant une panne. Nous montrerons dans la partie 5 l'effet de ces mécanismes sur le cas d'étude.

5 Évaluation du système

5.1 Expérimentation

L'expérimentation est construite sur le logiciel MASH [13]. MASH (Multiagent Software / Hardware simulator) est un outil permettant la simulation et l'exécution de systèmes multi-agents embarqués. Les agents sont implémentés en Java et sont exécutés par ce simulateur. Nous utilisons MASH afin de reproduire le cas d'étude et tester différentes solutions construites à partir de fichiers de configuration eux-même générés pseudo-aléatoirement à l'aide de scripts Python. Les agents fonctionnent avec le modèle BDI cohésif explicité dans la partie précédente. Comme nous l'avons vu, les cycles de raisonnements entre le modèle BDI classique tel que nous l'avons introduit et le modèle BDI cohésif sont très similaires. Cette similitude nous permet donc de comparer l'efficacité des deux modèles et voir quels avantages peuvent apporter les agents cohésifs au système.

Dans les parties suivantes, nous évaluerons la ré-

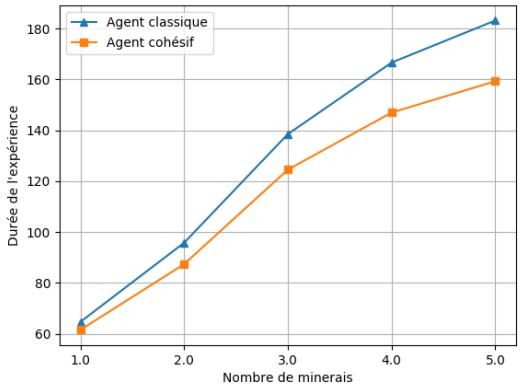


FIGURE 5 – Durée de l'expérience en fonction du type d'agent et du nombre de minerais dans l'environnement

silience du système cohésif afin de voir si les mécanismes de cohésion ont engendré une amélioration de sa résilience. Nous comparerons aussi les performances de ces systèmes dans le but de voir si l'ajout de ces mécanismes modifient leur efficacité.

5.2 Insertion de pannes

Des pannes sont insérées dans certains agents des systèmes exécutés. L'objectif est de voir les effets de ces dysfonctionnements sur ces systèmes composés soit d'agents cohésifs (implémentant des mécanismes de cohésion), soit d'agents classiques (sans les mécanismes de cohésion). L'effet de ces pannes sur chaque système est ensuite comparé afin de voir les bénéfices que les mécanismes des agents cohésifs peuvent apporter.

Dysfonctionnement du GPS. Dans ce scénario, les agents continuent de suivre un comportement normal et essaient d'acheminer le minerai foré par les agents foreurs jusqu'à la base. Les transporteurs possèdent un GPS défaillant les empêchant de temps en temps de se déplacer à la position souhaitée. Le mécanisme d'auto-évaluation de leur implication entre alors en jeu pour rectifier leur comportement.

Le graphique 5 compare la durée d'expérience (en secondes) entre un système composé d'agents classiques (système classique) et un système composé d'agents cohésifs (système cohésif) avec des nombres de minerais différents. Naturellement, plus il y a de minerais dans l'environnement, plus les agents prennent de temps pour les ramasser. De la même manière, il est naturel que les deux types d'agents composant les systèmes évalués ne soient pas tout à fait aussi rapides l'un que l'autre. Il est donc intéressant de noter que l'écart entre les deux courbes n'est pas constant mais qu'il se creuse petit à petit. Cet écart s'explique par la différence de réactivité des agents cohésifs par rapport aux agents classiques. En effet, comme expliqué dans les parties précédentes, les agents cohésifs intègrent un mécanisme d'implication permettant de s'auto-évaluer dans leur activité. Ainsi, lorsqu'un agent transporteur se rend aux mauvaises coordonnées, il est incapable de récupérer le minerai du foreur qui l'a contacté. Ne pouvant pas remplir son rôle, l'implication de l'agent diminue jusqu'à un certain seuil ce qui provoque la recherche d'une nouvelle tâche. Un agent transporteur classique ne présentant pas ce type de mécanisme se rend aux mauvaises coordonnées et attend longuement que le foreur lui donne le minerai. Finalement, l'agent transporteur est débloqué grâce à un mécanisme de réinitialisation permettant aux agents bloqués de retourner à la base pour continuer leur travail. En bref, dans ce scénario le système d'agents cohésifs est plus réactif face aux erreurs, les agents se reprennent en main plus rapidement et de manière dynamique en estimant leur implication dans leur activité par un ratio entre le temps d'inactivité et le temps de travail. Au contraire, le système d'agents classiques est peu réactif face aux pannes, ce qui peut se voir par une évolution plus rapide de la durée des expériences.

Propagation de fausses informations. Dans ce scénario, le détecteur n'est plus capable de distinguer les minerais vides des minerais pleins. Lorsqu'il détecte la position d'un minerai, il se déplace vers celui-ci et l'enregistre. Il informe alors le foreur de sa position pour le récupérer. Un détecteur en bon fonctionnement met à jour ses connaissances sur l'état des minerais autour de lui. Au contraire, un agent détecteur défectueux n'est plus capable de distinguer les minerais vides des autres. Ainsi, il demande de l'aide aux foreurs pour extraire du minerai sur des sites où il n'y en a plus. Le système classique ne peut donc maintenir son intégrité fonctionnelle puisque l'agent défectueux appelle sans cesse ses accointances qui essaient de l'aider sans jamais remettre en question ses demandes. En bref, le système tombe donc dans une boucle sans fin et est incapable de détecter la fin du travail. Dans ce cas, la défaillance est assez importante pour bloquer le fonctionnement complet du système, elle est donc considérée comme

une panne. Le mécanisme de sélection de messages rentre alors en jeu en permettant aux agents cohésifs de maintenir le fonctionnement du système. En effet, dans le cas d'un système composé d'agents cohésifs, les agents isolent l'individu défectueux grâce à un score de confiance tenu pour chacune de leurs accointances. Lorsqu'un agent est défectueux et que son comportement est contre-productif pour ses pairs, ses relations diminuent son score de confiance. À partir d'un certain seuil, les agents du système arrêtent de prendre en compte les messages que l'agent défectueux émet. Le système cohésif peut donc estimer correctement l'avancement du travail et maintenir son intégrité fonctionnelle. Il est alors plus résilient que le système classique face aux pannes de ce type.

Dans ce scénario, contrairement au précédent, le nœud du système distribué ne s'isole pas de lui même, il est exclu par le système à cause de son comportement déviant. En généralisant, ce type de mécanisme serait utile dans le cas d'une attaque du système. Un agent contrôlé de l'extérieur essayant de modifier le comportement prévu du système serait mis à l'écart automatiquement par celui-ci, assurant le maintien de l'intégrité fonctionnelle du système.

5.3 Comparaison de l'efficacité des systèmes

Afin de comparer l'efficacité des systèmes, nous comparons la vitesse de récupération de minerais, la durée des expériences et la charge en communications. Le graphique 6 montre que les agents cohésifs ont une vitesse de ramassage des minerais très proche des agents classiques. Alors que la courbe de ramassage de minerais des agents classiques marque 2 grand paliers (les 2 minerais sont ramassés très rapidement une fois trouvés), les agents cohésifs ont tendance à les ramasser en plusieurs fois. Ceci s'explique par le paramétrage des agents cohésifs qui ont tendances à rapidement changer d'activité lorsqu'ils ne travaillent pas (lorsque des foreurs attendent des transporteurs par exemple). Les agents trop longtemps inactifs rentrent donc à la base et perdent du temps dans la réalisation de l'objectif global. Cependant, le graphique montre aussi que l'exécution des expériences avec des agents cohésifs semblent plus courte qu'avec des agents classiques. En effet, les expériences composées d'agents cohésifs durent en moyenne 67 secondes contre 80 avec des agents classiques. Les agents cohésifs se réorganisent plus rapidement lorsque les détecteurs ne trouvent plus de minerais à forer et les agents rentrent plus rapidement à la base que dans le cas de systèmes composés d'agents classiques.

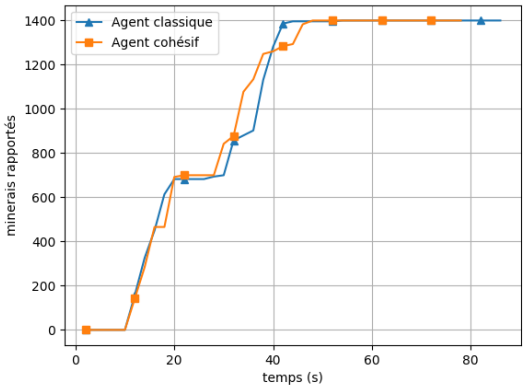


FIGURE 6 – Minerais rapportés en fonction du temps selon le type d'agent avec un seuil d'implication de 0.6 pour les agents cohésifs

	Agents cohésif	Agents classique
Messages / seconde	25.25	24.55
Messages / seconde / agent	3.16	3.07

TABLE 2 – Comparaison du nombre de messages envoyés dans les deux systèmes

De la même manière que pour la vitesse de récupération du minerai, le nombre de messages transmis dans le système est presque identique pour les deux types d'agents. Les agents cohésifs envoient quelques messages de plus, ce qui peut s'expliquer par le fait qu'ils mettent un peu plus de temps à récolter le minerai tout en ayant des temps d'exécution d'expérience globalement plus courts.

Finalement, on retiendra de ces expérimentations que le mécanisme de cohésion n'améliore pas l'efficacité du système, mais que le système cohésif démontre une meilleure résilience et résistance aux pannes que le système classique dans les scénarios étudiés sans introduire de surcoût notable dans son fonctionnement.

6 Conclusion

Nous avons présenté comment les mécanismes de cohésion de groupe peuvent présider à une nouvelle approche de la conception d'agents pour le maintien de l'intégrité fonctionnelle

d'un SMA. Ces mécanismes ont été évalués à l'aide d'un cas d'étude et comparés à des agents de conception plus classique. Comme première conclusion, nous avons vu que l'intégration de mécanismes de cohésion dans les systèmes artificiels a permis une augmentation de leur résilience. Bien que nous n'ayons pas noté d'amélioration concernant l'efficacité du système dans la collecte de minerais ni sur le nombre de communications, la réactivité du système cohésif est meilleure que la réactivité du système classique ce qui diminue le temps moyen des expériences. Pour conclure, les agents cohésifs ont la capacité de maintenir l'intégrité fonctionnelle de leur système tout en limitant l'impact négatif lié à leur comportement par rapport aux agents classiques.

Concrètement, cette approche pourrait être profitable à divers types d'applications distribuées communicantes. Elle peut permettre la transmission de données (messages, fichiers, etc.) entre les nœuds en veillant à préserver leur intégrité (sans oubli, sans redondance et sans modification d'informations). Comme l'ont montré les expérimentations, les mécanismes de cohésion légers rendent possible l'intégration de capacités d'auto-diagnostic de panne ainsi que la capacité de rétablissement du fonctionnement normal dans les nœuds (par exemple, lorsqu'un nœud ne reçoit ou ne transmet plus de données depuis longtemps). Dans le cas où le nœud retransmet plusieurs fois les même messages à ses voisins ou les retransmets avec des erreurs, le système a la capacité de l'isoler et de le rendre inopérant afin de maintenir l'intégrité fonctionnelle du système. Ainsi, par leur légèreté les critères de cohésion pourraient être intégrés à des systèmes distribués existants pour améliorer leur résilience. Bien que l'augmentation de la résilience soit un avantage pertinent, peu de critères de cohésion ont été intégrés et évalués ici. Pour de futures recherches, il sera intéressant d'essayer l'intégration de nouveaux critères de cohésion dans des cas différents et plus poussés afin de voir, par exemple, si les mécanismes de cohésion permettent une amélioration de la productivité ou encore s'ils peuvent engendrer (ou améliorer) l'auto-organisation d'un système d'agents totipotents.

Références

[1] Carole Adam, Catherine Garbay, and Julie Dugdale. Multi-factor model and simulation of social cohesion and its effect on evacuation. In *Proceedings of the 52nd Hawaii International Conference on System Sciences*, pages 667–677, 2019.

[2] John R. Anderson, Michael Matessa, and Christian Lebiere. ACT-R : A Theory of Higher Level Cognition. *Human-Computer Interaction*, 12 :439–462, 1997.

[3] Yasir Arfat and Fathi E. Eassa. A survey on fault tolerant multi agent system. *International Journal of Information Technology and Computer Science (IJITCS)*, 9 :39–48, 2016.

[4] Michael E. Bratman. Intention, Plans, and Practical Reason. *The Philosophical Review*, 100(2) :277–284, 1991.

[5] Sally A Carless and Caroline De Paola. The measurement of cohesion in work teams. *Small group research*, 31(1) :71–88, 2000.

[6] Albert V Carron. Cohesiveness in Sport Groups : Interpretations and Considerations. *Journal of Sport Psychology*, 4 :123–138, 1982.

[7] Albert V Carron. The Development of an Instrument to Assess Cohesion in Sport Teams : The Group Environment Questionnaire. *Journal of Sport Psychology*, 7 :244–266, 1985.

[8] Susan L Epstein. For the Right Reasons : The FORR Architecture for Learning in a Skill Domain. *Cognitive Science*, 18(3) :479–511, 1994.

[9] Jacques Ferber. *Les SMA*. Informatique et intelligence artificielle. InterEditions, 1995.

[10] Leon Festinger. Informal social communication. *Psychological Review*, 57(5) :271–282, 1950.

[11] Stan Franklin and F.G. Patterson. The LIDA architecture : Adding new modes of learning to an intelligent, autonomous, software agent. 703 :764–1004, 2006.

[12] Jean-Philippe Heuzé and Paul Fontayne. Questionnaire sur l'Ambiance du Groupe : A French Language Instrument for Measuring Group Cohesion. *Human Kinetics Publishers*, Journal of Sport & Exercise Psychology(24) :42–67, 2002.

[13] Jean-Paul Jamont and Michel Occello. Meeting the challenges of decentralised embedded applications using multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 5(1) :22–68, 2015.

[14] Meir Kalech and Gal A Kaminka. On the design of social diagnosis algorithms for multi-agent teams. In *Intl. Joint Conference on Artificial Intelligence (IJCAI)*, pages 370–375, 2003.

[15] Marek Kisiel-Dorohinicki and Edward Nawarecki. Functional Integrity of MAS through the Dynamics of the Agents' Population. pages 405–406, 1998.

[16] John E. Laird. *The Soar Cognitive Architecture*. MIT Press, 2012.

[17] Matthew D Manning, Caroline E Harriott, Sean T Hayes, Julie A Adams, and Adriane E Seiffert. Heuristic evaluation of swarm metrics' effectiveness. In *Proceedings of the tenth annual ACM/IEEE international conference on human-robot interaction extended abstracts*, pages 17–18. ACM, 2015.

[18] Alexander Mikalachki. *Group cohesion reconsidered ; a study of blue collar work groups*. School of Business Administration, University of Western Ontario, 1969.

[19] Kamil Piętak, Adam Woś, Aleksander Byrski, and Marek Kisiel-dorohinicki. Functional Integrity of Multi-agent Computational System Supported by Component-Based Implementation. In *Proc. of the Intl. Conf. on Industrial Applications of Holonic and Multi-Agent Systems*, pages 82–91. Springer, 2009.

[20] Amit Rathee and Jitender Kumar Chhabra. Improving cohesion of a software system by performing usage pattern based clustering. *Procedia Computer Science*, 125 :740–746, 2018.

[21] Dirk Van Rooy, Ian Wood, and Eric Tran. Modelling the Emergence of Shared Attitudes from Group Dynamics Using an Agent-Based Model of Social Comparison Theory. *Systems Research and Behavioral Science*, 33(1) :188–204, 2016.

[22] Ron Sun, Edward Merrill, and Todd Peterson. A Bottom-Up Model of Skill Learning. In *Proc. of 20th Cognitive Science Society Conference*, pages 1037–1042, 1998.

[23] Jordi Torrents and Fabrizio Ferraro. Structural cohesion : visualization and heuristics for fast computation. *Journal of Social Structure*, 16 :1–35, 2015.

[24] Y Wallach. Alternating Sequential/Parallel Processing Fundamental and Examples. *IEEE Transactions on Power Apparatus and Systems*, (11) :4397–4401, 1981.

Confidentialité dans les systèmes de réputation

Grégory Bonnet[a] gregory.bonnet@unicaen.fr
Laurent Vercouter[b] laurent.vercouter@insa-rouen.fr
Damien Lelerre[a] damien.lelerre@unicaen.fr

[a]Normandie Université, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France

[b]Normandie Université, UNIROUEN, UNIHAVRE, INSA Rouen, LITIS, 76000 Rouen, France

Résumé

Dans les systèmes de réputation, il a été observé, qu'afin d'éviter des évaluations vengeresses, les témoignages étaient majoritairement positifs, réduisant par là-même l'efficacité du système. Pour inciter les agents à diffuser tous leurs témoignages, les solutions classiques proposent d'anonymiser les témoignages. Dans la littérature, de nombreux travaux ont étudé des approches cryptographiques pour s'assurer à la fois de l'anonymat et de la non-répudiation des témoignages. Toutefois, ceci ne permet de garantir leur confidentialité en raison des corrélations entre transactions et diffusion d'un nouveau témoignage. Dans cet article, nous proposons d'étudier la faisabilité d'une autre approche dans laquelle les témoignages sont bruités et soumis à des délais de diffusion. Des résultats d'expérimentation mettent en lumière l'effet de ces perturbations sur les systèmes de réputation BetaReputation et EigenTrust.

Mots-clés : *Système de confiance et de réputation, Confidentialité*

Abstract

On reputation systems, it has been observed that testimonies are mostly positive in order to avoid vengeful evaluations. Such behaviour reduces the effectiveness of the system. In order to incite agents to publish all their testimonies, classic solutions consist in anonymizing them. In the literature, many works have proposed cryptographic approaches to ensure both anonymity and non-repudiation of testimonies. However, due to correlations between transactions and dissemination of new testimonies, it does not guarantee confidentiality. In this article, we propose to study the feasibility of another approach in which a noise is applied on testimonies and those latter are subject to delays of diffusion. Experimental results highlight the effect of these disturbances on the BetaReputation and EigenTrust reputation systems.

Keywords: *Trust and reputation system, Confidentiality*

1 Introduction

Dans un système multi-agents, il est courant qu'un agent ait besoin des services ou des ressources d'un autre agent. Deux agents peuvent effectuer des *transactions* afin que chacun obtienne ce dont il a besoin. Cependant, un agent peut être plus ou moins qualifié (ou plus ou moins honnête) dans la fourniture de ce qui lui est demandé. Il est donc dans l'intérêt d'un agent de bien choisir ceux avec lesquels il effectue des transactions. Une des approches permettant ceci est l'utilisation de *systèmes de réputation* [10, 15, 19]. Ces systèmes se fondent sur l'attribution de notes évaluant la façon dont une transaction s'est déroulée. Chaque agent est alors en mesure de développer dans un premier temps une *confiance* envers un autre agent à partir de sa propre expérience passée. Dans un second temps, les notes données par chaque agent peuvent être diffusées publiquement sous forme de *témoignages* et aggrégées par une fonction afin de construire une notion de confiance collective plus précise appelée *réputation*.

Sur les sites de vente en ligne utilisant des systèmes de réputation (et où les agents sont majoritairement des humains), il a été observé que les témoignages sont très majoritairement positifs. L'explication de ce phénomène repose sur le fait que, puisque ces notes sont publiques, il y a un risque d'un agent désire *se venger* lorqu'il reçoit une mauvaise évaluation de la part d'un tiers [13]. Pour éviter les vengeances, les agents préfèrent alors uniquement diffuser des témoignages positifs, réduisant ainsi l'efficacité du système de réputation. Afin d'inciter les agents à diffuser tous leurs témoignages, les solutions classiques proposent d'*anonymiser* les témoignages à l'aide d'un protocole cryptographique, permettant ainsi d'obtenir la réputation d'un autre agent sans connaitre les témoignages qui ont servi à calculer celle-ci [2, 7, 17, 20]. Toutefois, ne plus rendre les témoignages publics n'est pas suffisant car il pourrait être possible de déduire leur valeur de la dynamique

de la réputation. C'est pourquoi nous proposons d'étudier la faisabilité d'une autre approche, fondée sur la *confidentialisation* et dans laquelle les témoignages sont bruités et soumis à des délais, décorrélant ainsi les transactions de la diffusion de témoignages.

Le reste de cet article est structuré comme suit. Nous présentons en section 2 les systèmes de réputation, leurs propriétés principales ainsi que deux systèmes particuliers, BetaReputation et EigenTrust, qui nous serviront de référence pour nos expérimentations. La section 3 présente les méthodes de confidentialisation que nous utilisons et le système de réputation que nous proposons. Enfin, la section 4 étudie expérimentalement les effets de ces méthodes sur la performance des fonctions de réputation.

2 Systèmes de réputation

Les systèmes de réputation sont des systèmes d'évaluation distribués, et parfois décentralisés, de la fiabilité des agents. Dans cette section, nous présentons les principaux concepts associés à ces systèmes et détaillons les systèmes qui nous serviront de référence.

2.1 Principes généraux

Dans un système multi-agent, chaque agent peut posséder certaines compétences ou certaines ressources. Si un agent a besoin d'une compétence ou d'une ressource qu'il ne possède pas, il peut réaliser une *transaction* avec un agent qui la possède. Toutefois, chaque agent n'est pas qualifié de façon égale pour fournir un service. De plus, il est possible qu'il soit profitable au fournisseur de service d'en fournir délibérément un de mauvaise qualité. Aussi, lorsqu'un agent a besoin d'un service, il est dans son intérêt d'obtenir un service de bonne qualité, et donc de choisir un fournisseur fiable. Pour choisir un fournisseur, un agent peut se fonder sur ses propres observations afin de mesurer à quel point il peut avoir *confiance* en un autre agent. Au moment d'initier une transaction, un agent doit : (1) choisir un agent en fonction de la confiance qu'il a en lui ; (2) observer le résultat de la transaction ; (3) évaluer ce résultat pour mettre à jour la confiance qu'il attribue au fournisseur de service.

En fondant uniquement la confiance sur ses observations, un agent ne va pas pouvoir évaluer un autre qui lui est inconnu, tout comme il se prive des informations que d'autres agents pourraient avoir. La notion de confiance peut alors être étendue par celle de *réputation*. Cette dernière est une évaluation de la fiabilité d'un agent à partir des observations des autres agents. Pour ce faire, à chaque transaction, les agents publient leurs observations sous forme de *témoignages*. Une *fonction de réputation* permet ensuite à un agent donné d'agréger ces témoignages en une valeur unique. Les fonctions de réputation peuvent être classées selon leurs propriétés [6, 10, 15, 19]. Parmi celles-ci, deux propriétés nous intéressent :

Visibilité. Une fonction de réputation peut être *globale* ou *personnalisée*. Une fonction personnalisée construit une réputation dépendante du point de vue de l'agent qui la calcule : deux agents peuvent calculer deux valeurs de réputation différentes pour un même autre agent. Une fonction est globale si la réputation d'un agent est la même quelque soit l'agent qui la calcule.

Sémantique. Une fonction de réputation peut être *par valeur* ou *par rang*. Une fonction de réputation est par valeur si les valeurs de réputation qu'elle calcule représentent une information en elles-mêmes. Par exemple, certaines fonctions retournent la valeur moyenne des notes attribuées à un agent ou la probabilité que la prochaine transaction sera de bonne qualité. En revanche, une fonction par rang n'attribue pas de sens aux valeurs de réputation en soi mais uniquement à l'ordre qu'elles impliquent entre les agents. Ainsi, un agent ayant une meilleure réputation qu'un autre sera considéré comme meilleur mais sans qu'il soit possible de le quantifier. Notons qu'une fonction par valeur permet aussi de construire un ordre sur les agents.

2.2 Systèmes de référence

De nombreuses fonctions de réputation ont été proposées dans la littérature [4, 9, 11, 14, 18, 22]. Dans la suite de cet article, nous considérons les deux fonctions suivantes, à savoir BetaReputation [9] et EigenTrust [11], car ce sont deux fonctions de référence dans la littérature tout en étant à l'opposé l'une de l'autre sur chacune des propriétés que nous avons mentionné précédemment.

BetaReputation. Cette fonction *personnalisée* et *par valeur* caractérise la réputation d'un agent comme la probabilité qu'il fournisse un service de bonne qualité. Chaque agent a_j représente

1. Ceci peut se faire de diverses façons : mise à disposition sur demande, publication auprès d'une autorité centrale, diffusion générale au système, propagation de voisinage en voisinage, etc.

sa confiance envers un agent a_j par un couple $r_{i,j} \in \mathbb{N}$ et $s_{i,j} \in \mathbb{N}$, correspondant respectivement au nombre de « bonnes » et « mauvaises » transactions avec a_j. Lors d'une nouvelle transaction (r', s') à l'instant t, ces deux valeurs sont mises à jour avec un facteur d'oubli $\lambda \in [0, 1]$:

$$r_{i,j}^t = \lambda r^{t-1} + r'$$
$$s_{i,j}^t = \lambda s^{t-1} + s'$$

Pour simplifier les notations, hormis dans les cas ambigus, nous omettons dans la suite l'exposant t. La réputation d'un agent a_j du point de vue d'un agent a_i est donnée par :

$$Rep_j^i = \frac{R_j^i - S_j^i}{R_j^i + S_j^i + 2}$$

où :

$$R_j^i = \sum_{a_k \notin \{a_i, a_j\}} \frac{2r_{i,k} \times r_{k,j}}{(s_{i,k}+2)(r_{k,j}+s_{k,j}+2)+2r_{i,k}}$$

$$S_j^i = \sum_{a_k \notin \{a_i, a_j\}} \frac{2r_{i,k} \times s_{k,j}}{(s_{i,k}+2)(r_{k,j}+s_{k,j}+2)+2r_{i,k}}$$

EigenTrust. EigenTrust est une fonction de réputation *globale* et *par rang* où chaque agent représente sa confiance envers un agent a_j par un couple $r_{i,j} \in \mathbb{N}$ et $s_{i,j} \in \mathbb{N}$ de « bonnes » et « mauvaises » transactions. EigenTrust s'appuie ensuite sur une matrice de confiance normalisée $\mathcal{C}$ où chaque élément $c_{i,j}$ est défini comme suit :

$$c_{i,j} = \frac{\max(r_{i,j} - s_{i,j}, 0)}{\sum_{a_j \notin \{a_i\}} \max(r_{i,j} - s_{i,j}, 0)}$$

La réputation des agents est un vecteur Rep où la ième composante de Rep, notée Rep_i, est la réputation de l'agent a_i. Ce vecteur est défini comme le point fixe de l'équation suivante où $\vec{p}$ est un vecteur de réputation *a priori* :

$$Rep^{(t+1)} = (1-\alpha) \times \mathcal{C}^T \times Rep^{(t)} + \alpha \times \vec{p}$$

2.3 Politiques de sélection

Calculer des réputations n'est pas suffisant, encore faut-il que les agents s'en servent pour décider avec qui effectuer des transactions. Or, toujours porter son choix sur les agents qui disposent de la meilleure réputation à un instant donné peut conduire à se priver d'information sur les autres agents. Aussi, une politique de sélection doit permettre un compromis entre l'*exploitation* qui correspond au fait de sélectionner des agents ayant une bonne réputation dans l'espoir d'obtenir la meilleure transaction possible, et l'*exploration* qui consiste à choisir un agent non pas parce qu'il a bonne réputation mais pour obtenir une observation supplémentaire et mieux estimer sa fiabilité. Les trois méthodes que nous considérons sont inspirées des politiques de sélection pour le problème du bandit multi-bras [3, 21].

ϵ-gloutonne. Connue pour être simple mais peu performante, cette politique sélectionne l'agent qui dispose de la meilleure réputation avec une probabilité $(1-\epsilon)$ ou sélectionne un agent tiré aléatoirement de manière uniforme avec une probabilité ϵ. Cependant, cela peut conduire à la sélection d'un agent évalué comme peu fiable avec la même probabilité qu'un agent sur lequel aucune information n'est disponible (d'où parfois une faible performance).

UCB. Connue pour être très performante, cette politique choisit l'agent a_j qui maximise une valeur v_j avec :

$$v_j^i = Rep_j^i + \sqrt{\frac{2 \times \ln(1 + r_{i,j} + s_{i,j})}{1 + \Sigma_{a_k \in N}(r_{k,j} + s_{k,j})}}$$

Cela permet une exploration contextuelle qui dépend de la quantité d'information dont dispose l'agent sur les autres. Tant que l'agent dispose de peu d'information, le second terme domine le premier et conduit l'agent à sélectionner ceux sur lesquels il dispose du moins d'observations. Toutefois, cette politique est peu performante face à des agents non stationnaires .

β-softmax. Connue pour être moins performante que UCB mais plus robuste aux comportements non stationnaires, cette politique sélectionne les agents proportionnellement à leur réputation, avec une probabilité p_j^i tirée selon une fonction exponentielle normalisée :

$$p_j^i = \frac{e^{\beta . Rep_j^i}}{\Sigma_{a_k \in N} e^{\beta . Rep_j^i}}$$

Le paramètre $\beta \in \mathbb{R}$ est une température inverse. Si $\beta = 0$, la distribution de probabilité est une

2. Le calcul effectué est généralement un calcul approché à une erreur ϵ près. De plus, cette équation n'est pas garantie de converger vers un point fixe. Le facteur $\alpha \in]0, 1]$ permet de s'en assurer s'il est fixé à une valeur non nulle.

3. Un agent est non stationnaire si sa fiabilité change au fil du temps.

distribution uniforme : chaque agent est sélectionné avec la même probabilité. Si β tend vers ∞, la probabilité de sélectionner l'agent ayant la plus haute réputation tend vers 1 tandis que celles des autres tendent vers 0.

3 Confidentialiser les témoignages

Nous proposons dans cette section deux méthodes pour accroître la confidentialité des témoignages, qui s'inspirent de méthodes utilisées dans le domaine des bases de données. Lorsque des données stockées dans une base doivent rester confidentielles, la première étape est l'*anonymisation* en supprimant les informations permettant d'identifier un individu. Elle n'est cependant pas suffisante car il peut être possible reconstruire les informations cachées, par recoupement par exemple avec d'autres sources d'information. Ainsi, une seconde étape de *confidentialisation* doit être mise en place [1]. Il s'agit de modifier une base de données anonymisée pour empêcher la reconstruction des identités des individus associés aux données.

Quelques travaux se sont déjà intéressés à la confidentialisation des témoignages pour un système de réputation. Hasan *et al.* [7] indiquent qu'une solution consisterait à bruiter les témoignages afin de rendre plus difficile leur déduction à partir de la dynamique de la réputation. Toutefois, les auteurs se sont bornés à ce constat. De manière intéressante, Huang *et al.* [8] ont proposé de confidentialiser les témoignages en les agrégeant avant de les communiquer. Un agent ne déclare plus avoir confiance en un agent donné mais uniquement en un groupe donné. Plus les groupes considérés sont de grande taille, plus la confidentialité est forte car il est difficile d'y distinguer deux agents.

Nous proposons de partir du constat de Hasan *et al.* et d'étudier l'effet d'un bruit appliqué aux témoignages sur les systèmes de réputation. Nous distinguons deux types de bruits : un bruit sur les témoignages eux-mêmes qui correspond au bruit classiquement utilisé pour la confidentialisation des bases de données et un bruit sur la diffusion des témoignages sous forme d'un délai. En effet, si un agent veut observer l'évolution de sa réputation pour déduire la valeur des nouveaux témoignages à son encontre, il peut supposer que ces nouveaux témoignages sont diffusés aussitôt ou presque après la transaction. L'introduction d'un délai vient décorreler ces deux événements.

4. *Privacy preservation*.

3.1 Bruit sur les témoignages

L'ajout de bruit est une méthode courante pour confidentialiser une base de données [5, 16]. Cela consiste à altérer les valeurs numériques de cette base de données afin que les informations relatives à un même individu ne puissent plus être mises en corrélation. Toutefois, l'ajout de bruit modifie le résultat des requêtes : plus les données sont altérées, plus la confidentialité est forte mais moins l'information globale sur la base est conservée. Parmi les méthodes de bruitage, le bruit différentiel [5] permet d'éviter ce problème en calculant un bruit spécifique pour chaque requête possible sur la base tout en minimisant la dissimilarité entre les résultats des requêtes sur la base bruitée et non bruitée. Si cette méthode est particulièrement intéressante, elle s'accommode mal des données dynamiques comme peuvent l'être l'ensemble des agents et leurs témoignages dans un système de réputation. Il serait nécessaire de recalculer l'ensemble de la base synthétique lors de l'arrrivée d'un nouvel agent ainsi qu'à chaque nouveau témoignage. De plus, cette méthode est nécessairement dépendante de la fonction de réputation utilisée. Nous écartons donc pour ces raisons le bruit différentiel.

3.2 Délais de diffusion

Nous proposons d'utiliser différentes méthodes appliquant un délai sur la diffusion des témoignages afin de décorréler la transaction de son évaluation. Une approche naïve consiste simplement à faire usage d'un délai « temporel ». Un témoignage généré à un instant t ne sera diffusé (nous dirons *valide* par la suite) qu'à partir de l'instant $t + \epsilon$, où ϵ est un nombre aléatoire tiré uniformément dans un intervalle donné en paramètre. Ainsi, deux témoignages valides peuvent être confondus et l'agent concerné ne peut pas savoir à quelle transaction associer chacun. Toutefois, une stratégie consistant à attendre suffisamment entre deux transactions pour voir les témoignages être validés peut mettre à mal cette approche. Afin de passer outre ce problème, nous proposons de faire usage d'un délai non plus temporel mais de ne pas valider un témoignage tant qu'un certain nombre d'autres témoignages venant de témoins distincts n'a pas été généré. Cette méthode, en exigeant une diversité de témoins, évite qu'un agent qui serait évalué majoritairement par un unique agent puisse savoir à quelle transaction associer un témoignage. Toutefois, si un agent n'est que peu sollicité, un té-

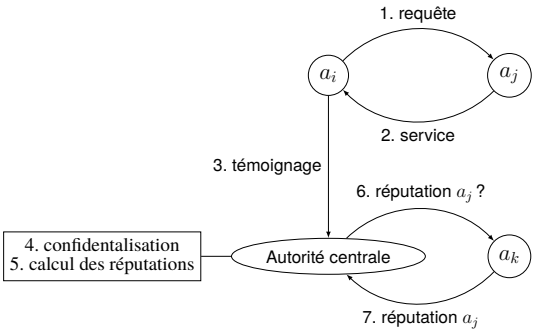


FIGURE 1 – Protocole de confidentialisation

moignage peut rester indéfiniment en attente.

3.3 Un système de réputation confidentiel

Le système de réputation à témoignage confidentiel que nous proposons est illustré en figure 1. Ici, une autorité centrale est en charge du stockage des témoignages et de l'application du bruit et du délai de diffusion. Un agent qui effectue à l'instant t une transaction avec un autre en observe la qualité et évalué celle-ci dans l'intervalle $[-1, 1]$ (-1 pour la plus mauvaise qualité possible, 1 pour la meilleure). Cette évaluation correspond à un témoignage $o_{i,j}$ qui est envoyé à l'autorité centrale. Cette dernière applique en premier lieu un *bruit additif* et enregistre le témoignage bruité $\tilde{o}_{i,j}$ tel que $\tilde{o}_{i,j} = o_{i,j} + X$ où X est une valeur aléatoire tirée selon une une loi normale centrée sur 0 écart-type ϵ. Ici, ϵ est un paramètre du système.

Ensuite, chaque témoignage bruité $\tilde{o}_{i,j}$ d'un agent a_i envers un agent a_j est mis en attente dans une liste $O_{i,j}$. Lorsque la taille de η_1 listes de témoignages en attente franchit un seuil η_2, alors un ensemble de témoignages en attente sur d'autres listes sont validés. Cela va concerner les η_3 plus anciens témoignages de η_4 listes ayant atteint ce seuil. Ici, $\eta_4 \leq \eta_1$ et $\eta_3 \leq \eta_2$, et les η_k sont tous tirés uniformément pour chaque témoignage dans des intervalles définis en tant que paramètres du système. Ainsi, si un agent participe à plusieurs transactions simultanément, chaque témoignage le concernant sera diffusé avec un délai distinct.

Enfin, lorsqu'un agent a_k demande à l'autorité centrale la réputation d'un agent, cette dernière la calcule soit avec la fonction BetaReputation, soit avec EigenTrust. Comme vu en section 2.2, ces fonctions s'appuient sur un couple $(r_{i,j}, s_{i,j})$ de "bonnes" et "mauvaises" transactions. Pour construire ce couple, l'autorité centrale utilise la méthode préconisée par Jøsang et Ismail [9] : chaque témoignage bruité valide $o_{i,j}$ donne un $(\tilde{r}_{i,j}, \tilde{s}_{i,j})$ comme suit et tous les couples $(\tilde{r}_{i,j}, \tilde{s}_{i,j})$ valides sont sommés pour être ensuite utilisé dans le calcul de la réputation.

$$\tilde{r}_{i,j} = \frac{1 + \tilde{o}^t_{i,j}}{2} \qquad \tilde{s}_{i,j} = \frac{1 - \tilde{o}^t_{i,j}}{2}$$

4 Expérimentations

Nous expérimentons dans cette section nos méthodes de confidentialisation en simulation. Nous considérons les fonctions de réputation BetaReputation et EigenTrust ainsi que les politiques de sélection ϵ-gloutonne, β-softmax et UCB. Chaque expérimentation est composée de 10 simulations de 1000 pas de temps durant lesquel 50 agents sélectionnent, interagissent avec et évaluent un autre agent.

4.1 Paramètres expérimentaux

Les transactions effectuées par chaque agent varient en qualité. Cette qualité est une vérité terrain non accessible aux agents. La qualité f_j d'un agent a_j est définie par une espérance et un écart-type tirés uniformément respectivement dans $[-1, 1]$ et $[0, 0.5]$. Lors d'une transaction, la qualité observée par un agent a_i est tiré aléatoirement à partir d'une loi normale paramétrée par la qualité de l'agent a_j sollicité. Cette qualité observée correspond aux témoignages $o^t_{i,j}$ évoqués dans la section précédente. La table 1 récapitule les paramètres de bruit et de délai considérés dans nos expérimentations. Les courbes noires sur les figures pages 7, 8 et 9 indiquent les résultats sans mécanisme de confidentialisation, représentant donc le comportement nominal des fonctions de réputation étudiés.

4.2 Mesures de performance

Afin de mesurer l'influence des procédures d'anonymisation sur les systèmes de réputation, nous considérons deux métriques : une distance entre les valeurs de réputations calculées et la fiabilité réelle des agents – vérifiant que la qualité de l'évalution n'est pas perturbée – et une mesure de regret – vérifiant que la politique de sélection ne l'est pas non plus.

5. Si le nombre d'agents a bien entendu une influence sur les performances absolues des systèmes de réputation, nous avons observé qu'augmenter le nombre d'agents ou le diminuer ne modifie pas la forme générale des résultats, ni leurs performances relatives.

Écart-type ϵ du bruit	Paramètres η_k du délai	Couleur des courbes
–	–	noir
0.2	$(3, 5, 2, 3)$	bleu
0.5	$(4, 6, 3, 4)$	rouge
1.0	$(5, 7, 4, 5)$	vert
2.0	$(6, 8, 5, 6)$	marron

TABLE 1 – Récapitulatif des paramètres d'expérimentation

La distance est la *distance moyenne de Kentall-tau* entre deux fonctions de rang [12]. Cette mesure calcule le nombre de *paires discordantes* entre réputation des agents et fiabilité des agents, divisant ensuite cette somme par le nombre de paires d'agents. Le *regret*, quant à lui, est la différence entre le gain qu'un agent aurait obtenu s'il avait interagit avec l'agent qui avait la meilleure réputation et le gain qu'il a effectivement obtenu en intergissant avec l'agent qui a été sélectionné [3, 21].

4.3 Résultats sur BetaReputation

Sur BetaReputation, quelle que soit la politique de sélection, le bruit et le délai pris séparément perturbent les performances comme attendu. Le bruit provoque une augmentation de la distance et du regret (voir figures 2 et 3), et le délai provoque un décalage dans la convergence de la distance et du délai (voir figures 4 et 5). Plus le bruit et le délai sont importants, plus cette perturbation est forte. Nous pouvons remarquer un effet contre-intuitif concernant l'effet du délai sur le regret de la politique UCB (voir figure 5.c) : le délai fait converger plus rapidement le regret et, plus le délai est important, plus l'augmentation du regret est faible. Ceci s'explique par le fait que le terme d'exploration de la politique UCB est facteur du nombre de témoignages.

Avec le délai, le terme d'exploitation prend le pas et le terme d'exploration n'arrive pas à le rattraper. La politique UCB devient alors proche d'une politique 0-gloutonne. La combinaison du bruit et du délai provoque naturellement à la fois une augmentation de la distance et du regret, ainsi qu'un décalage de leur convergence. Si les effets sur le regret sont assez semblables à ceux du délai seul, et en particulier pour la politique UCB, la figure 6.b montre que plus le bruit et le délai sont importants, plus vite le sysème converge vers une évaluation correcte des agents pour la politique β-softmax.

6. Deux agents a_i et a_j forment une paire discordante pour l'agent a_k si, et seulement si, $Rep_i^k > Rep_j^k$ et $f_i < f_j$.

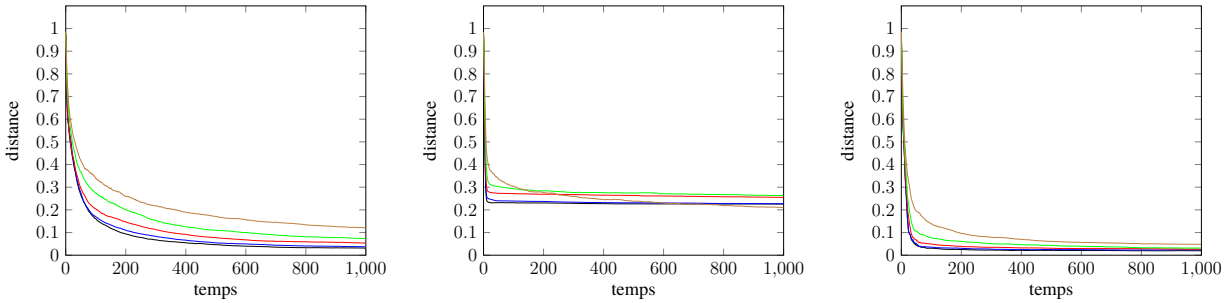
4.4 Résultats sur EigenTrust

Sur EigenTrust, le bruit et le délai ont des effets similaires mais plus contrastés à ceux sur BetaReputation. Dans certains cas, des effets contre-intuitifs ont lieu. En effet, EigenTrust est une fonction par rang et les valeurs de réputation des agents sont très proches les unes des autres. Cela conduit les politiques β-softmax et UCB à des comportements particuliers : sans un facteur β élevé la politique β-softmax se rapproche d'un tirage uniforme, et la politique UCB oscille car le facteur d'exploration domine périodiquement.

Les figures 9.a et 9.b montrent qu'un bruit modéré permet d'obtenir une meilleure évaluation des agents et, donc, un regret inférieur au regret sans confidentialisation. Comme les valeurs de réputation d'EigenTrust sont proches les unes des autres, de petites perturbations uniformes de ces valeurs ont pour effet de les disperser et de permettre une meilleure discrimination des agents. Enfin, pour les mêmes raisons que sur BetaReputation, un délai permet à la politique UCB, qui normalement oscille sans être efficace, de converger (voir figures 11.c et 13.c) . En effet, le délai diffuse les témoignages par paquets et une arrivée massive de témoignages valides discrimine subitement les agents, évitant au facteur d'exploration de dominer.

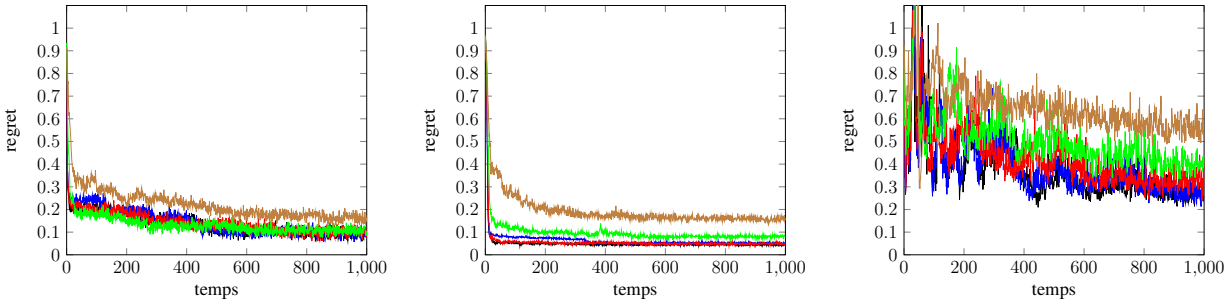
5 Conclusion

Pour conclure, les expérimentations montrent que la confidentialisation fondée sur un bruit additif et un délai de diffusion par agent peut être utilisable sur des systèmes de réputation de type BetaReputation à condition que les agents utilisent une politique de sélection de type ϵ-gloutonne. En effet, sur ce type de politique les effets du bruit et du délai sont les plus faibles et les plus attendus. De manière intéressante, le bruit et le délai peuvent être également utilisés sur EigenTrust pour améliorer les politiques ϵ-gloutonne et β-softmax ainsi que "régulariser" le comportement d'une politique UCB.



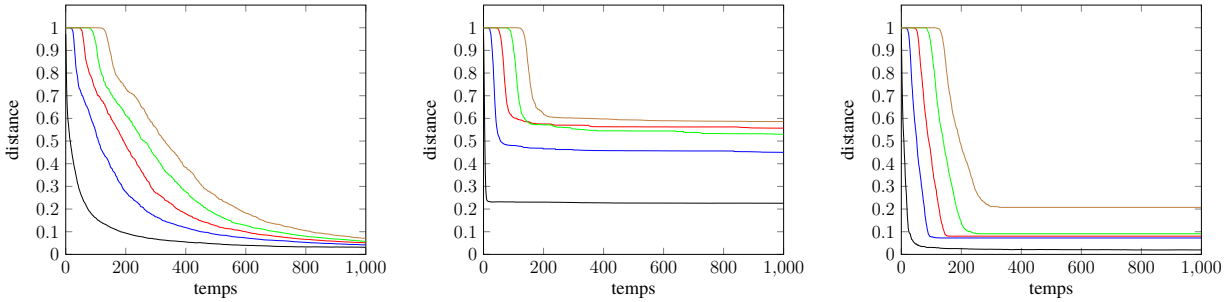
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 2 – Impact du bruit sur la distance pour BetaReputation



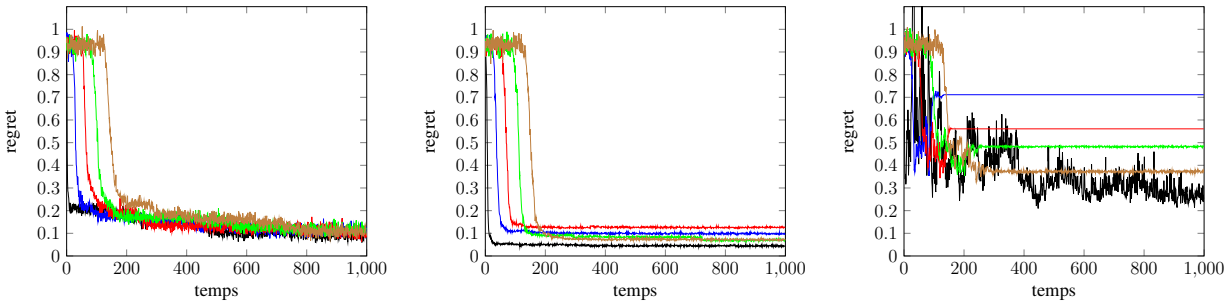
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 3 – Impact du bruit sur le regret pour BetaReputation



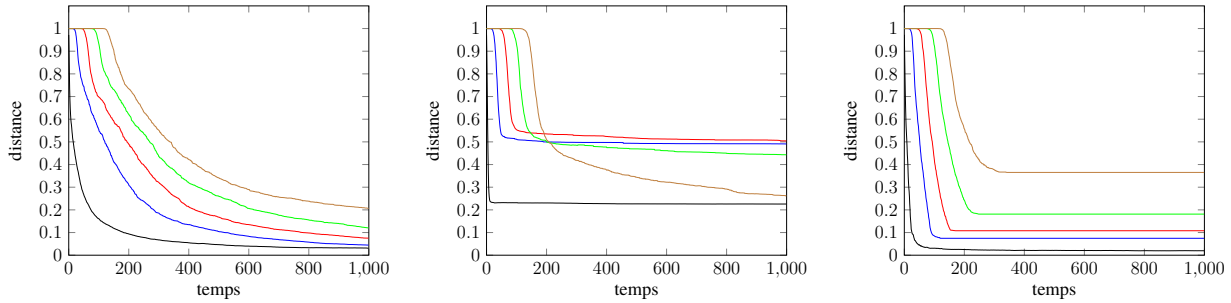
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 4 – Impact du délai sur la distance pour BetaReputation



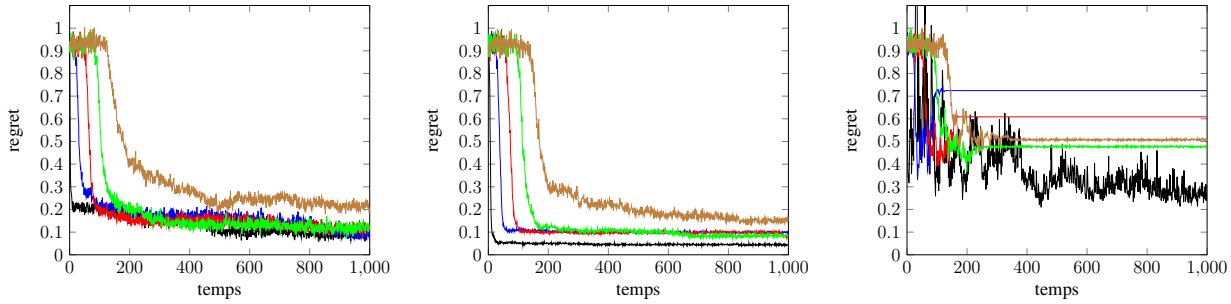
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 5 – Impact du délai sur le regret pour BetaReputation



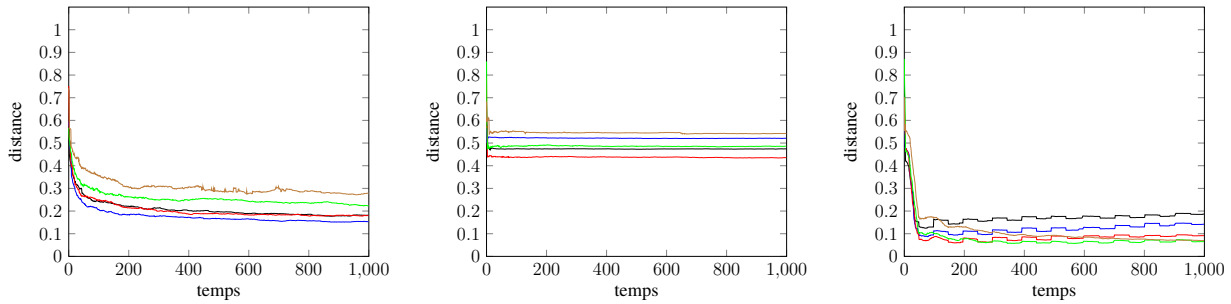
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 6 – Impact du bruit et du délai sur la distance pour BetaReputation



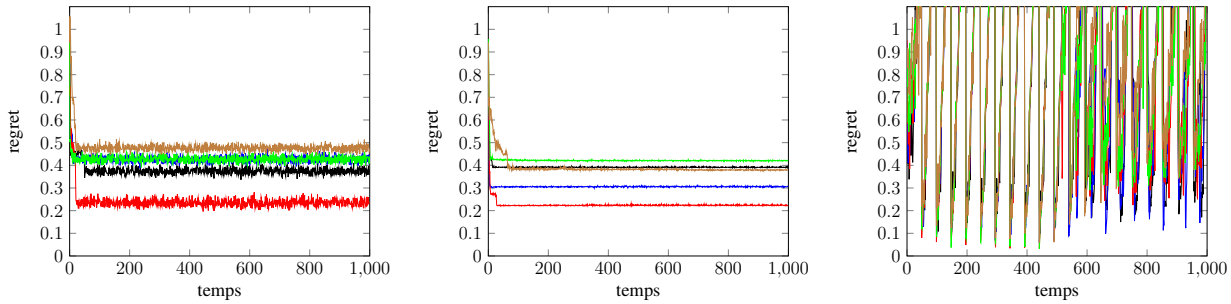
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 7 – Impact du bruit et du délai sur le regret pour BetaReputation



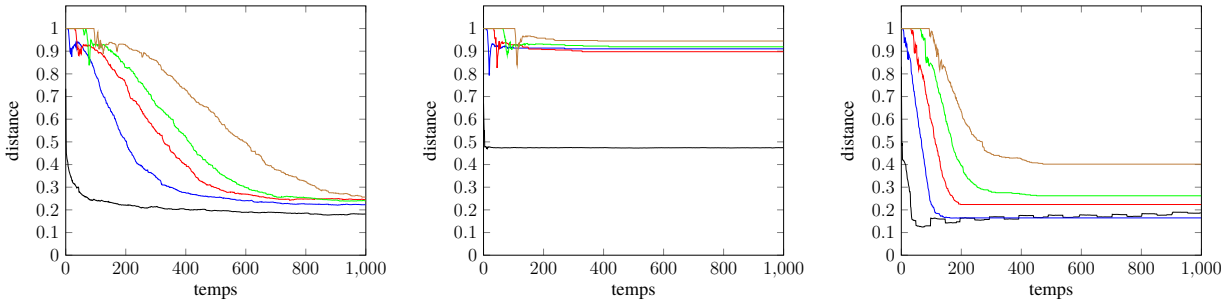
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 8 – Impact du bruit sur la distance pour EigenTrust



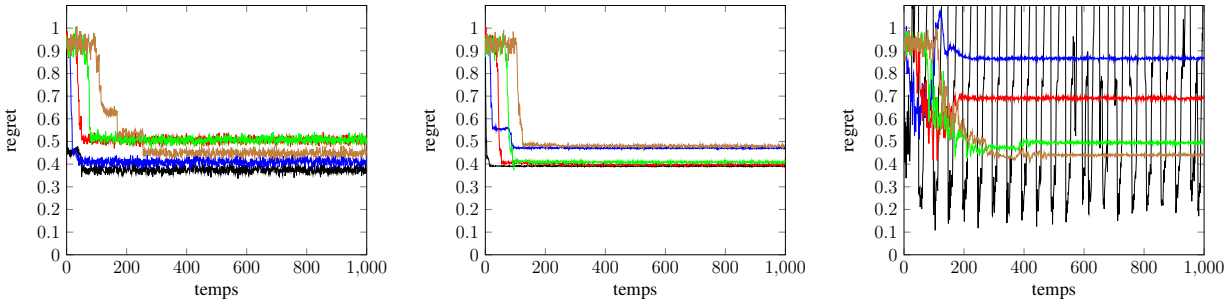
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 9 – Impact du bruit sur le regret pour EigenTrust



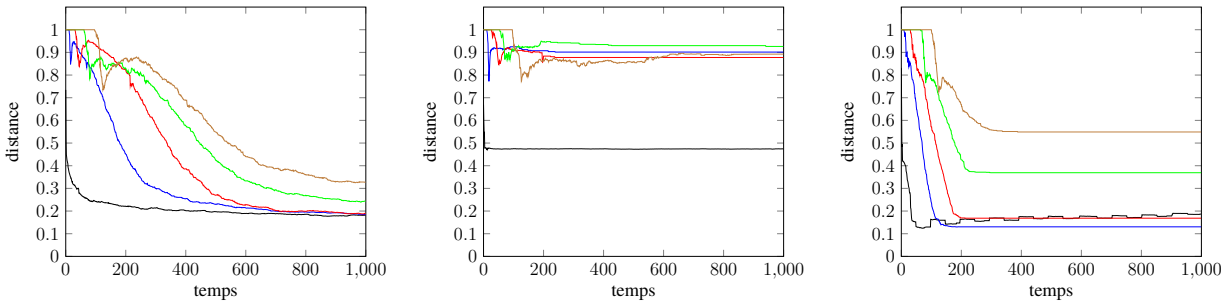
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 10 – Impact du délai sur la distance pour EigenTrust



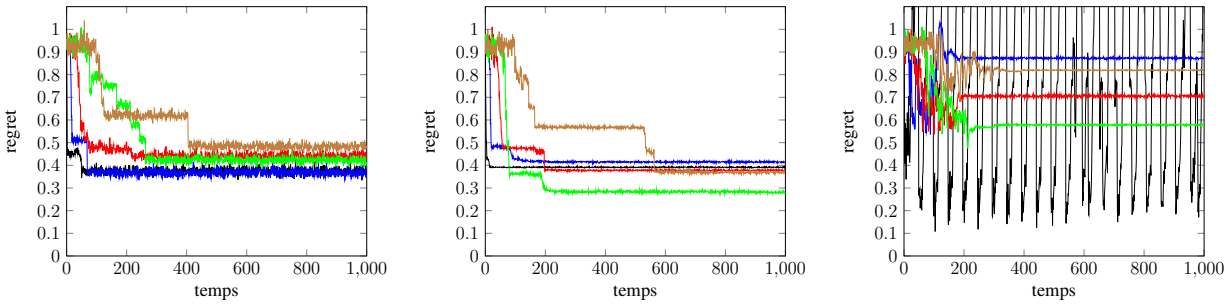
(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 11 – Impact du délai sur le regret pour EigenTrust



(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 12 – Impact du bruit et du délai sur la distance pour EigenTrust



(a) ϵ-gloutonne (b) β-softmax (c) UCB

FIGURE 13 – Impact du bruit et du délai sur le regret pour EigenTrust

Références

[1] Charu C. Aggarwal and Philip S. Yu. A general survey of privacy-preserving data mining models and algorithms. In *Privacy-Preserving Data Mining*, pages 11–52. Springer, 2008.

[2] Roberto Aringhieri, Ernesto Damiani, Sabine De Capitani Di Vimercati, Stefano Paraboschi, and Pierangelo Samarati. Fuzzy techniques for trust and reputation management in anonymous peer-to-peer systems. *Journal of the American Society for Information Science and Technology*, 57(4) :528–537, 2006.

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

[4] Javier Carbo, Jose M Molina, and Jorge Davila. Comparing predictions of SPORAS vs. a fuzzy reputation system. In *3rd International Conference on Fuzzy Sets and Fuzzy Systems*, volume 200, pages 147–153, 2002.

[5] Cynthia Dwork. Differential privacy. In *33rd International Colloquium on Automata, Languages and Programming*, pages 1–12, 2006.

[6] Tyrone Grandison and Morris Sloman. A survey of trust in Internet applications. *Communications Surveys & Tutorials, IEEE*, 3(4) :2–16, 2000.

[7] Omar Hasan, Lionel Brunie, and Elisa Bertino. Preserving privacy of feedback providers in decentralized reputation systems. *Computers and Security*, 31(7) :816 – 826, 2012.

[8] Kuan L. Huang, Salil S. Kanhere, and Wen Hu. A privacy-preserving reputation system for participatory sensing. In *37th Annual IEEE Conference on Local Computer Networks*, pages 10–18, 2012.

[9] Audun Jøsang and Roslan Ismail. The beta reputation system. In *15th Bled Conference on Electronic Commerce*, pages 324–337, 2002.

[10] Audun Jøsang, Roslan Ismail, and Colin Boyd. A survey of trust and reputation systems for online service provision. *Decision support systems*, 43(2) :618–644, 2007.

[11] Sepandar D. Kamvar, Mario T. Schlosser, and Hector Garcia-Molina. The eigentrust algorithm for reputation management in P2P networks. In *12th International Conference on World Wide Web*, pages 640–651, 2003.

[12] Maurice Kendall. A new measure of rank correlation. *Biometrika*, 30 :81–89, 1938.

[13] Ross A. Malaga. *Information Systems Research Methods, Epistemology, and Applications*, chapter The retaliatory feedback problem : evidence from eBay and a proposed solution, pages 342–349. Hershey, 2009.

[14] Zaki Malik and Athman Bouguettaya. Rateweb : Reputation assessment for trust establishment among web services. *International Journal on Very Large Data Bases*, 18(4) :885–911, 2009.

[15] Sergio Marti and Hector Garcia-Molina. Taxonomy of trust : Categorizing P2P reputation systems. *Computer Networks*, 50(4) :472–484, 2006.

[16] Kato Mivule. Utilizing noise addition for data privacy, an overview. *CoRR*, 2013.

[17] Xinlei Oscar Wang, Wei Cheng, Prasant Mohapatra, and Tarek Abdelzaher. ARTsense : Anonymous reputation and trust in participatory sensing. In *32nd International Conference on Computer Communications*, pages 2517–2525, 2013.

[18] Jordi Sabater, Mario Paolucci, and Rosaria Conte. Repage : Reputation and image among limited autonomous partners. *Journal of Artificial Societies and Social Simulation*, 9(2), 2006.

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

[20] Aameek Singh and Ling Liu. Trustme : Anonymous management of trust relationships in decentralized P2P systems. In *International Conference on Peer-to-Peer Computing*, page 1, 2003.

[21] Thibaut Vallée, Grégory Bonnet, and François Bourdon. Multi-armed bandit policies for reputation systems. In *13th International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 279–290, 2014.

[22] Runfang Zhou and Kai Hwang. Powertrust : A robust and scalable reputation system for trusted peer-to-peer computing. *IEEE Transactions on Parallel and Distributed Systems*, 18(4) :460–473, 2007.

Explicabilité et offuscation d'objectifs : un modèle pour la coopération et la confidentialité

N. Cointe
nicolas.cointe@tudelft.nl

Department of Engineering Systems and Services,
Faculty of Technology, Policy and Management,
Delft University of Technology, Pays-Bas

Résumé

Lorsque des agents agissent dans un système où leur comportement est au moins partiellement observable par d'autres agents, des techniques de reconnaissance de plans peuvent permettre d'inférer leurs objectifs et compromettre leur stratégie ainsi que la vie privée d'éventuels utilisateurs. Cet article se positionne du point de vue de l'agent observé et propose une fonction permettant à un agent d'évaluer la quantité d'information transmise au travers de son comportement à un éventuel observateur. Une telle fonction fournit une mesure de l'impact de la sélection d'un plan sur l'offuscation ou la transparence des objectifs de l'agent. Cet article explore enfin la problématique de la justification d'un comportement volontairement offusqué à des agents de confiance. Une preuve de concept accompagne la présentation de ce modèle afin d'illustrer sa mise en œuvre au sein d'un système composé d'agents BDI.

Mots-clés : *Confidentialité, Modèles de comportement agent, Planification*

Abstract

When agents act in a system where they are able to observe the behavior of the peers, plan recognition techniques might be used to reveal their goals, make their strategy no longer relevant and eventually threaten the privacy of the user. This paper provides a model to evaluate the amount of information leaked through the behavior of an agent in order to obfuscate her goals and eventually to justify an obfuscated behavior to a potential user or collaborator. A proof of concept is also provided to illustrate the implementation of this function in a BDI agent.

Keywords: *Privacy, Agent's behavior modeling, Planification*

1 Introduction

La reconnaissance de plans consiste en l'observation du comportement d'un agent et sa comparaison avec un ensemble de plans connus, afin d'évaluer l'ensemble de plans le plus vraisemblablement exécutés ou en cours d'exécution et les buts qu'ils permettent d'atteindre. De telles techniques permettent par exemple de contribuer à la sécurité d'un système en identifiant des agents ayant des objectifs dangereux [16] ou de reconnaître et contrer la stratégie d'un adversaire dans le cadre de jeux [10]. Bien qu'une telle analyse de comportements des agents d'un système puisse constituer un atout en matière de sécurité et permettre d'écarter d'éventuels agents malveillants, l'usage de ces techniques peut également relever du profilage[14] et constituer une menace pour la vie privée et le libre arbitre des utilisateurs du système.

Cet article propose de renverser la vision du problème pour se situer du point de vue de l'agent observé. Doter un agent d'une fonction permettant de mesurer l'impact de l'exécution d'une séquence d'actions sur les conclusions qu'un observateur pourrait en tirer semble constituer une nécessité dans des domaines applicatifs où ces enjeux de vie privée, de liberté et d'intérêt de la confidentialité pour assurer la viabilité d'une stratégie doivent être pris en compte dans la décision. Afin de maximiser ses chances de maintenir ses objectifs secrets et dans l'éventualité où un agent aurait la possibilité de choisir parmi plusieurs plans permettant de les atteindre, l'agent peut ainsi opter pour celui qui fournit les informations les moins pertinentes pour un observateur. Le modèle proposé a en outre été conçu avec le soucis de laisser à l'agent la possibilité d'expliquer sa décision (par exemple à un ensemble d'agents jugés dignes de confiance, un utilisateur humain ou une autorité). Les travaux présentés ici s'inspirent d'une proposition plus simple préalablement présentée [3]. Ici nous étendons cette idée à l'évaluation de successions de choix conduisant à l'exécution de séquences d'actions. Notre objectif est de rendre cette proposition opérationnelle sous la forme d'un module pouvant être intégré à un processus de dé-

cision existant, d'étendre la réflexion aux problèmes posés par les interactions entre agents et d'illustrer sa mise en œuvre par une preuve de concept.

La section 2 introduit le cadre conceptuel dans lequel nous définissons le problème et présentons notre approche. La section 3 expose ensuite le fonctionnement d'un processus d'évaluation de la transparence d'un comportement et présente son utilisation dans le cadre de la prise de décision. La mise en oeuvre de ce modèle dans le processus de décision d'un agent BDI est illustrée en Section 4 à l'aide d'une preuve de concept implémentée en Jason. La section 5 présente des travaux de sciences sociales en lien avec les stratégies d'offuscation et énonce des pistes de recherche pour la poursuite de travaux sur cette thématique.

2 Cadre conceptuel

Cette section introduit brièvement l'un des principaux paradigmes employés en reconnaissance de plans [10] opérée sur la base d'un réseau hiérarchique de tâches (HTN) [5] afin de présenter le point de vue d'un agent observateur du comportement d'un autre.

Une bibliothèque de plans est ici définie comme un tuple $L = \langle A, G, I, R\rangle$ (pour *plan Library*) avec A l'ensemble des actions, G un ensemble de buts, $I \subseteq G$ un ensemble de buts désirables (c'est-à-dire pouvant être considérés par l'agent comme des buts actifs) et R un ensemble de relations de compositions de la forme $g \leftarrow \tau$ signifiant que le but g peut être accompli en réalisant l'ensemble partiellement ordonné τ d'éléments de $A \cup G$. Par soucis de simplicité, nous considérons dans cet article que chaque but désirable correspond à un seul et unique plan.

Une bibliothèque de plans se représente habituellement sous forme d'un ensemble de structures arborescentes. Chaque arbre représente un plan ayant pour racine un but désirable de I, pour feuilles des actions de A et pour tout autre nœud des buts intermédiaires de G. Si tous ses fils doivent être accomplis pour considérer le but intermédiaire représenté par un nœud comme accompli, les liaisons avec ses fils sont reliées par un arc pour symboliser une conjonction. On parle alors de nœud-ET. Si R impose en plus un ordre sur l'accomplissement des fils, celui-ci est représenté à l'aide de flèches. Tout nœud qui n'est ni une feuille ni un nœud-ET est un nœud-OU.

La résolution d'un problème de reconnaissance de plans est défini [13] comme visant à associer à un couple $\langle L, b\rangle$, constitué d'une bibliothèque de plans L et d'un comportement b (défini comme un ensemble ordonné d'éléments de A), un ensemble d'explications consistantes. Une explication est définie comme l'association d'un but désirable à chaque action d'un comportement, conformément à la structure des plans présents dans la bibliothèque. La résolution de tels problèmes a fait l'objet de nombreuses propositions d'algorithmes tels que ELEXIR [6], PHATT [7], YAPPR [8], DOPLAR [10] et SLIM [12].

Une approche naïve consisterait à vérifier, pour toute combinaison d'associations d'actions d'un comportement à un but désirable, si cette combinaison constitue une explication consistante. L'espace à explorer étant de l'ordre de $\mathcal{O}(|I|^{|b|})$, une telle approche souffrirait évidement d'une explosion combinatoire. Certaines propositions de la littérature tirent parti d'une construction incrémentale des explications (au fur et à mesure des observation) [1] et/ou cherchent à élaguer dans une structure arborescente de construction des résultats, les branches représentant les explications évaluées comme étant moins "vraisemblables" [10].

L'évaluation de la *vraisemblance* d'une explication peut varier d'une approche à l'autre et se base généralement dans les travaux mentionnés précédemment sur le principe du rasoir d'Ockham : plus le nombre de buts désirables différents associés aux actions dans une explication est faible, plus l'observateur la considère comme vraisemblable. À la notion de vraisemblance d'une explication vient s'ajouter la question de la proportion de l'emploi de chaque but désirable dans l'ensemble des explications. En combinant ces deux critères, les techniques de reconnaissance de plan permettent d'évaluer la probabilité de chaque plan et d'en déduire le *plan le plus probable* (ou *MPP* pour *Most Probable Plan*).

Exemple Considérons un exemple illustratif volontairement simple dans lequel les agents représentent des humains travaillant au second étage d'un bâtiment. Il sont ici capables d'utiliser leur fonction de perception pour connaître les actions effectuées par les agents se trouvant sur le même étage et partagent la bibliothèque de plans L.

Nous employons dans cet exemple trois but désirables distincts pouvant inciter les agents à quitter ce deuxième étage pour aller prendre un en-

cas, faire une pause café ou améliorer l'état de l'art de leur prochain article en allant emprunter un livre. La bibliothèque de plans partagée décrit les connaissances dont disposent les agents pour en déduire les comportements permettant d'atteindre ces buts.

La bibliothèque de plans L contient les éléments suivants :

A, l'ensemble des actions, tel que $A = \{$ Go to ground floor, Go to second floor, Go to third floor, Buy a coffee, Drink a coffee, Buy a snack, Eat a snack, Take a book, Read the book $\}$;

G, l'ensemble des buts tel que $G = \{$ Have a coffee break, Find a coffee machine, Enjoy coffee anywhere, Have a snack, Find a vending machine, Enjoy snack anywhere, Improve the state-of-the-art, Read anywhere $\}$;

$I \subseteq G$, l'ensemble des buts désirables, tel que $I = \{$ Have a coffee break, Have a snack, Improve the state-of-the-art $\}$;

R, l'ensemble des règles de composition. Bien que l'exemple employé ici soit simple, l'ensemble des règles de composition ne sera pas détaillé intégralement ici par soucis de concision. Il contient des règles telles que

Have a coffee break $\leftarrow$ [[Find a coffee machine, Buy a coffee, Enjoy coffee anywhere], $\{(1,2),(2,3)\}$]

signifiant que le but "Have a coffee break" peut être atteint en atteignant ou exécutant les éléments de l'ensemble $\beta = \{$Find a coffee machine, Buy a coffee, Enjoy coffee anywhere$\}$, dans tout ordre consistant avec l'ensemble de contraintes $C = \{(1,2),(2,3)\}$ représentant un ensemble de relations de précédence entre des éléments représentés par leur indice dans la liste. Cette bibliothèque de plans inclut également des alternatives possibles dans l'exécution des plans telles qu'exprimées par la paire de règles de décompositions suivantes :

Find a coffee machine $\leftarrow$ [[Go to ground floor], $\varnothing$]

Find a coffee machine $\leftarrow$ [[Go to third floor], $\varnothing$]

signifiant que le but "Find a coffee machine" peut être accompli en effectuant l'une des deux actions mentionnées ici.

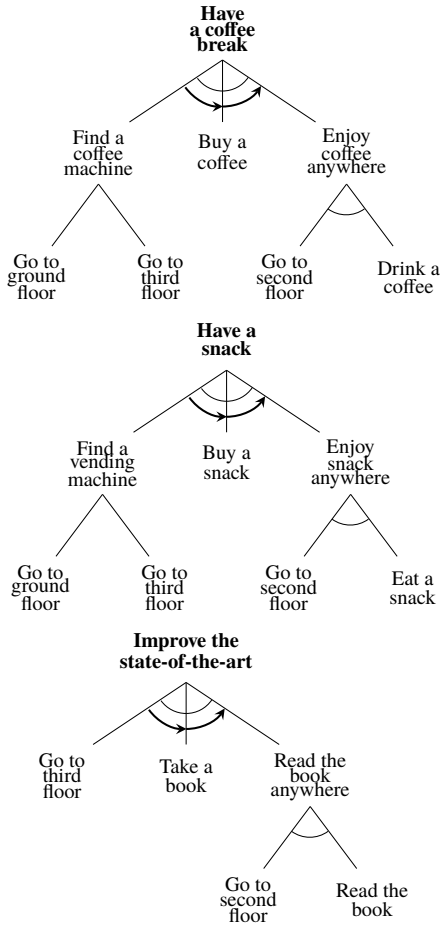


FIGURE 1 – Exemple de bibliothèque de plans

La figure 1 illustre la bibliothèque de plans mentionnée dans cet exemple sous la forme d'une forêt d'arbres de plans. Le premier plan décrit par exemple comment prendre une pause café en se rendant à la machine à café au rez-de-chaussée ou au troisième étage, en achetant son café, puis en le buvant avant ou après être revenu au bureau situé au deuxième étage.

Considérons le point de vue d'un agent observateur dans un système où les actions sont partiellement observables et la bibliothèque de plans illustrée dans la figure 1 est commune à l'ensemble des agents. Supposons que l'agent observateur perçoit les deux comportements suivants : $b_1 = \{$ Go to ground floor, Go to second floor $\}$ et $b_2 = \{$ Go to third floor, Go to second floor, Read the book $\}$.

Dans la bibliothèque de plans, les deux pre-

miers des trois plans peuvent individuellement mener à effectuer les actions mentionnées dans b_1 dans cet ordre puisqu'ils ont ces actions pour feuilles, que celles-ci ne sont pas dans des sous-arbres distincts reliés par un nœud-ou et que les contraintes d'ordre sont respectées. En revanche pour b_2 seul le troisième plan peut à lui seul être employé dans une explication.

Notons qu'il existe pour chacun de ces comportements d'autres explications moins *vraisemblables* : par exemple {[Go to ground floor, Have a coffee break], [Go to second floor, Have a snack]} est une explication associant chaque action à des buts désirables différents.

3 Évaluation de la transparence d'un plan

Nous cherchons ici à définir un moyen de mesurer la quantité d'information transmise à un autre agent à travers l'observation d'un comportement et permettant d'inférer les buts de l'agent observé. Nous supposons tout d'abord que l'agent observateur est doté d'une *fonction d'explication* FE permettant de produire l'ensemble $\mathcal{E}_b$ des explications d'un comportement observé b au regard d'une bibliothèque de plans L, ainsi que d'une fonction d'évaluation de la probabilité d'un plan associant à tout plan $ig \in IG$ une probabilité $P(ig|b, L)$.

Afin d'évaluer la quantité d'information transmise au travers d'un comportement et au regard d'une bibliothèque de plans, nous proposons de calculer l'entropie de Shannon [18] (définie sur $[0; 1]$) sur la distribution des probabilités de chaque plan :

$$H_{b,L} = -\sum_{ig \in IG} P(ig|b, L) \cdot log(P(ig|b, L))$$

Ainsi, un comportement dont l'ensemble $\mathcal{E}_b$ des explications ne peut associer les actions qu'à un seul plan obtiendra une entropie $H_{b,L}$ nulle, signifiant que le comportement observé transmet un maximum d'information sur les buts de l'agent. Notons que ce cas extrême peut ne pas être atteignable dans le cas d'un agent dont le comportement vise à atteindre un objectif si le plan associé nécessite l'exécution d'une action, ou d'une séquence d'actions, commune à plusieurs plans. Nous qualifions dans la suite de cet article le comportement d'un agent dont résulte la plus faible entropie possible compte tenu de la bibliothèque de plans de *transparent*.

À l'inverse, un ensemble d'explications associant de manière strictement égale l'ensemble des but désirables aux actions du comportement observé conduit à une entropie maximale, signifiant que la quantité d'information transmise par ce comportement, au regard de la bibliothèque L employée, est nulle. Le caractère réalisable d'un comportement menant à la réalisation d'un objectif et produisant une entropie maximale est également dépendant de la bibliothèque de plans. Nous qualifions dans la suite le comportement d'un agent dont résulte une entropie aussi grande que possible d'*offusqué*.

Le cœur de notre proposition, est de permettre à un agent d'observer son propre comportement et, dans son processus de décision, de prendre en compte la quantité d'information transmise à un éventuel observateur afin de maximiser la transparence ou au contraire l'offuscation de ses buts.

4 Preuve de concept

Nous cherchons maintenant à illustrer la mise en œuvre de notre proposition au sein d'une architecture d'agent existante et permettre d'évaluer son impact sur le comportement effectif des agents. Nous nous sommes tournés vers l'architecture BDI [17] pour l'aisance apportée à l'emploi des notions de but et de plan. Cette section présente une implémentation de l'exemple présenté plus haut à l'aide de Jason et Cartago, faisant partie du cadriciel JaCaMo [2]. Le code source complet est accessible en ligne .

4.1 Présentation générale du système

Dans JaCaMo, les plans, règles d'inférence, croyances et buts des agents sont décrits à l'aide du langage AgentSpeak [19]. Ici tout élément de G (et non pas seulement l'ensemble des buts désirables IG) sera implémenté sous la forme de but en utilisant les mécanismes proposés par Jason pour les activer et les atteindre. L'architecture d'agent proposée par défaut fournit les mécanismes internes permettant entre autre de percevoir l'environnement, sélectionner des plans de manière proactive et réactive, assurer leur exécution au sein d'un ensemble d'intentions courantes pour atteindre des buts, et communiquer avec les autres agents .

1. http://www.nicolascointe.eu/projects/

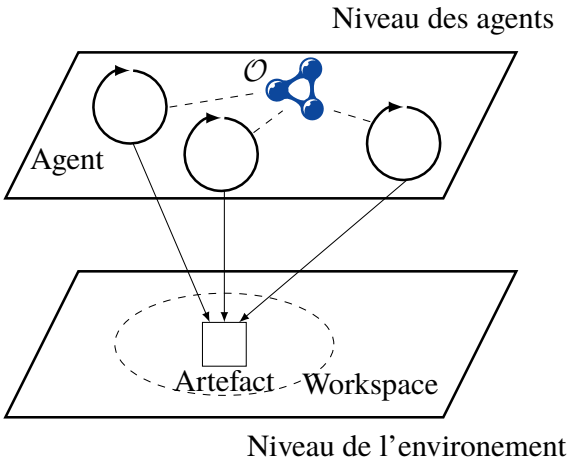


FIGURE 2 – Structure de la preuve de concept

La figure 2 illustre la conception de ce système dans le cadre de notre preuve de concept, avec un environnement partagé comprenant un unique artefact permettant à l'ensemble des agents d'agir et percevoir les actions des autres. En se liant à l'artefact, les agents obtiennent la possibilité d'exécuter l'ensemble A des actions de la bibliothèque de plans. Lors de l'exécution d'une action, l'artefact émet un signal vers la fonction de perception de chaque agent lié. Les agents partagent également un ensemble de connaissances $\mathcal{O}$, contenant la bibliothèque de plans L. Par simplicité, $\mathcal{O}$ est ici implémenté comme un ensemble de croyances, règles et plans donné à chaque agent lors de son initialisation.

Nous symbolisons ici les agents par des boucles pour rappeler leur processus interne continuellement à l'écoute d'événements et à la recherche de plans à sélectionner pour atteindre les buts.

Les agents introduits dans ce système ne sont pas homogènes. Un premier ensemble d'agents comporte des agents aléatoires, explorant les divers comportements possibles pour atteindre les buts qui leurs sont assignés sans tenir compte de l'information transmise par leur comportement. Deux autres ensembles d'agents, *offuscateurs* et *transparents* cherchent respectivement à maximiser et minimiser la quantité d'information transmise à travers leur comportement. Enfin un agent observateur réagit aux comportements de l'ensemble des agents du système et exprime les résultats afin de nous permettre de visualiser l'évolution de ses évaluations.

4.2 Représentation de la bibliothèque de plans

L'exécution d'un plan pouvant être accompli par des comportements différents en raison des disjonctions, les agents ont besoin d'être capable de manipuler une représentation mentale des plans de la bibliothèque, tant pour planifier des actions futures qu'évaluer un comportement observé. L'ensemble partagé de connaissances $\mathcal{O}$ contient un ensemble de croyances formaté tel que l'illustre l'extrait d'implémentation 1.

```
1  // Nom des feuilles (correspond
       aux signaux)
2  leaf(buySnack,"buySnack").
3
4  //****** Premier plan ******
5  isIntendableGoal(haveSnack,
6     "haveSnack").
7
8  isSonOf(buySnack,haveSnack,
       haveSnack).
9
10 // Ordre sur les AND-nodes
11 orderConstraint(findVendingMachine
       ,buySnack,haveSnack).
12
13 // OR-nodes
14 exclusionConstraint(
       goToGroundFloor,goToThirdFloor,
       haveSnack).
```

Implementation 1 – Extrait de planForestDescriptor.asl

Chaque feuille de l'ensemble des arbres de la bibliothèque de plan est décrit comme illustré à la seconde ligne de l'exemple, par son symbole employé dans le raisonnement de l'agent et le signal correspondant émis par l'artefact lors de l'exécution de l'action. Les buts désirables sont représentés de manière similaire.

L'ensemble des relations filiales des arbres, du but désirable situé à la racine aux actions représentées par les feuilles sont décrites par des croyances de la forme `isSonOf(A,B,T)` tel que l'illustre la ligne 8. Ce prédicat signifie que `A` est un enfant de `B` dans l'arbre de racine `T`.

Les contraintes d'ordre sur les nœuds-ET sont aussi représentées par un ensemble de croyances de la forme `orderConstraint(A,B,T)` tels qu'illustré à la ligne onze, et signifiant que le sous-arbre ayant pour racine `A` doit être exécuté avant celui ayant pour racine `B` dans l'arbre de racine `T`. De manière analogue, les nœuds-OU sont représentés par une contrainte d'exclusion entre deux sous-arbres d'un arbre identifié.

Alors que l'ensemble de croyances présenté ci-avant décrit les relations d'arborescence entre les nœuds de la bibliothèque de plans pour permettre à l'agent de raisonner sur la structure elle-même, l'ensemble de connaissances $\mathcal{O}$ contient également un second fichier décrivant le contenu de ces nœuds sous la forme de plans BDI afin de permettre leur parcours et l'exécution des actions par le processus de sélection des intentions. L'extrait d'implémentation 2 illustre la description des nœuds au sein de plans BDI.

```
1  +!haveSnack :
2      decideIn(Choice,[goToThirdFloor,
         goToGroundFloor])
3      <- !findVendingMachine(Choice);
4          !buySnack;
5          !enjoySnackAnywhere;
6          !finishIfIGotSnack.
7
8  +!findVendingMachine(
         goToGroundFloor) :
9      baseWaitingTime(T)
10     <- .my_name(Name);
11         goToGroundFloor(Name);
12         .wait(T);
13         .print("I move to the ground
         floor").
14
15 +!findVendingMachine(
         goToThirdFloor) :
16     baseWaitingTime(T)
17     <- .my_name(Name);
18         goToThirdFloor(Name);
19         .wait(T);
20         .print("I move to the third
         floor").
```

Implementation 2 – Extrait de planLib.asl

Le premier plan fait appel au prédicat `decideIn(C,L)` permettant à l'agent de sélectionner un choix de nœud `C` parmi une liste de nœuds `L`. L'implémentation de ce prédicat permet de définir la stratégie employée par les agents. Lorsqu'un plan représentant un nœud feuille est exécuté, l'action qu'il comporte est effectuée au travers de l'artefact placé dans l'environnement.

4.3 Évaluation d'un comportement

Lors de l'exécution d'une action, l'artefact émet un signal en direction de tous les agents mentionnant le nom de l'agent effectuant l'action et l'action elle-même. En réaction à ce signal, l'agent met à jour une croyance `behavior(A,B)` où `A` est le nom de l'auteur de l'action et `B` le comportement observé sous la forme d'une liste d'actions associées à la date de leur exécution.

```
1  +!reactToBehavior(Agent) :
       behavior(Agent,Behavior)
2       <- !
       detailBehaviorExplanatihEART
       2018on(Agent,Behavior).
3
4  +!detailBehaviorExplanation(Agent,
       Behavior):
5     listOfIntendableGoals(IGs)
6     & time(T)
7     & exploreExplanations(Behavior,
       [],Exp, IGs)
8     & countPlansInExp(Exp,[],
       PlansOccurrences)
9     & searchMostProbableGoal(
       PlansOccurrences,P,N)
10    &
11 getSpecificCountPlanInExplanationSet
       (haveCoffeeBreak,
       PlansOccurrences,
       NofHaveCoffeeBreak)
12    &
13 getSpecificCountPlanInExplanationSet
       (haveSnack,PlansOccurrences,
       NofHaveSnack)
14    &
15 getSpecificCountPlanInExplanationSet
       (improveSOTA,PlansOccurrences,
       NofImproveSOTA)
16    <-   .length(Exp, L);
17         jia.shannonEntropy(H,
       NofHaveCoffeeBreak,NofHaveSnack
       ,NofImproveSOTA);
18         .print("Agent=", Agent, " T=
       ", T," MPG=", P," Entropy=", H,
        "(", NofHaveCoffeeBreak, ",",
        NofHaveSnack, ",",
       NofImproveSOTA, ")");
19         .print("I can explain the
       behavior ", Behavior, " of
       agent ", Agent, " with the ", L
        ," possible explanations : ",
       Exp).
```

Implementation 3 – Extrait de onlooker.asl

Les agents observateurs analysent les comportements observés à l'aide du plan présenté dans l'exemple d'implémentation 3. Le prédicat de la forme `exploreExplanations(B, [],Exp, IGs)` sur la septième ligne insère dans la liste `Exp` l'ensemble des explications $\mathcal{E}$ présenté en section 3.

À la ligne suivante, le prédicat de la forme `countPlansInExp(Exp,[],PlansOcc)` unifie `PlansOcc` avec une liste de tuples associant à chaque but désirable son nombre d'occurrences dans l'ensemble des explications. Le prédicat `searchMostProbableGoal(PlansOcc,P,N)` unifie `P` avec le nom de la racine du plan le

plus probable et affiche les résultats pour leur analyse.

4.4 Intégration dans un processus de décision

Comme mentionné précédemment lors de la description de la bibliothèque de plans, la définition du `decideIn(C,L)` dépend de la stratégie implémentée dans chaque type d'agent.

```
1 decideIn(Choice, Options) :-
2   myName(Name)
3   & myGoal(Tree)
4   & time(T)
5   & behavior(Name, Behavior)
6   & makeEntropyList(T, Options, Tree
      , Behavior, EntropyList)
7   & pickTheMostObfuscatingOptionIn
      (EntropyList, Choice, _).
```

Implementation 4 – Extrait de obfuscator.asl

Les *agents aléatoires* tirent simplement au hasard un élément `C` dans la liste des choix proposés `L`.

Les *agents offuscateurs* sont initialisés avec un but désirable à atteindre et emploient le prédicat présenté dans l'implémentation 4 pour prendre leur décision. Le prédicat mentionné à la sixième ligne, de la forme `makeEntropyList(T,Options,Tree,B,HL)` unifie `HL` avec une liste associant à chaque élément de la liste `Options` l'entropie estimée en cas d'exécution du sous-arbre dont il est la racine. Enfin, le dernier prédicat unifie `Choice` avec l'option associée à la valeur maximale de cette liste.

Les *agents transparents* utilisent un processus similaire à celui des agents offuscateurs, en sélectionnant l'option associée à l'entropie minimale.

4.5 Analyse des résultats

La simulation est initialisée avec trois agents aléatoires, trois agents offuscateurs et trois agents transparents assignés avec pour chaque catégorie un but désirable différent à atteindre pour chaque agent. Un agent observateur est ajouté pour observer et évaluer l'ensemble des comportements des agents. L'expérience est répétée plusieurs fois et les résultats sont agrégés afin de laisser les agents aléatoires explorer les différents comportements possibles.

La figure 3 illustre le comportement des agents qui se sont vus assigner le but désirable "Have a coffe break" du point de vue de l'agent observateur au cours de la simulation.

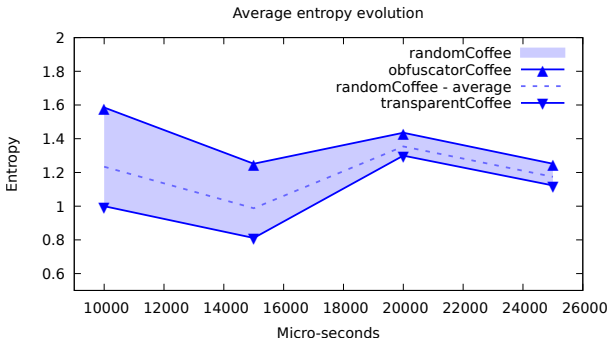


FIGURE 3 – Évolution de l'entropie dans la distribution de probabilité des plans au cours de la simulation

Comme attendu, l'entropie sur la distribution de probabilité entre les buts désirables décrit des courbes différentes d'une simulation à l'autre (l'espace exploré est représenté par la surface coloriée). Le front supérieur de cette aire est parcouru par les agents offuscateurs qui, grâce à leur processus de décision, minimisent les informations transmises à l'agent observateur. À l'inverse, l'entropie associée au comportement des agents transparents reste sur la frontière inférieure décrite par l'ensemble des comportements possibles. Nous proposons plusieurs preuves de concept employant les mécanismes de décision présentés ici avec des bibliothèques de plans plus conséquentes afin de faire la démonstration du caractère générique de la solution et permettre de produire des résultats issus de l'exécution de plans comprenant davantage de choix possibles et un nombre plus conséquent d'actions afin de mettre en lumière les propriétés du modèle, indépendamment des exemples employés. Nous cherchons à implémenter un générateur aléatoire de bibliothèques de plan afin de pouvoir vérifier si certaines propriétés observées (telle que la tendance à la diminution de l'entropie) sont inhérentes au modèle proposé ou liées à des caractéristiques des plans employés (proportion de nœuds-OU, tailles des arbres, proportion d'actions ou de sous-arbres communs à différents plans, etc.).

5 Offuscation, coopération et explicabilité

Si promouvoir la transparence au sein d'une société semble pratique, moral et intuitif, nous cherchons à mettre en lumière dans les sections suivantes des travaux de la littérature montrant l'intérêt de mécanismes d'offuscation dans la modélisation du comportement humain ou pour la conception d'agents autonomes artificiels. Il s'agit essentiellement de dresser ici une rapide présentation des différentes perspectives envisagées pour la suite de ces travaux.

La section 5.1 présente quelques travaux de sciences humaines s'intéressant à la tendance chez l'être humain à éviter volontairement de connaître, dans certaines circonstances, les buts des autres. Nous montrons abordons la possibilité offerte par notre contribution d'envisager dans de futurs travaux un modèle de coopération conditionnée par l'offuscation afin de modéliser ce type d'attentes. Ensuite la section 5.2 met en lumière l'intérêt du cadre conceptuel présenté ici pour gérer l'explicabilité d'un comportement au sein d'une organisation d'agents comme un atout stratégique.

5.1 Évitement d'informations et coopération conditionnée par l'offuscation

Nous considérons ici le cas dans lequel les agents doivent interagir dans un environnement partagé au sein duquel ils sont capables de percevoir, au moins partiellement, le comportement des autres. Disposant de telles observations, les agents peuvent volontairement éviter de coopérer avec ceux qui n'assurent pas un niveau minimum d'offuscation de leurs objectifs. Une telle restriction prend par exemple tout son sens si l'agent considère que la protection de la vie privée est une valeur morale nécessaire dans le cadre d'un mécanisme de coopération fondée sur l'éthique [4], ou bien que cela constitue un atout stratégique. Cela peut par exemple constituer un atout pour une communauté d'agents cherchant à éviter des punitions de la part d'une autorité tyrannique interdisant la poursuite de certains objectifs.

En économie, le concept d'*évitement actif d'information* (ou AIA pour *Active Information Avoidance*) [9] est défini comme la tendance que peut avoir un agent, dans le cas où il sait qu'une information est disponible sans coût, à refuser de la percevoir. La littérature autours de ce concept mentionne diverses motivations possibles tel que l'évitement de la déception ou du regret, le maintien d'un certain optimisme ou l'évitement de dissonance (pouvant être comparé à un problème de consistance des croyances chez les agents artificiels). De telles attentes peuvent conduire un humain à opter pour une coopération conditionnée par l'offuscation telle que nous l'avons décrite ici. Au sein d'un système multi-agent dans lequel les agents peuvent recevoir des responsabilités (par exemple liées à un rôle dans une organisation ou en raison de l'impact que peut avoir leur comportement sur des utilisateurs humains) et ont besoin de collaborer avec d'autres agents pour atteindre leurs objectifs, il peut être préférable et stratégiquement pertinent d'imposer de telles conditions.

Le modèle de l'*ignorance rationnelle* [15] s'intéresse à la représentation de devoirs moraux dans lesquels un *homo œconomicus* peut considérer préférable d'ignorer des informations si cela tend à décroître l'image qu'il a de lui-même (définie comme une satisfaction engendrée par son comportement au regard d'un ensemble de devoirs).

Si l'évitement de la perception d'information sur le comportement non-éthique d'un collaborateur peut être un moyen pour l'être humain d'éviter un phénomène de *dissonance cognitive* [15], c'est-à-dire une contradiction dans ses motivations, ou de la culpabilité, cela ne permet en revanche pas d'éviter les conséquences sociales telles que l'accusation de négligence ou de collusion avec un agent malicieux. Pour remédier à ce problème l'agent cherchant à éviter la dissonance et à mimiser le risque social encouru pourrait exiger de son collaborateur un niveau minimum d'offuscation de ses objectifs.

Notons que bien que ce modèle permette dans une certaine mesure de reproduire des comportements mentionnés dans des travaux reconnus d'économie et de psychologie, nous n'avons pas cherché dans cet article à proposer une modélisation proche du fonctionnement de la cognition humaine.

5.2 L'explicabilité comme un atout stratégique pour une organisation

Supposons un système dans lequel des agents sont autorisés à coopérer, capables de contrer certains plans des autres agents et ont une motivation pour le faire (par exemple en cas de jeux coopératifs à somme nulle). Si un agent est engagé dans une coopération, une mise en échec de

ses plans pourrait impacter les chances de réussite de sa coalition. Dans ce type de configurations, offusquer ses objectifs et opter pour des stratégies plus difficilement identifiables peut constituer un atout évident. En contrepartie, être capable de fournir une explication de son comportement aux autres agents d'une coalition peut être également un outils permettant de diminuer les risques de trahison et faciliter la coordination. La possibilité d'évaluer la vraisemblance de l'explication d'un comportement au regard d'une connaissance commune pourrait également contribuer à assurer la pertinence de cette collaboration. L'emploi de l'offuscation combiné à la possibilité de laisser les agents expliquer leur comportement aux autres membres d'un collectif permettrait d'établir un mécanisme de gestion des informations stratégiques et semble constituer une piste intéressante dans le cadre de la poursuite de ces travaux.

Si l'agent est en interaction et/ou en collaboration avec un être humain, l'offuscation de son comportement pourrait conduire à une expérience d'utilisateur perturbante, inattendue et contre-intuitive. La lisibilité d'une décision étant considérée comme un atout dans les interactions entre humain et agent autonome [11], il semble nécessaire d'expliquer les raisons de la sélection d'un plan à l'utilisateur. Il peut de même être pertinent d'alerter un utilisateur, et éventuellement stopper l'exécution de l'agent si celui-ci constate qu'il n'est plus capable d'assurer un niveau minimum d'offuscation dans la poursuite de ses actions. Cette fonctionnalité serait d'autant plus intéressante dans les cas où l'agent est amené à agir dans un contexte où son comportement peut révéler des informations sur la vie privée de l'utilisateur.

6 Conclusion

Nous avons présenté un processus permettant d'incorporer dans un agent BDI la possibilité d'évaluer la quantité d'information transmise au travers d'un comportement à un observateur en matière de reconnaissance de plans. Une telle fonction peut être à la fois utilisée pour identifier les buts probables d'un agent, mais aussi pour anticiper lors d'une prise de décision, la quantité d'information qui sera transmise à un informateur afin de maximiser la transparence du comportement ou au contraire d'accroître son offuscation. La mise en œuvre de ce modèle a été illustrée à l'aide d'une preuve de concept. Pour clôturer notre propos, nous avons exposé un ensemble de travaux montrant l'intérêt d'un tel modèle pour la simulation de certains comportements humains et la gestion de l'information dans un collectif d'agents, et présenté des pistes de réflexion pour les travaux futurs.

Au delà de nécessaires efforts de formalisation des idées évoquées ici, il serait intéressant d'explorer des variantes du modèle présenté, par exemple en cherchant à optimiser l'entropie du comportement en temps réel et non à terme, ou en permettant à un agent d'effectuer des actions issues d'autres plans dans le but de perturber les observations.

Nous pouvons aussi souligner la question posée par les cas dans lesquels le but ne peut être offusqué au-delà d'une certaine étape du plan. Par exemple si un observateur perçoit une action ne pouvant être associée au sein d'une explication qu'à un seul but désirable, il peut inférer que celui-ci fait manifestement partie des buts poursuivis par l'agent. L'agent observé pourrait alors considérer que l'offuscation de ses buts n'a plus d'intérêt ou décider d'opter pour un autre plan.

Remerciements

Ce travail a été réalisé grâce au support financier du Conseil Européen de la Recherche (ERC) au sein du programme recherche et innovation Horizon 2020 de l'Union Européenne (numéro d'agrément 724431).

Références

[1] Dorit Avrahami-Zilberbrand and Gal A Kaminka. Fast and complete symbolic plan recognition. In *IJCAI*, pages 653–658, 2005.

[2] 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.

[3] Caspar G Chorus. How to keep your AV on the right track ? An obfuscation-based model of decision-making by autonomous agents. In *7th Symposium of the European Association for Research in Transportation (hEART)*, 2018.

[4] Nicolas Cointe, Grégory Bonnet, and Olivier Boissier. Ethics-based cooperation in multi-agent systems. In *14th Social Simulation Conference (SSC18)*, august 2018.

[5] Kutluhan Erol. *Hierarchical task network planning : formalization, analysis, and implementation*. PhD thesis, 1996.

[6] Christopher W Geib. Delaying commitment in plan recognition using combinatory categorial grammars. In *IJCAI*, pages 1702–1707, 2009.

[7] Christopher W Geib and Robert P Goldman. Partial observability and probabilistic plan/goal recognition. In *Proceedings of the International workshop*

on modeling other agents from observations (MOO-05), volume 8, pages 1–6, 2005.

[8] Christopher W Geib, John Maraist, and Robert P Goldman. A new probabilistic plan recognition algorithm based on string rewriting. In *ICAPS*, pages 91–98, 2008.

[9] Russell Golman, David Hagmann, and George Loewenstein. Information avoidance. *Journal of Economic Literature*, 55(1) :96–135, 2017.

[10] Froduald Kabanza, Julien Filion, Abder Rezak Benaskeur, and Hengameh Irandoust. Controlling the hypothesis space in probabilistic plan recognition. In *IJCAI*, pages 2306–2312, 2013.

[11] Alexandra Kirsch. Explain to whom ? Putting the User in the Center of Explainable AI. In *Proceedings of the First International Workshop on Comprehensibility and Explanation in AI and ML 2017*, Bari, Italy, 2017.

[12] Reuth Mirsky et al. Slim : Semi-lazy inference mechanism for plan recognition. *arXiv preprint arXiv :1703.00838*, 2017.

[13] Reuth Mirsky, Roni Stern, Kobi Gal, and Meir Kalech. Sequential plan recognition : An iterative approach to disambiguating between hypotheses. *Artificial Intelligence*, 260 :51–73, 2018.

[14] Commission Nationale de l'Informatique et des Libertés (CNIL). Profilage et décision entièrement automatisée, mai 2018. `https://www.cnil.fr/node/24408`.

[15] Karine Nyborg. I don't want to hear about it : Rational ignorance among duty-oriented consumers. *Journal of Economic Behavior & Organization*, 79(3) :263–274, 2011.

[16] Jérémy Patrix, Abdel-Illah Mouaddib, Simon Le Gloannec, Dafni Stampouli, and Marc Contat. Discrete relative states to learn and recognize goals-based behaviors of groups. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 933–940. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[17] A.S. Rao and M.P. Georgeff. BDI agents : From theory to practice. In *1st International Conference on Multiagent Systems*, pages 312–319, 1995.

[18] Claude Elwood Shannon. A mathematical theory of communication. *Bell system technical journal*, 27(3) :379–423, 1948.

[19] Renata Vieira, Álvaro F Moreira, Michael Wooldridge, and Rafael H Bordini. On the formal semantics of speech-act based communication in an agent-oriented programming language. *Journal of Artificial Intelligence Research*, 29 :221–267, 2007.

Une représentation hiérarchique de comportements agents pour l'apprentissage progressif et continu

François Suro suro@lirmm.fr
Jacques Ferber ferber@lirmm.fr
Tiberiu Stratulat stratulat@lirmm.fr
Fabien Michel fmichel@lirmm.fr

LIRMM, Université de Montpellier, CNRS, Montpellier, France

Résumé

Dans la perspective d'un développement ouvert et continu il est cruciale qu'un comportement final souhaité à un instant donné puisse être un élément pour la création future de comportements plus complexes, dont le but ne peut être anticipé. Nous proposons MIND (Modular Influence Network Design), une architecture de contrôle pour agent artificiel conçue pour apprendre des comportements à partir d'un Curriculum établi par un instructeur humain. MIND encapsule les comportements dans des modules et les combine en une hiérarchie reflétant la nature modulaire et hiérarchique des tâches complexes, et permet la préservation et la réutilisation des compétences acquises.

Mots-clés : *Robotique développementale, Apprentissage agent, Architecture modulaire, Architecture hiérarchique, Apprentissage par Curriculum*

Abstract

In open ended and continuous development the desired behaviour at a given time can be an element of future behaviours of greater complexity, the purpose of which cannot be anticipated. MIND (Modular Influence Network Design) is a control architecture for artificial agents designed to learn behaviours from a curriculum developed by a human instructor by encapsulation into modules. These skill modules are combined into a hierarchy that perform behaviours of greater complexity while allowing the preservation and reuse of acquired skills.

Keywords: *Developmental robotics, Agent learning, Modular architecture, Hierarchical architecture, Curriculum learning*

1 Introduction

La théorie de Jean Piaget [13][14] sur le développement cognitif chez l'homme comme processus dynamique de schémas de coordination en plusieurs étapes (depuis les schémas sensorimoteurs jusqu'aux opérations de niveau abstrait) est un point de départ pour toute recherche sur la robotique épigénétique [9]. L'apprentissage se fait progressivement par l'interaction entre l'enfant et son environnement, des tâches plus complexes s'ajoutant à des tâches plus simples. Cette idée a influencé le domaine de la robotique développementale, où il est fondamental que la complexité des connaissances et des compétences acquises par un agent artificiel augmente progressivement [11]. Selon Piaget, la complexité de l'apprentissage peut évoluer selon trois axes :

1. La complexité de l'environnement peut croître progressivement, du bac à sable aux situations réelles.
2. Les motivations peuvent être de plus en plus complexes, de saisir un objet à la construction d'une maison.
3. Les compétences et leur structuration en comportements doivent se développer pour faire face à l'augmentation de la complexité de l'environnement et des motivations.

Depuis les premiers pionniers de la Renaissance qui ont identifié des zones du cerveau humain liées à des fonctions spécifiques jusqu'aux travaux plus récents sur la coordination entre ces zones, la recherche montre que les systèmes biologiques utilisent une approche modulaire [6]. Différentes zones du cerveau sont dédiées à des processus spécifiques et organisées en modules pour accomplir des tâches. Les travaux sur le cortex visuel des primates visant à créer un pont entre la biologie et la vision par ordinateur ont mis en évidence la structure hiérarchique du cerveau et la nécessité d'une conception hiérarchique en vision par ordinateur [8].

Les systèmes de vision par ordinateur tirent parti de structures hiérarchiques pour identifier des objets dans un monde structuré hiérarchiquement, dont objets peuvent naturellement être divisés en parties et sous-parties, caractéristiques complexes et caractéristiques simples [8]. De même, un système de représentation de comportements bénéficierait lui aussi d'une structure hiérarchique pour décrire des comportements complexes qui peuvent être naturellement divisés en parties et sous-parties, à la manière d'un algorithme.

Dans cet article, nous visons à fournir une représentation des connaissances pour les agents artificiels apprenant d'un instructeur à travers un Curriculum, ou cursus, soigneusement conçu. Nous définissons un Curriculum comme un ensemble de tâches indépendantes à accomplir, chaque tâche ayant un environnement, un but et des mécanismes de récompense ou de motivation. Les différentes tâches d'un Curriculum sont ordonnées en fonction de leur complexité croissante, ces dernières pouvant dépendre des

connaissances acquises lors des tâches précédentes. Cet ensemble de tâches couvrira divers aspects d'un comportement désiré, et leur maîtrise conduira à la maîtrise de ce comportement. Dans la perspective d'un développement continu, nous attachons une importance cruciale au fait que le comportement final actuel souhaité sera un élément pour la création future d'un, voir de plusieurs, comportements plus complexes, dont le but ne peut être prévu. En se basant sur des principes de conception modulaire, nous proposons l'architecture Modular Influence Network Design (MIND), une architecture adaptée à l'apprentissage progressif et par Curriculum. Cette architecture reflète les propriétés modulaires du Curriculum dans la représentation des connaissances en créant des modules qui peuvent encapsuler de nombreuses formes de structures d'apprentissage (réseaux neuronaux, approximateur de fonctions, etc.) et permet à chaque module de se coordonner avec les autres pour accomplir une tâche complexe. Cette architecture permettra d'accumuler continuellement de nouvelles compétences et d'utiliser la coordination des compétences antérieures pour maîtriser rapidement de nouvelles tâches d'une complexité croissante.

Après un bref état de l'art à la section 2 nous présenterons l'architecture MIND dans la section 3, puis nous décrirons dans la section 4 le protocole et les résultats des expériences que nous avons menées afin d'évaluer l'architecture. Avant de conclure, nous discuterons des limites et perspectives de MIND dans la section 5.

2 Contexte

2.1 Ontologie

Une *tâche* est un objectif à accomplir pour un agent situé dans un environnement. Si la tâche est destinée à entraîner un agent, elle doit comprendre un système de motivation, un signal de feedback ou une fonction de fitness.

Un *Curriculum*, ou cursus, est une série de tâches d'entraînement, ou leçons, indépendantes et d'une complexité croissante, couvrant les différents aspects d'un comportement souhaité.

Un *comportement* est l'expression d'une compétence, d'une capacité, dont le but est d'accomplir une action en rapport avec la tâche.

Un *skill* ou capacité, est un élément mémorisé, acquis par expérience ou entraînement, qui s'exprime sous forme d'un comportement.

Un *Base skill* ou skill sensori-moteur, est un skill qui associe directement les capteurs avec les actionneurs, et représente le niveau le plus bas de décision, les réflexes. Le fait qu'un base skill contrôle directement les actionneurs le met en concurrence avec les autres base skills souhaitant contrôler les mêmes actionneurs.

Un *Complex skill*, réalise la coordination de skills, ou délégation à d'autres skills, dans le but de représenter un comportement plus complexe que ses sous-skills.

2.2 Apprentissage par Curriculum et Architectures

L'apprentissage par Curriculum [2] est une méthode d'apprentissage progressive étudiée à ce jour dans le domaine du machine learning comme moyen d'accélérer l'apprentissage ou de résoudre des problèmes d'apprentissage qui sont autrement impossibles. L'apprentissage nécessite un Feedback, soit de l'environnement, soit d'une entité de supervision, mais lorsque la tâche et l'environnement d'apprentissage sont trop complexes, le signal de Feedback peut se révéler trop complexe pour permettre l'apprentissage. Par exemple, considérons le cas de l'accumulation de différentes sources de récompense pour différentes actions, où il ne serait pas possible de déterminer quelle action devrait être récompensée.

L'apprentissage par Curriculum subdivise une tâche d'apprentissage en sous-tâches différentes mais complémentaires (Source tasks) à apprendre séparément, et a été appliqué avec succès aux problèmes de robotique et de jeu vidéo [10]. Ici, la compétence est mémorisée par un approximateur de fonction unique grâce à l'apprentissage par transfert des différentes tâches sources [12].

L'apprentissage par Curriculum a également été appliqué dans un contexte plus abstrait pour enseigner à un réseau neuronal à réaliser l'approximation d'une fonction [7]. Dans ce travail, l'idée est d'entrainer le réseau sur une fonction cible simplifiée, puis de faire évoluer cette fonction cible jusqu'à ce qu'elle corresponde à la fonction désirée. Bien qu'il s'agisse d'une méthode d'apprentissage progressif intéressante, l'utilisation du terme Curriculum est discutable. Cette méthode d'apprentissage ajoute simplement plus de complexité à la même tâche d'apprentissage au lieu d'être une collection de tâches d'apprentissage complémentaires. Dans ce travail, la connaissance est également représentée comme un module unique, le réseau neuronal.

Bien que ces méthodes permettent d'apprendre par des moyens progressifs, leur portée est limitée à la tâche spécifique à accomplir. Ils ne couvrent pas l'apprentissage continu, ouvert et cumulatif d'une variété de tâches, un défi au coeur de la robotique dévelopementale qui se caractérise par l'acquisition progressive de

compétences et de connaissances [11].

Afin de pouvoir assimiler le Curriculum sous forme de modules de compétence indépendants l'architecture utilisée doit pouvoir coordonner des modules dont les sorties sont concurrentes.

Des travaux antérieurs dans le domaine des systèmes de commande de robots tels que l'architecture Subsomption [3] offrent des solutions pour alterner l'utilisation de différentes compétences selon le contexte. D'autres, comme AuRA [1] ou Sat-Alt [16], utilisent un mécanisme de composition vectorielle, permettant de choisir entre alterner ou combiner les actions de compétences distinctes. L'échelle de ces architectures reste limitée et elles ne sont pas destinées à des hiérarchies profondes.

2.3 Travaux connexes

Les travaux qui nous inspirent le plus, et que nous espérons développer, sont les techniques de Robot Shaping de Dorigo et Colombetti [4][5]. Ici, les comportements s'apprennent par Curriculum et sont représentés comme des compétences indépendantes. Ces compétences sont ensuite combinées pour obtenir des comportements de plus haut niveau. Plusieurs architectures sont discutées, des hiérarchies monolithiques aux hiérarchies multi-niveaux. Des méthodes d'apprentissage et de récompense adaptées aux agents artificiels, pertinentes dans le cadre de notre travail, sont également proposées. Nous présentons ici quelques points ou les Hiérarchies de Robot Shaping (*RSH*) et MIND diffèrent et expliquons comment et pourquoi MIND représente une amélioration par rapport aux *RSH*.

En *RSH*, les compétences de niveau inférieur sont les seules à avoir accès aux données des capteurs et à envoyer des demandes d'intervention aux compétences de niveau supérieur (coordinateurs). En se basant uniquement sur ces demandes, la compétence de niveau supérieur choisit quelles sous-compétences doivent être coordonnées et comment. Dans MIND, les compétences de niveau supérieur (Complex skills) basent leur décision de coordination sur l'environnement et le contexte en accédant directement aux données des capteurs, y compris celles des capteurs qui ne sont pas utilisés par les sous-compétences. Nous pensons qu'il s'agit là d'un concept important, car lors d'une prise de décision, l'information sur le *pourquoi*, le *quand* et le *si* sont souvent occultés pas le *comment*, ce qui conduit à la même réponse lorsque des détails subtils dans le contexte exigent une approche totalement différente.

Dans MIND, la sortie de toutes les compétences(Skills) est un vecteur de nombres réels entre 0 et 1 au lieu des chaînes binaires de *RSH*. Bien que cela ait un coût computationel élevé, cela permet une certaine finesse dans les commandes de sortie. La méthode d'Influence dépend également de cette sortie en nombre réel pour servir un rôle similaire à celui des opérateurs de combinaison du coordinateur dans la méthode *RSH*. L'utilisation de la méthode d'Influence, à la base de MIND, élimine le besoin de définir un ensemble fixe d'opérateurs de combinaison et permet à la place d'apprendre l'opération de combinaison.

L'utilisation d'un seul type de sortie pour toutes les commandes (influence, information capteur ou commande moteur) permet toute combinaison entre les éléments de l'architecture. *RSH* fait une distinction claire entre les compétences de base (dont les sorties sont des commandes de moteur) et les coordinateurs (dont les sorties sont des commandes de coordination). Dans MIND, nous faisons également la différence entre le Complex skill de plus haut niveau, les Complex skill subordonnés et les Base skill, mais cette distinction n'est que conceptuelle. Il n'y a pas de limite stricte qui empêcherait un Complex skill d'influencer plusieurs Base skills et en même temps d'envoyer des commandes motrices à plusieurs actionneurs.

3 Modular Influence Network Design

MIND (Modular Influence Network Design) est une hiérarchie de modules capables de coordonner des comportements (*Skills*) distincts pour accomplir une tâche complexe. Grâce à cette modularité nous souhaitons obtenir les propriétés suivantes :

1. **Encapsulation** : les comportements génériques seront encapsulés et combinés avec d'autres. On conserve la possibilité d'être la source de plusieurs nouveaux comportements, accélérant ainsi l'apprentissage de tâches différentes ayant une base commune. Cette propriété permet aussi d'identifier et de modifier le comportement localement.
2. **Méthode générique** : MIND a peu de contraintes sur la méthode de décision utilisée par chaque module, réseaux neuronaux et procédures de programmation peuvent être coordonnés par le mécanisme d'influence.
3. **Flexibilité** : les modules de comportement peuvent être remplacés soit par un nouveau module, soit par une hiérarchie de modules favorisant une évolution constante du système. Grâce à son mécanisme de

coordination générique, MIND facilite l'ajout de nouveaux capteurs et actionneurs dans un système déjà existant sans perdre les comportements acquis précédemment.

3.1 Base skill, complex skill, et influence

Considérons un agent dont les informations sensorielles et les commandes motrices sont représentées comme un vecteur de réels, normalisés entre 0 et 1. On peut créer un module qui encapsule une fonction $f(x)$ qui prend en entrée le vecteur $V_I = [I_1, I_2, ..., I_n]$ et fournit en sortie le vecteur $V_O = [O_1, O_2, ..., O_2, ..., O_m]$. Cette fonction peut être implémentée comme une procédure de programmation, ou peut être un approximateur de fonction, un réseau neuronal, ou tout autre type de fonction qui associe deux vecteurs de nombres réels. Nous appellerons un tel module un *skill*, et le module dont le vecteur de sortie est utilisé directement comme commande moteur un *base skill*.

$$V_O = f(V_I) \quad (1)$$

Il est possible de créer un unique base skill qui utilise toutes les entrées sensorielles et toutes les sorties motrices de l'agent pour apprendre à accomplir une tâche, même relativement complexe, chaque leçon du Curriculum étant mémorisée dans la même structure unique. Ce skill est donc la somme de toutes les différentes expériences, sans qu'il soit possible de différencier ce qui a été enseigné.

Au lieu d'accomplir une tâche complexe au moyen d'un seul skill, nous accomplirons de nombreuses sous-tâches, potentiellement concurrentes, avec des base skills distincts. Chaque base skill n'associera que les entrées et sorties nécessaires à l'exécution de la tâche qui lui est associée. Pour accomplir la tâche complexe, nous allons ensuite créer un *complex skill* qui va coordonner plusieurs compétences de base, que nous appelons ses *subskills* (Fig.1). Un *complex skill* coordonne ses *subskills* en envoyant un signal appelé *influence* qui détermine le poids (influence) qu'un subskill aura sur l'action résultante, ce qui revient à déléguer à une ou plusieurs sous-capacités la résolution de la tâche courante.

Un Complex skill, comme n'importe quel skill, encapsule une fonction qui prend en entrée un vecteur de valeurs issues des capteurs V_I et sort un vecteur de nombres réels V_O, dont la sortie est dirigée vers ses subskills. Ce vecteur de sortie est appelé le vecteur d'influence V_{Infl} = $[Infl_1, Infl_2, ..., Infl_m]$, et ses éléments $Infl_x$ sont appelés *influences*.

Un complex skill peut avoir d'autres complex skills comme subskills, créant des hiérarchies

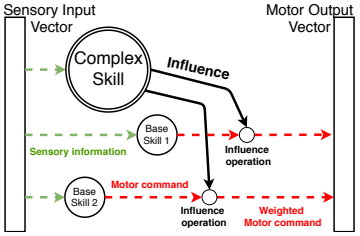


Fig. 1 – Un *complex skill* influençant deux *base skills*

de skills. Au sommet de la hiérarchie se trouve le *Master skill* dont la seule particularité est de recevoir une influence constante de 1.0, une impulsion permettant de lancer le mécanisme de calcul [15]. Cette hiérarchie de skills forme

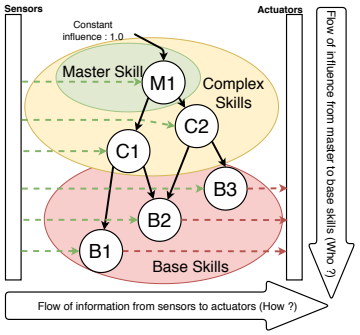


Fig. 2 – Une hiérarchie de skills, le *master skill* influence des *complex skills* qui à leur tour influencent des *base skills*

un graphe orienté acyclique (Fig.2). L'influence s'exerce le long d'un axe vertical, du master skill jusqu'aux base skills, et détermine qui est responsable de l'action résultante. L'information des capteurs est disponible pour tous les skills de la hiérarchie et les commandes des moteurs sont envoyées par les base skills vers les actionneurs formant un flux d'information horizontal, ce flux détermine comment l'action résultante va être exécutée.

3.2 Utilisation de l'influence pour déterminer les commandes moteurs

A partir du master skill, chaque complex skill calcule son vecteur de sortie V_O et multiplie chaque élément par la somme des influences reçues, formant le vecteur d'influence V_{Infl}. Le skill envoie alors chaque élément $Infl$ de V_{Infl} au subskill correspondant (Fig.3).

$$V_{Infl} = V_O * \sum_{x=1}^{n} Infl_x \quad (2)$$

avec V_{Infl} le vecteur d'influence vers les subskills, $V_O = f(V_I)$ le vecteur de sortie de la fonction interne du skill, et $\sum_{x=1}^{n} Infl_x$ la somme de toutes les influences reçues par le skill (également noté $\Sigma Infl$).

Un base skill, comme un complex skill, calcule

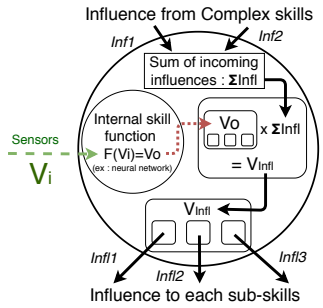


Fig. 3 – Architecture interne d'un skill

son vecteur de sortie et multiplie chaque élément par la somme des influences qu'il a reçues (Eq.2), formant le vecteur de commandes moteur $V_{Com} = [Com_1, Com_2, ..., Com_m]$. Le base skill envoie alors chaque élément Com_x du vecteur de commandes moteur au module moteur correspondant, ainsi que la somme des influences reçues par lui même $\Sigma Infl$.

Chaque module moteur calcule ensuite la commande correspondant à son actionneur sous la forme d'une somme pondérée normalisée :

$$M = \frac{\sum_{x=1}^{n} Com_x}{\sum_{x=1}^{n} \Sigma Infl_x} \quad (3)$$

avec M la commande de moteur résultante, x l'index de la compétence de base qui envoie une commande de moteur, Com_x la commande de moteur pondérée pour ce module moteur depuis le base skill x, $\Sigma Infl_x$ la somme des influences du base skill x.

4 Expérimentation

Pour évaluer l'architecture proposée, nous avons réalisé une série d'expériences dans un simulateur avec un agent réactif, un robot, utilisant un algorithme génétique et des réseaux neuronaux. La première expérience consiste à apprendre la tâche d'aller chercher un objet et de le ramener dans une zone spécifique. Nous avons ensuite modifié la tâche pour inclure une contrainte de consommation d'énergie.

4.1 Le robot

Le robot utilisé dans notre simulation est composé de deux moteurs, un pour chaque roue, et d'une pince pour saisir l'objet cible. Il dispose également de 18 capteurs, détecteurs d'obstacles, orientation de la cible, indicateur de charge de batterie. Le logiciel du robot, au moyen de ce que nous appelons un module capteur, est capable de calculer la dérivée de n'importe quelle entrée de capteur et de l'utiliser comme un capteur virtuel. Il peut également générer une onde sinusoïdale qui peut être utilisée à la manière d'un capteur (pour, par exemple, résoudre des situations de deadlock réactif).

La commande du moteur, transmise par le module moteur à son actionneur sous la forme d'un nombre réel compris entre 0 et 1, est interprétée comme un pourcentage de la capacité de l'actionneur (soit puissance, vitesse, position angulaire, etc.). Dans le cas des roues, 1 correspond à la pleine vitesse avant, 0 à la pleine vitesse arrière et 0,5 à l'arrêt. Dans le cas de la pince, le nombre correspond à son état (traitée comme binaire) : au-dessus de 0,5 fermé, au-dessous de 0,5 ouvert.

4.2 Fonctions internes des skills et algorithme d'apprentissage

Les skills que nous avons utilisées mettent en oeuvre un perceptron multicouche comme fonction interne, dont la couche d'entrée correspond au vecteur d'entrée de la compétence et la couche de sortie à son vecteur de sortie. Selon la compétence, son réseau neuronal utilisera 2 ou 3 couches cachées de N_{HIDDEN} neurones (Eq.4).

$$N_{HIDDEN} = Max(N_{Vi}, N_{Vo}) + 2 \quad (4)$$

N_{Vi} le nombre de neurones de la couche d'entrée et N_{Vo} le nombre de neurones de la couche de sortie

La fonction de transfert des couches du réseau neuronal est sigmoïde, à l'exception de la dernière couche qui utilise une fonction de transfert linéaire qui est fixée entre 0 et 1.

Cette configuration du perceptron est suffisamment générique pour couvrir la plupart des types de compétences, au prix d'un coût supplémentaire pour l'apprentissage génétique.

Contrairement à une approche non hiérarchique qui utilise une compétence unique avec un réseau neuronal unique qui relie tous les capteurs et actionneurs, dans l'approche hiérarchique, chaque compétence a son propre réseau neuronal utilisant uniquement les actionneurs, capteurs ou sous-capacités appropriés.

Pour apprendre les tâches, nous avons utilisé un algorithme génétique simple. Les poids des connexions du réseau neuronal sont ordonnés dans un vecteur, qui correspond au génome de l'individu du point de vue de l'algorithme génétique. Le génome sera croisé et muté en fonction du résultat de la fonction de fitness (d'adaptation) que l'environnement fournit en retour (Fig.4). Nous avons choisi d'utiliser un algorithme génétique pour sa simplicité, ses propriétés exploratoires et sa bonne performance avec récompenses différées. Par nature, il n'est pas nécessaire de lui fournir une description de

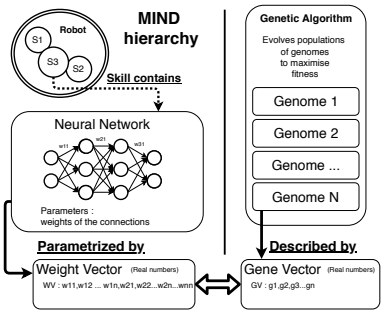


Fig. 4 – Relation entre un skill et l'algorithme génétique

la façon d'atteindre un but, contrairement aux méthodes qui utilisent la rétropropagation qui nécessite un set de couples d'entrées-sorties, mais seulement un moyen de mesurer si la performance d'un individu est meilleure ou pire que celle des autres individus sur une période d'évaluation. Contrairement à d'autres algorithmes d'apprentissage par renforcement où le signal de récompense modifie le comportement, la qualité du comportement est jugée à la fin du cycle de vie de l'individu. Cela permet à l'individu d'obtenir des récompenses négatives s'il en résulte une récompense positive supérieure par la suite, contournant ainsi la question de la récompense différée.

L'un des inconvénients majeurs de l'algorithme génétique est son coût élevé en ressources, en particulier l'évaluation de la fonction de fitness qui consiste à observer le comportement de l'individu, c'est-à-dire un agent, dans un environnement simulé pendant une période suffisamment longue. Toutefois, comme l'évaluation de chaque agent est totalement indépendante, elle peut être exécutée en parallèle. Ceci permet l'utilisation de solutions de calcul haute performance (HPC, mésocentres).

4.3 Scénarios

Les comportements appris présentés dans les scénarios suivants se basent sur la hiérarchie initiale de collecte présentée dans le scénario 1. Afin d'établir la fiabilité de la méthode en dépit de de la nature stochastique des algorithmes génétiques, chaque comportement du scénario 1 à été appris en 10 tentatives indépendantes. Les résultats numériques dépendants des fonctions de récompenses utilisées donnent une indication sur l'évolution et peuvent être utilisés a titre de comparaison, toutefois l'évaluation et l'interprétation d'un comportement complexe repose sur l'observation. Les vidéos des résultats sont disponibles à l'adresse suivante :

`www.lirmm.fr/~suro/Works.html`

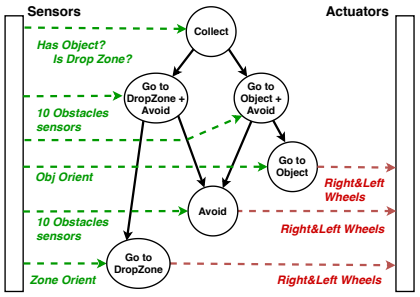


Fig. 5 – La hiérarchie Collect, relation entre les skills, les capteurs et actionneurs

Scenario 1 : Collect. Dans ce scénario, l'objectif est d'apprendre au robot une tâche de ramassage qui consiste à ramasser un objet et à le ramener dans une zone de dépôt sans entrer en collision avec des obstacles.

Nous voulons démontrer ici que l'architecture MIND est capable d'organiser des comportements simples en comportements complexes et qu'elle est capable de le faire de manière fiable même en utilisant un processus d'apprentissage génétique simple.

Nous avons choisi de diviser la tâche complexe en un Curriculum de cinq sous-tâches, dont trois sont des tâches de base : atteindre l'objet dans un environnement vide (GoToObject), atteindre la zone de dépôt dans un environnement vide (GoToDropZone), et éviter les collisions en se déplaçant dans un environnement avec des obstacles (Avoid). Sur ces tâches de base, nous construisons deux tâches complexes de niveau supérieur : atteindre l'objet en évitant les collisions dans un environnement avec des obstacles (GoToObject+Avoid), et atteindre la zone de dépôt en évitant les collisions dans un environnement avec des obstacles (GoToDropZone+Avoid).

Pour créer une hiérarchie initiale, il existe de nombreuses façons d'organiser les différentes sous-compétences, qui sont toutes valides, du moment qu'elles soient capables d'assimiler le Curriculum.

Apprendre la tâche Collect est un défi intéressant. Nous pouvons voir dans ses sous-tâches que le fait de se rendre à l'objet et de se rendre à la zone de dépôt utilisent les fonctions motrices d'une manière complètement opposée et devra être utilisée de façon séquentielle et exclusive. Au contraire, la tâche Avoid serait mieux utilisée comme une composition de vecteurs, en modifiant la trajectoire définie pour éviter un obstacle en douceur.

Une tentative sur dix des skills GoToObject et GoToDropZone n'atteignent pas le résultat optimal dans le nombre de générations données (Fig.6). Il est intéressant de noter que les deux

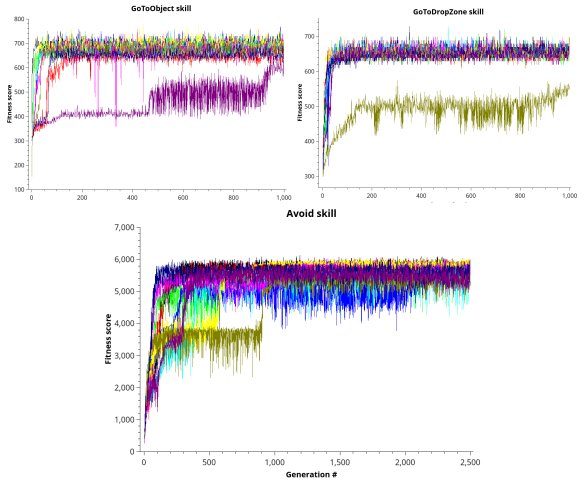


Fig. 6 – Base skills, 10 tentatives distinctes. GoToObject et GoToDropZone : 1000 générations. Avoid : 2500 générations.

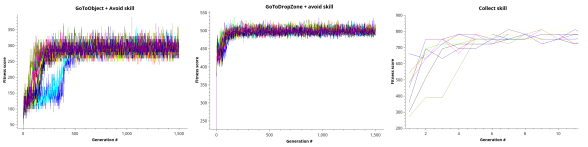


Fig. 7 – Complex skills GoToObject + Avoid, GoToDropZone + Avoid et Master skill Collect : 10 tentatives distinctes, 1500 générations.

tentatives échouées continuent de s'améliorer à chaque génération, ce qui est positif dans le contexte de notre travail visant le développement continu d'agents artificiels. La complexité des tâches peut être difficile à évaluer, mais il est facile de comprendre que le réseau neuronal affecté au skill GoToDropZone, qui utilise deux entrées, aura beaucoup moins de connexions, et donc moins de poids à ajuster, que le réseau neuronal affecté au skill Avoid, qui utilise plus de 10 entrées. Les graphiques de la figure 6 montrent comment cette différence de complexité affecte l'apprentissage. La plupart des tentatives des skills GoToDropZone et GoToObject atteignent leur valeur maximale en 100 générations, alors que la plupart des tentatives du skill Avoid continuent à évoluer après 500 générations. En se basant sur les meilleures base skills, nous apprenons les complex skills GoToObject+Avoid et GoToDropZone+Avoid. Le skill GoToObject+Avoid semble un peu plus difficile à apprendre que GoToDropZone+Avoid, ce qui s'explique par le fait que l'objet est plus petit que la dropZone et nécessite plus de précision (la "pince" doit être alignée avec l'objet pour le saisir). L'apprentissage des deux skills réussit pour chacune de leurs 10 tentatives respectives, dans le nombre de générations données.

Enfin, le master skill Collect, ayant un réseau très simple à entraîner, réussit son apprentissage pour chacune de ses 10 tentatives en moins de 10 générations (Fig.7).

Scenario 2 : apprendre avec des subskills sub-optimaux, rééducation et apprentissage dans un contexte plus général. Après que le scénario 1 ait réussi à établir une hiérarchie capable d'accomplir la tâche Collect, nous avons observé que le skill Avoid (évitement) était la cause principale d'échec pour l'agent et une perte de performance pour la hiérarchie. En nous inspirant de la citation suivante, nous avons expérimenté avec une autre stratégie d'apprentissage.

> *Les architectures hiérarchiques sont particulièrement sensibles à la stratégie de construction (Shaping Policy) ; en effet, il semble raisonnable que les modules de coordination soient façonnés après que les modules inférieurs aient appris à produire les comportements simples. Les expériences (...) montrent qu'en fait, on obtient de bons résultats en formant les modules les plus bas, puis en les "gelant" pour ensuite en former les coordinateurs, et enfin en laissant tous les éléments libres de continuer à apprendre ensemble.*
>
> *Dorigo et Colombetti*[4]

Dans un premier temps, le processus d'apprentissage génétique utilisait 1000 générations pour chaque skill. Le skill Collect résultant était bien capable d'accomplir la tâche de collection, mais échouait encore régulièrement à cause de collisions. Il n'est pas surprenant, du fait de son plus grand réseau neuronal qui coordonne plus de 10 capteurs, que le skill Avoid ait besoin de plus de ressources que les autres skills aux réseaux plus petits pour être entraîné correctement.

Nous avons recommencé l'entrainement Avoid avec 2500 générations ainsi que toutes les compétences de niveau supérieur. Avec 2500 générations, le score de fitness final du skill Avoid a augmenté de 10% par rapport au skill Avoid entraînée avec 1000 générations. Le skill Collect basé sur le nouveau skill Avoid a vu son score final augmenter de 14%, ce qui indique fortement que le skill Avoid est le facteur limitant dans la performance globale de la hiérarchie.

Même en allouant plus de ressources au skill Avoid (plus du double), nous n'avons pas été en mesure d'améliorer significativement la performance globale. Cela nous amène à nous interroger sur la qualité de la tâche d'évitement dans le Curriculum.

Les choix faits dans les éléments de la fonction de récompense sont certainement la cause de la mauvaise qualité de l'apprentissage. La nature du processus d'évolution est d'exploiter toutes

les règles mise en place pour obtenir le meilleur score, sans soucis de l'esprit de ces règles et des raisons qui nous on conduit a les établir. Par exemple, au cours d'expériences préliminaires, la seule récompense pour le skill Avoid que nous ayons donné était une récompense négative que l'individu recevait lorsqu'il heurtait un obstacle, de sorte que l'individu ayant obtenu le meilleur score ne bougeait tout simplement pas. Chacune des règles ajoutées par la suite a été à son tour exploitée jusqu'à ce que nous trouvions un ensemble de règles qui nous a permis d'obtenir un comportement proche de ce que nous souhaitions. Ceci illustre le problème de la définition de la fonction de fitness, qui peut être aussi difficile que de simplement programmer le comportement que l'on souhaite.

Nous avions besoin d'un skill Avoid pour construire la hiérarchie, mais maintenant que chaque skill a son propre rôle défini dans la hiérarchie, au lieu d'essayer d'améliorer la qualité d'Avoid par lui-même, nous allons l'entrainer dans le contexte final de la tâche de collecte, en tant qu'élément de la hiérarchie.

Pour ce faire, nous avons utilisé la hiérarchie déjà entraînée du scénario 1 en utilisant le skill Collect comme master skill. Nous avons utilisé l'environnement final de collecte, et le signal de récompense (la fonction de fitness) provenait de l'accomplissement de la tâche de collecte. Dans ce contexte, nous avons recommencé l'entraînement du skill Avoid (en ignorant la fonction de fitness définie à la main pour la tâche Avoid en faveur de la fonction de fitness de la tâche Collect). En n'utilisant que 1500 générations pour entraîner Avoid avec cette méthode, le score de fitness final du skill Collect s'est amélioré de 30%.

Ce résultat suggère qu'il pourrait y avoir d'autres stratégies d'apprentissage à considérer lorsqu'on entraîne une hiérarchie au-delà d'un simple déroulement du Curriculum, des tâches de base aux tâches complexes. Revenir sur les tâches précédentes pour améliorer les subskills dans le contexte final, une fois que tout le Curriculum a été sommairement enseigné, peut donner de meilleurs résultats que d'essayer de maximiser chaque subskill avant de passer au suivant.

L'entraînement progressif initial de la hiérarchie est toujours d'une grande importance, même si chaque compétence n'a pas besoin d'être entraînée à la perfection, cette première étape du Curriculum est celle où chaque skill se spécialisera dans son rôle au sein de la hiérarchie. Au cours d'expériences précédentes non détaillées dans le présent article, nous avons tenté d'entraîner la hiérarchie de collecte

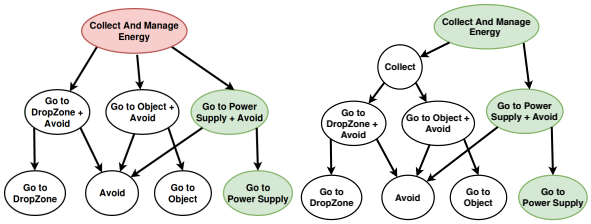


Fig. 8 – Collecte + gestion d'énergie, à gauche ré-entrainée, à droite encapsulée

en apprenant toutes les compétences en même temps. Dans ces conditions la plupart des branches de la hiérarchie restait inactives et un seul skill a tenté d'apprendre la tâche complète.

Scenario 3 : Collect et gestion d'énergie, la modularité de MIND. Nous souhaitons montrer que l'architecture MIND est adaptée au développement ouvert par l'ajout de nouvelles compétences qui étendent les capacités d'une hiérarchie déjà formée. Nous ajoutons à la compétence de collecte initiale la consommation d'énergie et la nécessité de recharger la batterie du robot. Un capteur qui indique le niveau de la batterie et des capteurs qui donnent des informations sur la direction de la source d'alimentation sont ajoutés au robot. Ce scénario ajoute au scénario précédent la tâche de base d'aller à la source d'alimentation pour se recharger et la tâche complexe d'aller à la source d'alimentation sans entrer en collision avec les obstacles, en réutilisant le skill Avoid.

Pour intégrer un nouveau skill dans une hiérarchie pré-existante, il y a deux façons possibles de procéder :

1. modifier le complex skill existant pour qu'il intègre le nouveau skill, et la ré-entraîner pour l'adapter (Variante ré-entraînée).
2. créer un nouveau complex skill qui coordonne le nouveau skill avec le complex skill existant. Le nouveau complex skill sera entraîné et le complex skill pré-existant ne sera pas modifié (Variante encapsulée).

L'évaluation pour ce scénario est la même que celle du scénario 1 avec l'ajout de la gestion de l'énergie. La batterie du robot se décharge de manière constante à moins que le robot ne se trouve dans le zone de recharge, auquel cas la batterie se recharge rapidement. L'évaluation est exécutée pendant une période donnée ou jusqu'à ce que le robot échoue (collision ou batterie vide). Chaque fois que le robot ramène l'objet dans la zone de dépôt, il marque un point et un nouvel objet est placé au hasard dans l'environnement.

Aucune indication de ce qui constitue un niveau de batterie faible n'a été donnée au robot, seulement le niveau de batterie actuel. Le robot

a déterminé par lui-même le niveau de batterie pour lequel il devait abandonner la tâche en cours et se diriger vers la source d'énergie. Cette décision se base sur sa propre expérience avec de la vitesse de décharge de sa batterie et la taille et complexité de l'environnement.

Les deux variantes de la hiérarchie (Fig.8), ré-entrainée et encapsulée, obtiennent des scores comparables montrant que les deux sont valables. Il faut noter que la variante encapsulée préserve le skill original Collect, la rendant disponible pour de futures combinaisons, ce qui est une des propriétés que nous souhaitions obtenir en proposant MIND (sec. 3).

5 Perspectives

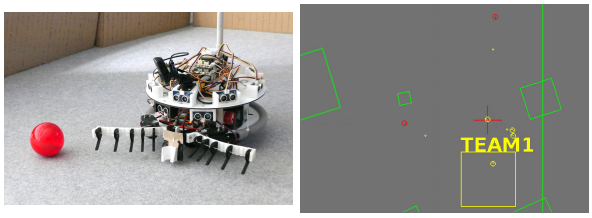


Fig. 9 – à gauche, Notre robot hi-tech, à droite, Capture de drapeau avec élimination

5.1 MIND sur un robot physique

Étant donné que la hiérarchie MIND apprise en simulation est entièrement réactive, nous n'avons pas anticipé de problème pour l'utiliser dans une application réelle. Nous avons conçu une tâche simple en utilisant les deux base skills GoToObject et Avoid, et le complex skill (dans ce cas aussi le master skill) GoToObject+Avoid.

La seule difficulté que nous avons rencontrée est le délai d'acquisition des capteurs et de réponse du moteur, très probablement à cause du matériel amateur utilisé. Néanmoins, en définissant des zones mortes pour les capteurs appropriées et une portée maximale, notre robot produit le comportement attendu, et ce en utilisant la hiérarchie et les compétences acquises dans une simulation qui n'a été calibrée d'aucune façon pour s'adapter au robot ou à l'environnement du monde réel.

Notre robot utilise un équipement sensori-moteur issu d'un kit de robotique amateur, afin de représenter le capteur d'orientation de l'objet cible nous avons utilisé un bras pan-tilt et une webcam pour tracker une balle lumineuse ("l'objet"). La position du bras panoramique est convertie en une information de capteur reproduisant le capteur d'orientation.

5.2 Applications multi-agents

Nous étudions actuellement l'apprentissage social en utilisant des hiérarchies MIND sur une tâche de capture de drapeau compétitive. Deux équipes de 10 agents doivent capturer le drapeau adverse et le ramener à leur base tout en protégeant le leur. Les agents ont à leur disposition, en plus de capteurs et d'actionneurs appropriés, un système d'émetteur et de récepteur de signaux. La technique d'apprentissage reste génétique, les agents de la même équipe utilisent une instance de la même hiérarchie de skills, mais le score de fitness est évalué collectivement. Les skills de bas niveau restent essentiellement sensori-moteurs, mais plus on monte dans la hiérarchie plus l'aspect social est pris en compte. Pour accomplir la tâche finale, gagner une partie de capture de drapeau, une organisation stratégique, et donc sociale, est nécessaire. Les agents sont capables de réagir à leur contexte respectif, leur position relative aux drapeaux, aux ennemis, on obtient une répartition des tâches, mais pour obtenir une répartition en rôles spécifiques, il est nécessaire de leur fournir une identité, dans notre cas une variable mémoire (constante dans ce cas). Pour le moment nous avons une preuve de concept programmée, mais nous espérons à terme être capable d'apprendre une répartition des rôles à la manière du niveau de batterie critique dans le scénario 3, par émergence.

5.3 Limites

Nos expériences ont montré la capacité de MIND à gérer de petites hiérarchies. Toutefois, on peut se demander si MIND reste applicable pour de grandes hiérarchies, en particulier pour des hiérarchies profondes où le transfert successif d'influence pourrait donner lieu à une forme de bruit sur les commandes motrices due à l'imprécision des réseaux de neurones. Si cette préoccupation est justifiée et qu'un entraînement plus fin des fonctions internes est coûteux et peu pratique, nous envisagerons des moyens de réduire le bruit dans les grands réseaux tels que des filtres ou des couches seuils.

6 Conclusion

Les résultats positifs obtenus sur la tâche Collecte sont encourageants, l'architecture MIND a su transformer le Curriculum en une hiérarchie de skills avec à la fois coordination et exclusion mutuelle de skills. MIND a été en mesure de traiter les tâches sensori-motrices de bas niveau ainsi que les opérations complexes d'influence des skills, sans fournir aucune information a priori sur la nature de la tâche.

Même si nous n'avons discuté que de skills utilisant des perceptrons multicouches comme fonctions interne en raison de notre focalisation sur le problème d'apprentissage, l'encapsulation de la fonction interne dans un skill peut donner à

toute fonction la capacité de s'intégrer dans une hiérarchie MIND. Du moment qu'il est possible d'obtenir des vecteurs d'entrées et de sorties sous forme de nombres réels, n'importe quel type de fonction peut être utilisé. Par exemple, la fonction contrôlant la "pince" du robot dans notre expérience est une procédure Java triviale qui a été intégrée à un skill.

D'autres auteurs[10][7] ont déjà établi les avantages de l'apprentissage progressif, en soumettant l'entité apprenante à une augmentation progressive de la complexité de la tâche ou en apprenant un Curriculum de tâches complémentaires visant à accomplir une tâche complexe. Les résultats positifs que nous avons présentés dans cet article en ré-entraînant les subskills de la hiérarchie semblent indiquer qu'il est intéressant de repasser plusieurs fois sur les leçons du Curriculum plutôt que d'essayer d'obtenir des résultats parfaits sur les tâches de base avant de passer aux plus complexes. Nous utilisons cette technique, en conjonction avec la capacité de certains algorithmes d'apprentissage de reprendre l'entraînement à partir d'un état sauvegardé, pour optimiser les hiérarchies MIND.

Au moment de la rédaction de cet article, plusieurs améliorations sont prévues pour la mise en oeuvre de l'architecture MIND, notamment l'utilisation de l'algorithme NEAT[17] pour remplacer notre algorithme génétique naïf. Non seulement cela améliorera grandement les performances, mais aussi éliminera complètement la nécessité de définir la topologie du réseau neuronal de la fonction interne des skills.

L'utilisation d'une méthode de communication standard entre tous les modules ouvre un certain nombre de possibilités, telles que l'utilisation de l'état précédent d'un actionneur de la même manière que nous utilisons les données du capteur. Il permet également de créer un certain nombre de modules de mémoire qui peuvent stocker la sortie d'un skill de la même manière qu'un actionneur reçoit une commande motrice, et peuvent fournir leur valeur à d'autres skills comme le ferait un module capteur. Grâce à ces modules mémoires se conformant au système d'influence MIND nous espérons permettre à l'agent d'évoluer au-delà du simple comportement réactif. Nous étudions actuellement un moyen d'enseigner une représentation de la connaissance avec la contrainte de ne pas franchir la barrière de l'intériorité de l'agent.

Notre prochaine étape vers la création d'un agent apprenant autonome est la création automatique de nouveaux skills. Pour une nouvelle leçon donnée par l'instructeur, quelles sont les entrées de capteurs pertinentes ? Quelles sorties d'actionneur ou subskills l'agent doit-il utiliser ? Quel langage et quelles stratégies dans la formulation d'une leçon conviendrait à l'agent tout en étant naturel et simple à définir pour l'instructeur humain ?

Références

[1] R. C. Arkin and T. Balch. Aura : Principles and practice in review. *Journal of Experimental & Theoretical Artificial Intelligence*, 9(2-3) :175–189, 1997.

[2] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009.

[3] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1) :14–23, 1986.

[4] M. Dorigo and M. Colombetti. Robot shaping : Developing autonomous agents through learning. *Artificial intelligence*, 71(2) :321–370, 1994.

[5] M. Dorigo and M. Colombetti. *Robot shaping : an experiment in behavior engineering*. MIT press, 1998.

[6] D. J. Felleman and D. C. Van Essen. Distributed hierarchical processing in the primate cerebral cortex. *Cerebral cortex*, 1(1) :1–47, 1991.

[7] Ç. Gülçehre, M. Moczulski, F. Visin, and Y. Bengio. Mollifying networks. *CoRR*, abs/1608.04980, 2016.

[8] N. Kruger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. J. Rodriguez-Sanchez, and L. Wiskott. Deep hierarchies in the primate visual cortex : What can we learn for computer vision ? *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1847–1871, 2013.

[9] M. Lungarella, G. Metta, R. Pfeifer, and G. Sandini. Developmental robotics : a survey. *Connection science*, 15(4) :151–190, 2003.

[10] S. Narvekar, J. Sinapov, M. Leonetti, and P. Stone. Source task creation for curriculum learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*, pages 566–574. International Foundation for Autonomous Agents and Multiagent Systems, 2016.

[11] P.-Y. Oudeyer. Developmental robotics. In *Encyclopedia of the Sciences of Learning*, pages 969–972. Springer, 2012.

[12] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10) :1345–1359, 2010.

[13] J. Piaget. *The construction of reality in the child.* Basic Books, New York, 1954.

[14] J. Piaget and E. Duckworth. Genetic epistemology. *American Behavioral Scientist*, 13(3) :459–480, 1970.

[15] A. Schopenhauer. *Die Welt als Wille und Vorstellung*. Brockhaus, Leipzig, 1819.

[16] O. Simonin and J. Ferber. Modeling self satisfaction and altruism to handle action selection and reactive cooperation. In *Proceedings of the 6th International Conference on the Simulation of Adaptive Behavior*, volume 2, pages 314–323, 2000.

[17] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2) :99–127, 2002.

Exploration et couverture par stigmergie d'un environnement inconnu avec une flotte de robots autonomes réactifs

Nicolas Gauville[a, b] nicolas.gauville@loria.fr
François Charpillet[a] francois.charpillet@inria.fr

[a] Universite de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy

[b] Safran Electronics & Defense

Résumé

L'exploration autonome d'un environnement inconnu peut être envisagée de différentes manières. On peut notamment citer les approches par frontières, où des robots sont affectés à des zones inexplorées de la carte. Ces dernières méthodes sont efficaces mais nécessitent de partager une carte, globaliser les décisions d'affectation. Les approches Brick and Mortar, quant à elles, utilisent un marquage au sol avec une prise de décision locale, mais donnent des performances beaucoup moins intéressantes. L'algorithme présenté ici est un compromis entre ces deux approches, permettant une prise de décision locale et, de façon surprenante, des performances proche des approches par frontières globales. Nous proposons également une étude comparative de la performance des trois différentes approches : Brick & Mortar, frontières globales et frontières locales. Notre algorithme local est également complet pour le problème d'exploration et peut être facilement distribué sur des robots avec une perte de performance mineure.

Mots-clés : *Multi-robot, stigmergie, exploration*

Abstract

Different approaches exist for multi-robot autonomous exploration. These include frontier approaches, where robots are assigned to unexplored areas of the map, which provide good performance but require sharing the map and centralizing decision-making. The Brick and Mortar approaches, on the other hand, use a ground marking with local decision-making, but give much lower performance. The algorithm presented here is a trade-off between these two approaches, allowing local decision-making and, surprisingly, performances are closed to centralized frontier approaches. We also propose a comparative study of the performance of the three different approaches : Brick & Mortar, Global Frontiers and Local Frontiers. Our local algorithm is also complete for the exploration problem and can be easily distributed on robots with a minor loss of performance.

Keywords: *Multi-robot, stigmergy, exploration*

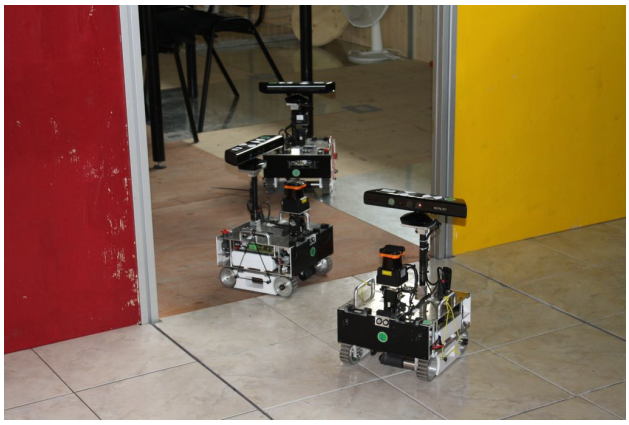


FIGURE 1 – Robot Minirex (utilisé pour le projet Cart-O-Matic) explorant une zone test.

1 Introduction

L'exploration autonome d'un environnement inconnu est un défi important en robotique mobile. L'exploration peut être une fin en soi : des robots de nettoyage (aspirateur, tondeuse à gazon, etc.) explorent systématiquement chaque zone à couvrir. Dans les applications plus sophistiquées, il s'agit principalement d'acquérir des informations : cartographier l'environnement [2], rechercher un intrus, localiser une source sonore ou olfactive [14, 17], ou encore rechercher d'éventuelles personnes à secourir dans un bâtiment en feu, . . .

L'objectif de cet article est de proposer une approche par frontières *locales* pour aborder le problème de l'exploration multi-robot. L'objec-

tif principal de ce travail est de proposer une approche évolutive et efficace, même dans un environnement où la communication est restreinte, pour résoudre le problème de l'exploration avec un grand nombre de robots. Nous avons également essayé de fournir un algorithme aussi simple que possible, exécutable avec des ressources limitées.

Notre approche, bien qu'inspirée des approches par frontières globales, aborde différemment le problème de l'exploration : dans le cas de l'approche globale, c'est avant tout un problème de planification, où l'algorithme doit décider de l'affectation des robots aux frontières connues. Dans notre version locale, les robots ne connaissent pas le nombre ou la position des autres robots et ne s'en soucient pas. Ils réagissent uniquement aux informations locales. La collaboration entre robots se fait par stigmergie, dans notre cas simulée à l'aide d'une carte qui est partagée entre robots à intervalles réguliers, mais d'autres méthodes pourraient être envisagées pour limiter au mieux la quantité et la fréquence des informations partagées [11].

2 État de l'art

L'exploration multi-robot vise à explorer un environnement inconnu le plus rapidement possible pour une équipe homogène de robots mobiles. C'est une question proche du problème de couverture [10], mais avec une carte inconnue.

Différentes approches ont été développées en fonction des contraintes. On y retrouve notamment les approches de type *Brick & Mortar* [8, 9, 1] qui peuvent être classées dans les « algorithmes fourmis », où les robots partagent une carte commune pour *marquer* l'environnement avec des traces similaires aux phéromones des insectes sociaux (à la différence que ces traces ne changent pas avec le temps). Les décisions sont alors prises localement, mais le partage d'une carte globale entre robots est nécessaire pour partager les traces virtuelles laissées sur la carte.

Des approches « purement fourmis » existent aussi, utilisant des phéromones dispersées par l'environnement pour répartir les robots [6]. Ces approches sont souvent plus difficiles à mettre en place que les approches *Brick & Mortar*, car il faut également faire évoluer les phéromones déposées (évaporation, dispersion, . . .). Garnier et al. ont par exemple proposé un système de projection de points avec des robots équipés de capteurs de lumière pour simuler ces phéromones partagées [11, 15], mais cela nécessite d'avoir un appareil pouvant projeter des points sur l'ensemble de la surface à explorer, ce qui n'est pas envisageable dans le cas de l'exploration d'un bâtiment inconnu. Nous préférerons donc ici les approches *Brick & Mortar* où il suffit de partager les traces laissées.

L'exploration peut également être considérée comme un problème d'ordonnancement ; l'approche par frontières [18, 7, 5] consiste à attribuer à chaque robot une frontière, c'est-à-dire une limite entre les parties explorées et inexplorées de la carte. Dans ce cas, un consensus global doit également être assuré.

Si les approches locales sont efficaces, elles restent beaucoup plus lentes que les approches par frontières en termes de temps d'exploration : les robots doivent effectuer de nombreux marquages pour ne pas se bloquer mutuellement ou « oublier » une zone à explorer. D'autre part, les approches par frontières sont plus efficaces et basées sur des algorithmes plus simples, mais nécessitent une coordination des décisions entre robots qui peut poser différents problèmes dans des situations réelles où une communication constante ne peut être assurée [12]. De plus, l'algorithme d'assignation des frontières utilisé nécessite souvent un temps de calcul non linéaire en fonction du nombre de robots et de la taille de la grille, ce qui peut la rendre inapplicable dans une situation réelle si la carte est trop grande ou s'il y a trop de robots. Cependant, il existe différentes méthodes pour optimiser l'affectation et les données échangées, par exemple en partageant uniquement les frontières et non la carte entière [13]. Enfin, notre approche locale peut facilement être distribuée, et fonctionne également avec des communications limitées (voir section 5).

L'algorithme proposé ici est un compromis entre l'approche *Brick & Mortar* et l'assignation de frontières : c'est une approche de frontières locales, où les robots partagent une carte, mais avec une prise de décision locale pour chaque robot. Notre approche est donc moins sensible à l'asynchronisme et réduit significativement les calculs effectués par chaque robot, car le nombre de cellules prises en compte à un moment donné est beaucoup moins important.

Ce travail fait suite au projet *Cart-O-Matic* [3]. Cart-O-Matic était l'un des cinq projets fondés par l'Agence Nationale de la Recherche (ANR) pour sa participation au concours de robotique organisé par la *Délégation générale pour l'ar-*

mement (DGA). Ce concours intitulé « Defi *CAROTTE* » avait pour objectif de définir un système robotique capable d'explorer un environnement intérieur inconnu et identifier des objets localisés dans cet environnement. Notre projet Cart-O-Matic a choisi de déployer un système multi-robot. Cart-O-Matic a remporté le concours final en 2012.

Toutes les approches considérées ici utilisent des grilles d'occupation pour représenter le monde, et considèrent que chaque robot est équipé d'un Lidar principalement utilisé pour la localisation et la cartographie simultanées (SLAM). Cependant, notre approche, comme celle par frontières globales, peut être adaptée à d'autres représentations du monde [16].

2.1 Frontières globales

Dans le cas de l'approche par frontières globales, chaque robot possède une grille d'occupation pouvant prendre trois états : *inexploré*, *exploré* et *obstacle*, initialisée à *inexploré*. À chaque itération, chaque robot marque l'ensemble des cellules dans son rayon de perception comme *exploré* ou *obstacle*. Les frontières entre les zones explorées et inexplorées de la carte peuvent alors être calculées en détectant toutes les cellules contiguës qui ont à la fois des voisines immédiates explorées et inexplorées.

Une fois toutes les frontières de la carte détectées, l'algorithme d'assignation (*MinPos* [4]) est utilisé toutes les 4 itérations pour assigner une frontière à chaque robot, et ceux-ci peuvent alors se diriger vers la frontière qui leur a été assigné. La stratégie d'assignation *MinPos* est décrite en 4.1.

2.2 Algorithme *BMILRV*

L'algorithme *BMILRV*[1] utilise de nombreux marquages sur la grille d'occupation : un état parmi un ensemble d'états { *inexploré, exploré, fermé, obstacle, rendez-vous* }, une valeur indiquant si un robot contrôle une cellule donnée, la direction prise lors du dernier passage sur une cellule, et une variable booléenne indiquant si un robot donné est déjà passé sur une cellule. Les robots peuvent également être dans différents modes parmis { *détection de boucle, contrôle de boucle, fermeture de boucle, nettoyage, pause, arrêt* }.

De manière informelle, l'algorithme *BMILRV* consiste à maintenir une liste de cellules ouvertes permettant d'accéder à toutes les zones inexplorées de la carte, tout en fermant progressivement toutes les cellules qui peuvent l'être (qui ne bloqueront pas le passage vers une zone inexplorée), évitant ainsi de repasser dans les zones déjà visitées.

Lorsque des boucles sont présentes dans le chemin constitué par les cellules explorées, les robots peuvent les fermer en plusieurs étapes permettant d'éviter de bloquer un robot. Le robot détectant la boucle (entrant dans une cellule explorée sur laquelle il est déjà passé en venant de la même direction), va alors passer en mode « *contrôle de boucle* » et parcourir cette boucle pour la marquer comme contrôlée avec son identifiant, vérifiant ainsi qu'il est le seul à chercher à la fermer à ce moment. S'il parvient à marquer la totalité de la boucle, il effectue alors un second passage en mode « *fermeture* » ou il va fermer les cellules de la boucle jusqu'à la première intersection, et enfin parcourir la boucle une troisième fois en sens inverse en mode « *nettoyage* » pour enlever les traces indiquant qu'il contrôle la boucle. S'il ne parvient pas à contrôler la boucle, alors il passe directement en mode « *nettoyage* » si le robot qui la contrôle a une priorité supérieure, ou en mode « *pause* » si il est prioritaire, en attendant que l'autre robot ait nettoyé ses traces sur la boucle.

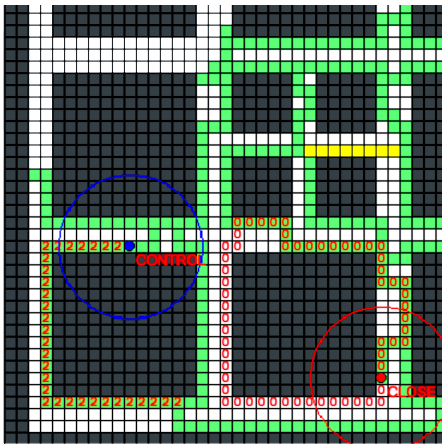


FIGURE 2 – Représentation de la grille lors d'une exploration par l'algorithme *BMILRV*. Les cellules fermées sont en gris, ouvertes en vert, point de rendez vous en jaune et inexplorées en blanc. Les chiffres dans les cellules correspondent aux numéros des robots contrôlant ces cellules, l'état des robots est indiqué en rouge.

Les différentes étapes de l'algorithme BMILRV ne sont pas triviales ; elles nécessitent de nom-

breuses vérifications pour fermer correctement les cellules, et un décalage ou asynchronisme dans la procédure de marquage des cellules pourra bloquer les robots. Pour des raisons de place, nous n'entrerons pas dans les détails du fonctionnement de l'algorithme ici.

3 Suivi de frontières locales

3.1 Exploration exhaustive

De façon informelle, l'algorithme local consiste à marquer tout chemin parcouru comme visité, à se déplacer le plus loin possible vers des zones inconnues, et revenir sur ses pas lorsqu'il n'y a plus de zone inconnue visible, jusqu'à ce que l'on en retrouve une.

Lorsque le robot est revenu à son point de départ sans avoir vu aucune zone inexplorée, toute la carte a été explorée.

Dans le cas de plusieurs robots, selon les conditions initiales et la topologie de la carte, un ou plusieurs robots peuvent être revenus à leur point de départ avant la fin de l'exploration, ce qui entraîne une mauvaise distribution de l'exploration et réduit la performance de l'algorithme. Pour réduire cette perte de performance, on peut envisager d'utiliser dans ce cas les traces des autres robots pour reprendre l'exploration une fois qu'ils sont revenus au point de départ, si tous les autres robots ne se sont pas arrêtés. Cette piste d'amélioration est implémentée dans la partie 6.

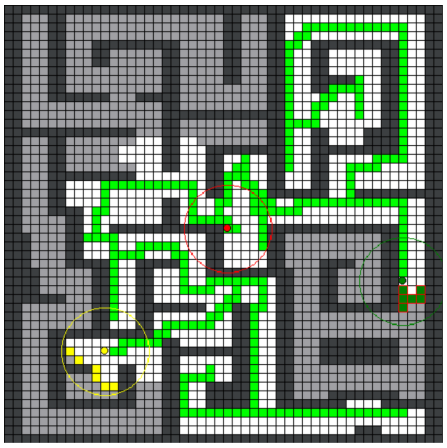


FIGURE 3 – Représentation de la grille lors d'une exploration par l'algorithme par frontières locales (cellules explorées en blanc, traces en vert ; les cellules vert foncé et jaune correspondent aux frontières locales des robots vert foncé et jaune).

3.2 Formalisation (frontières locales)

Nous considérons une flotte homogène de m robots mobiles $R = \{r_1, \dots r_m\}$ équipés de capteurs leur permettant de détecter des obstacles dans un rayon de perception ρ. Les robots utilisent une grille d'occupation $\mathcal{O}$ représentant la carte, où chaque cellule peut être dans trois états : *inexploré*, *obstacle* ou *exploré*, ainsi qu'une grille $\mathcal{P}$ pour assigner un booléen à chaque cellule, indiquant si une cellule est visitée ou non (c'est à dire si un robot a été sur cette cellule), initialisée à *faux*. Ces cartes sont échangées à intervalles réguliers entre les robots (voir section 5). Enfin, chaque robot peut définir une cellule objectif (représentant un point à atteindre sur la carte), et possède une trace $\mathcal{G}r_i$ correspondant à la distance maximale parcourue depuis le point de départ.

Pour chaque déplacement (d'une cellule), chaque robot connaît l'état des cellules dans son rayon de perception et sa position sur la grille (x, y). On notera $\mathcal{V}_{r_i}^t$ l'ensemble des cellules visibles perçues par le robot à un moment donné t qui ne sont pas des obstacles.

Les cartes locales ($\mathcal{G}$ et $\mathcal{P}$) sont partagées entre les robots toutes les d_{sync}itérations avec la procédure décrite dans la partie 5.

Le paramètre d_{max} est utilisé pour forcer les robots à réassigner la cellule objectif à intervalles réguliers, évitant ainsi de choisir une cible maintenue trop longtemps si le rayon de perception est grand.

Chaque robot r_i applique l'algorithme suivant :

1. Marquer toutes les cellules visibles comme explorées :
   $\forall (x, y) \in \mathcal{V}_{r_i}^t, \mathcal{O}_{r_i}(x, y) = \textit{exploré}$
2. Si aucune cellule objectif n'est définie, déterminez les frontières locales $\mathcal{F}$, c'est-à-dire toutes les cellule de $\mathcal{V}_{r_i}^t$ ayant des voisins inexplorés.
   (a) Si $|\mathcal{F}| > 0$, choisir la cellule frontière la plus éloignée de toute cellule visitée (dans la grille $\mathcal{P}_{r_i}$) à une distance maximale d_{max} de la cellule comme nouvel objectif.
   (b) Si $|\mathcal{F}| = 0$ et si on est de retour à la cellule de départ, s'arrêter.
   (c) Sinon, retournez un pas en arrière sur la trace G_{r_i} : choisir la cellule, parmi les 4 cellules adjacentes, ayant la valeur $\mathcal{G}_{r_i}(x, y)$ la plus basse.

3. Sinon, si une cellule objectif est définie
   (a) Si $\mathcal{G}_{r_i}(x,y) = 0$, mettre à jour la trace : $\mathcal{G}_{r_i}(x,y) = 1 + \max(\mathcal{G}_{r_i}(x-1,y), \mathcal{G}_{r_i}(x+1,y), \mathcal{G}_{r_i}(x,y-1), \mathcal{G}_{r_i}(x,y-1), \mathcal{G}_{r_i}(x,y+1))$.
   (b) Mettre à jour la grille : $\mathcal{P}_{r_i}(x,y) = 1$
   (c) Se rapprocher de l'objectif (à l'aide d'un algorithme de *pathfinding* parmi les cellules visibles), puis revenir à l'étape 1.
4. Si plus de d_{sync}itérations ont été effectuées depuis le dernier partage de carte, partagez la carte $(\mathcal{O}_{r_i}, \mathcal{P}_{r_i})$ avec les autres robots (voir section 5).

Dans nos expériences, nous avons utilisé l'algorithme *A** pour rechercher le chemin optimal entre les robots et leurs cellules objectifs.

4 Comparaison des différentes approches

4.1 Algorithmes choisis

Puisque l'algorithme présenté ici est un compromis entre les approches *Brick & Mortar* et par *frontière*, nous avons choisi de le comparer à un algorithme par frontières globales et à un algorithme *Brick & Mortar*.

Les approches par frontières peuvent être mises en œuvre avec différentes stratégies d'affectation et différentes conditions de réaffectation. Nous avons choisi la stratégie d'affectation *MinPos* présentée dans [4], et réassignons les frontières toutes les d_{max} itérations (d_{max} est le paramètre utilisé dans l'algorithme local, et correspond au nombre maximum de déplacement entre chaque changement de cellule objectif, donnant ainsi une condition de réaffectation proche). La stratégie d'affectation *MinPos* consiste à calculer, pour chaque robot i et chaque frontière j, un rang $r_{i,j}$, correspondant au nombre de robots plus proches de la frontière j que le robot i, puis à attribuer à chaque robot la frontière la plus proche pour laquelle son rang est minimal.

Il existe également différentes approches *Brick & Mortar*. Celle que nous avons implémenté ici, *BMILRV* (Brick & Mortar Improved Long Range Vision) [1], permet de considérer un ensemble de cellules dans le rayon de perception du robot, et pas seulement la cellule sur laquelle le robot se trouve, ce qui permet une comparaison plus directe avec notre approche locale, considérant à un instant donné pour un robot le même nombre de cellules voisines.

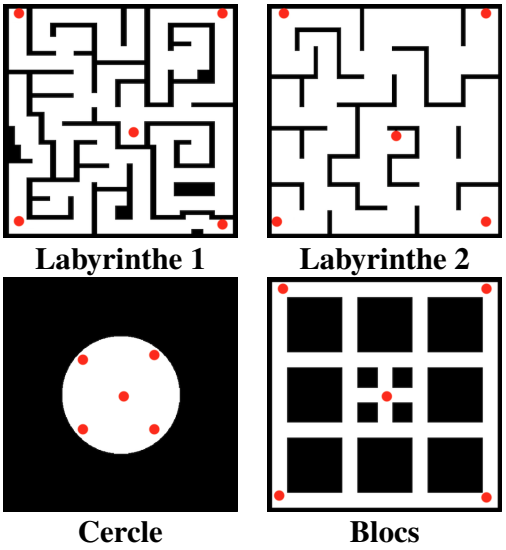


FIGURE 4 – Cartes testées (les points rouges correspondent aux points de départ des robots).

4.2 Conditions expérimentales

Nous allons comparer les trois approches afin d'explorer une carte inconnue. L'approche par frontières globales reproduit l'expérience décrite dans [7] avec réaffectation de l'objectif tous les d_{max} changements de cellule, et la stratégie d'assignation *MinPos*. Comme cette approche ne fournit pas de point de rendez-vous, nous considérerons le nombre total d'itérations jusqu'à ce que la carte soit entièrement visitée. Nous observerons ainsi le pourcentage de cellules explorées par rapport au nombre d'itérations passées. L'approche *BMILRV* est une reproduction de l'algorithme décrit dans [1]. Enfin, l'approche par frontières locales correspond à l'algorithme décrit dans la partie 3.2.

Cette partie présente les résultats expérimentaux réalisés sur simulateur pour 4 cartes différentes présentées figure 4. Les points de départ des robots sont représentés par les points rouges. Dans ces expériences, la carte est partagée à chaque itération ($d_{\text{sync}} = 1$, voir section 5 pour la version distribuée), le rayon de perception ρ est de 5 cellules, et $d_{\text{max}} = 2$.

Le *labyrinthe 2* est celui utilisé dans [1], le *labyrinthe 1* est une variante avec des couloirs plus étroits. La carte *Cercle* permet de tester un environnement avec peu d'obstacles (par rapport au rayon de perception des robots), et la carte *Blocs* correspond à un environnement avec de nombreux obstacles isolés, formant ainsi des boucles

pour les robots.

4.3 Simulateur utilisé

Pour obtenir une première comparaison expérimentale des performances des différents algorithmes, nous avons implémenté un simulateur en Python permettant d'exécuter les algorithmes dans des conditions identiques sur les 4 cartes présentées figure 4.

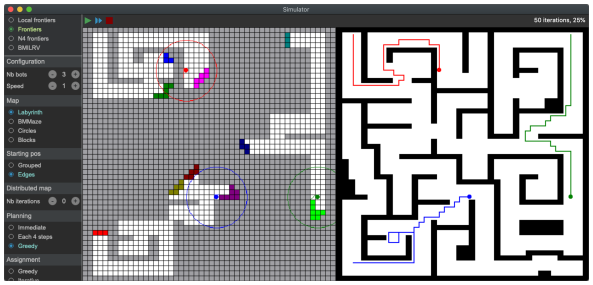


FIGURE 5 – Simulateur utilisé pour les expériences, ici avec l'algorithme par frontières globales. La partie gauche montre la carte des robots (cellules explorées en blanc, inexplorées en gris et frontières en couleurs), et la partie droite le monde « réel » avec les trajectoires des robots. Le menu de gauche permet de sélectionner les différents algorithmes (frontières globales, frontières locales, BMILRV) et les conditions de l'expérience (points de départ robot, cartes, synchronisation).

Notre simulateur (capture d'écran 5) permet une exécution décentralisée des algorithmes des robots. Dans nos expériences, nous supposons que la localisation est parfaite, que les robots ont une taille plus petite que la taille d'une cellule dans la grille d'occupation et que la détection des obstacles est toujours correcte.

4.4 Durée de l'exploration

La figure 6 représente le nombre d'itérations nécessaires pour explorer la carte entière avec $m = 3$ robots pour les différentes approches.

L'approche *BMILRV* est beaucoup plus lente que les approches par frontières locales et globales, surtout lorsqu'il y a des obstacles isolés (forçant l'algorithme *BMILRV* à contrôler les cycles), ce qui est le cas des cartes *Labyrinthe 1* et *Blocs*. Les deux approches frontières donnent des résultats comparables, l'approche globale restant légèrement meilleure.

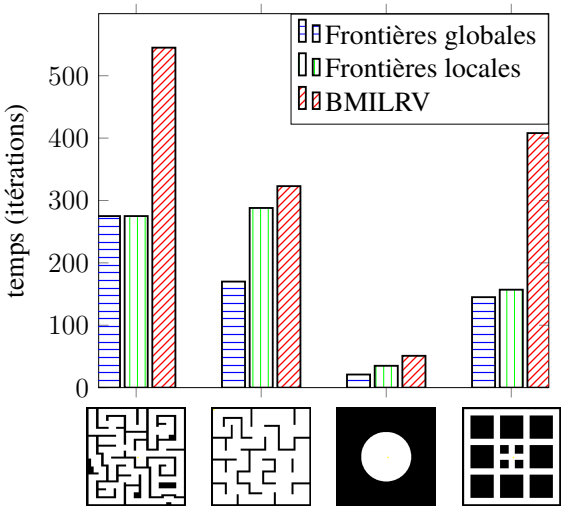


FIGURE 6 – Durée totale de l'exploration pour chaque approche sur les cartes testées avec $m = 3$ robots.

L'approche par frontières locales donne parfois une moins bonne dispersion des robots, et donc de moins bonnes performances, en particulier sur la carte *Labyrinthe 2*. Plusieurs améliorations peuvent être envisagées pour se rapprocher des performances de l'approche globale (voir 7.2).

4.5 Progression de l'exploration

La figure 7 présente l'évolution du pourcentage de cellules vues au cours du temps (en nombre d'itérations) pour les approches par frontières locales et globales, sur les deux premières cartes.

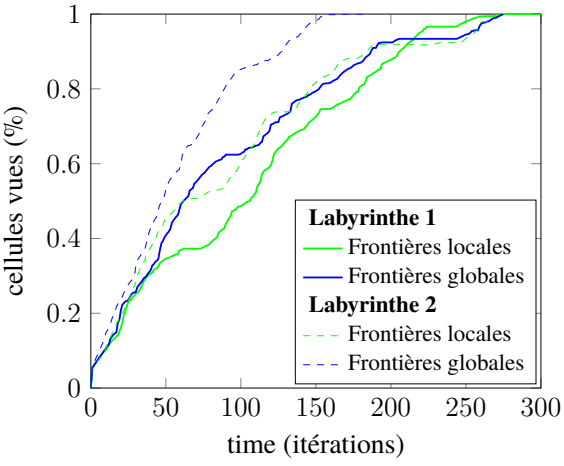


FIGURE 7 – Évolution du pourcentage de cellules vues au cours du temps ($m = 3$ robots) sur les deux premières cartes.

L'algorithme local perd facilement du temps lorsque les robots reviennent en arrière parce qu'ils prennent le même chemin que lors de leur premier passage, contrairement à l'algorithme global qui calcule le meilleur chemin pour atteindre une frontière. Cependant, les deux algorithmes montrent des résultats similaires, les deux courbes se croisent plusieurs fois pour la carte *Labyrinthe 1*.

4.6 Nombre de robots

La figure 8 montre le nombre d'itérations nécessaires pour explorer les 4 cartes pour différents nombres de robots. Sur ces 4 cartes, les approches par frontières locales et globales donnent des résultats comparables.

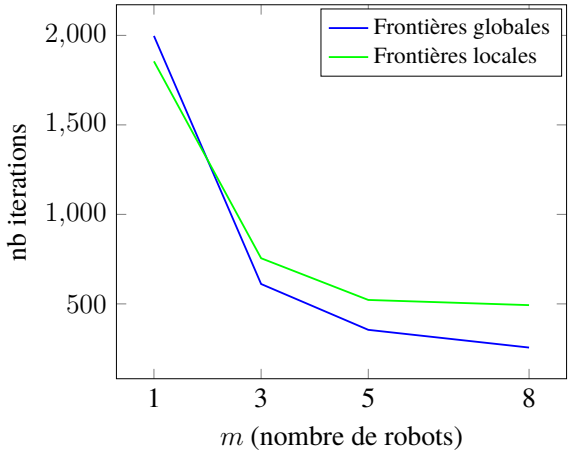


FIGURE 8 – Nombre d'itérations nécessaires pour explorer toutes les cartes pour différents nombres de robots.

Lorsque le nombre de robots augmente, l'algorithme par frontières globales prend l'avantage en offrant une meilleure répartition des robots sur la carte.

5 Version distribuée

Cette partie se concentre sur la version décentralisée de l'algorithme par frontières locales et tente d'évaluer l'impact de la décentralisation sur les performances.

5.1 Procédure de partage de carte

Le partage de la carte est nécessaire pour deux raisons : la grille d'occupation permet aux robots de définir les frontières locales ; si elle n'est pas partagée, les robots ne peuvent pas savoir si autre un robot a déjà exploré une partie donnée de la carte. De plus, la trace est également partagée pour permettre une meilleure répartition des robots, car elle est utilisée dans le choix de la frontière à définir comme objectif.

Toutes les d_{sync}itérations, chaque robot peut envoyer sa grille d'occupation $\mathcal{O}$ et ses cellules visitées $\mathcal{P}$ à tous les autres robots. Lorsqu'un robot r_i reçoit des données d'un autre robot r_j (grille d'occupation $\mathcal{O}_{r_j}$ et cellules visitées $\mathcal{P}_{r_j}$), il met à jour sa grille d'occupation de la façon suivante :

$$\mathcal{P}_{r_i}(x, y) = \mathcal{P}_{r_i}(x, y) \vee \mathcal{P}_{r_j}(x, y)$$

Un *ou* logique est appliqué entre la valeur du robot et la valeur reçue : une cellule est visitée si le robot ou tout autre robot l'a déjà marquée comme visitée.

Il n'est pas nécessaire que les robots soient synchronisés pour partager leurs cartes en même temps, ils peuvent les envoyer ou les recevoir à tout moment.

5.2 Résultats expérimentaux sur l'effet de la décentralisation sur la durée de l'exploration

La figure 9 montre le nombre d'itérations nécessaires pour explorer les 4 cartes avec 3 robots pour différentes valeurs d_{sync}.

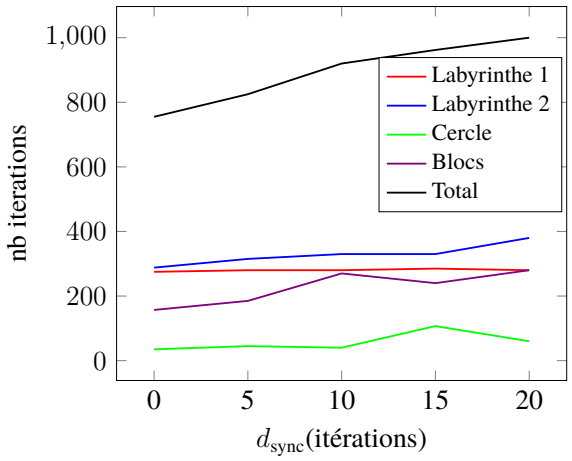


FIGURE 9 – Évolution de la durée de l'exploration des 4 cartes pour différentes valeurs de d_{sync}

L'augmentation de la durée de l'exploration lorsque d_{sync}augmente est due à une moins bonne

dispersion, notamment lorsque plusieurs robots sont proches les uns des autres : ils explorent alors la même zone simultanément. Concrètement, si l'on suppose que deux robots donnés partagent leur carte d'autant plus souvent qu'ils sont proches, cette diminution sera alors limitée la plupart du temps (la présence de boucles peut cependant faire explorer une même zone à deux robots éloignés).

5.3 Perte de communication

Bien qu'augmenter d_{sync}diminue les performances de l'exploration, une perte de communication, ou une communication éparse ne causera pas de problème dans l'exploration (« oubli » d'une zone, robot bloqué,...) et ne peut, dans le pire des cas, que prolonger la durée de l'exploration. Dans des applications concrètes, le partage de cartes entre robots peut être adapté en fonction de leur vitesse et de leur rayon de perception, éventuellement en fonction de la distance entre les robots.

6 Amélioration : « seconde chance »

Il peut arriver lors de l'exploration qu'un ou plusieurs robots retournent à leur point de départ et ne trouvent plus de frontières visibles avant que l'exploration soit terminée, laissant les robots encore actifs terminer seuls. Lorsque cela se produit, l'exploration reste complète, mais les robots revenus à leur point de départ ne participent plus, ce qui peut augmenter la durée totale de l'exploration.

Pour éviter cette perte de performances, nous pouvons envisager un système de « seconde chance », autorisant ces robots à repartir explorer les zones restantes en suivant les traces des autres robots. Nous allons donc modifier l'algorithme décrit dans la partie 3.2.

6.1 Algorithme avec seconde chance

Pour permettre aux robots de repartir, nous allons ajouter une variable *mode*, assigné à « *normal* » par défaut, et qui vaudra « *secondeChance* » lorsqu'un robot repart en suivant les traces des autres robots.

Pour activer le mode seconde chance, nous modifions le point 2. (b) de l'algorithme :

2. (b) Si $|\mathcal{F}| = 0$ et si on est de retour à la cellule de départ :
   i. Si il existe au moins un autre robot qui ne s'est pas arrêté et qui n'est pas en mode seconde chance, passer en mode seconde chance : *mode* $\longleftarrow$ *secondeChance*. Se déplacer vers une cellule voisine déjà visitée par un autre robot, si possible parmi celles que l'on a pas soit-même visité. Si on voit à nouveau une frontière, repasser en mode normal.
   ii. Sinon, s'arrêter.

6.2 Évolution de l'exploration

Nous allons maintenant comparer trois algorithmes : frontières globales (avec *MinPos*), frontières locales sans seconde chance, et frontières locales avec seconde chance. Cette fois, nous choisissons $m = 10$ robots qui partirons tous du centre du labyrinthe.

La figure 10 présente l'évolution de l'exploration pour les trois algorithmes sur le premier labyrinthe. Sur cette carte, où les approches globales et locales donnaient des résultats très proches pour $m = 3$ robots, l'algorithme local prend l'avantage avec la seconde chance, finissant l'exploration avant l'algorithme global.

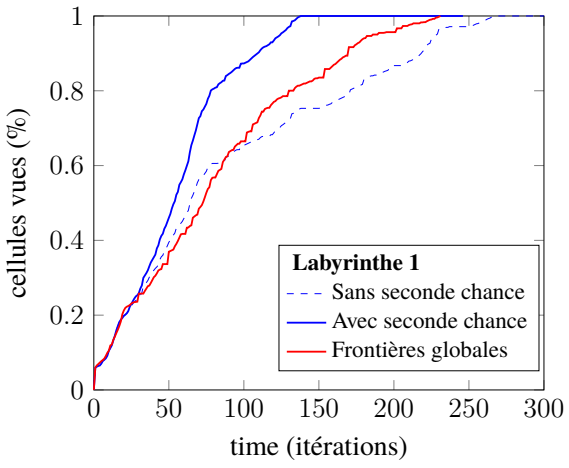


FIGURE 10 – Évolution du pourcentage de cellules vues au cours du temps ($m = 10$ robots) avec et sans seconde chance.

6.3 Durée de l'exploration

Nous allons maintenant nous intéresser à la durée totale de l'exploration des quatre cartes par les trois algorithmes, pour $m \in \{3, 5, 8\}$.

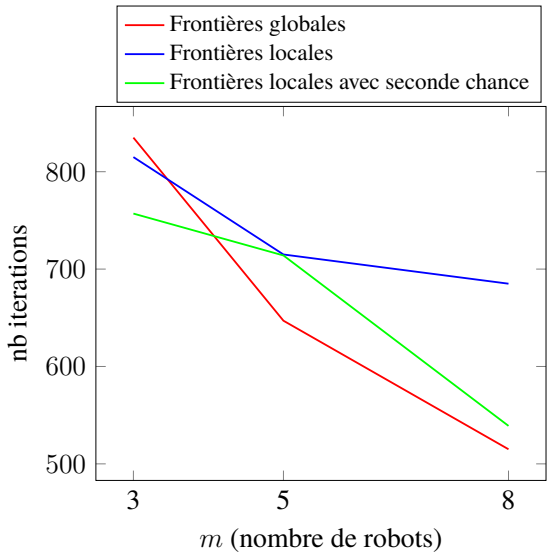


FIGURE 11 – Nombre d'itérations nécessaires pour explorer toutes les cartes pour différents nombres de robots. Les robots partent ici groupés au centre des cartes à explorer.

La figure 11 présente le nombre d'itérations nécessaires pour explorer les quatre cartes avec 3, 5 ou 8 robots. L'approche locale s'avère nettement meilleure avec la seconde chance pour 3 ou 8 robots et s'approche des performances de l'approche globale.

Notons que nous mesurons ici la durée nécessaire pour parcourir tout le labyrinthe. Si nous prenons en compte le temps de retour au point de départ des robots (leur permettant notamment de savoir que l'exploration est terminée), la seconde chance peut dans certains cas allonger l'exploration.

7 Discussion

7.1 Forces et faiblesses des différents algorithmes

L'approche *BMILRV* est locale, mais a un fonctionnement complexe (de nombreux états pour les robots, la fermeture des cellules nécessite de nombreux calculs [1]) et donne de mauvais résultats lorsque des obstacles sont isolés ou qu'il y a des boucles sur la carte, forçant les robots à faire plusieurs passages pour marquer les cellules successivement comme contrôlées, fermées et finalement nettoyées (comme le confirment les expériences sur la figure 6). A notre connaissance, cette approche n'a pas été adaptée pour une mise en œuvre décentralisée.

L'approche par frontières (globales) est à la fois simple et efficace. Elle peut être distribué facilement, mais nécessite une synchronisation fréquente de la carte et des positions du robot. Avec la stratégie d'affectation *MinPos*[4], chaque robot prend en compte la position de tous les autres robots pendant chaque affectation, ce qui implique que la carte des frontières globales et la position du robot sont fréquemment partagés. De plus, le temps de calcul de cette stratégie d'affectation dépend du nombre de robots et du nombre de frontières (les distances entre toutes les paires frontière/robot sont calculées à chaque itération), ce qui peut rendre cette méthode inutilisable si le nombre de robots est trop grand, ou si la carte est trop grande.

Enfin, l'approche par frontières locales présente des résultats proches de l'approche globale dans la plupart des cas, tout en ayant une complexité constante, ne dépendant ni du nombre total de robots ni de la taille de la carte. Il est également facilement distribuable sans contrainte sur la fréquence de partage des cartes, et les robots n'ont pas besoin de connaître le nombre ou la position des autres robots. Sans l'ajout de la « seconde chance », cependant, certains robots peuvent s'arrêter avant la fin de l'exploration, réduisant ainsi la performance de l'exploration.

7.2 Possibilités d'amélioration

La version de l'algorithme décrite ici ramène au point de départ des robots qui ne trouvent plus de frontières à explorer. Selon la carte et les conditions initiales, un robot peut revenir à son point de départ et s'arrêter avant que la carte entière ne soit explorée (laissant les autres robots terminer l'exploration). La distribution entre les robots est alors moins équilibrée et les performances de l'algorithme sont réduites. Pour éviter cela, nous pouvons envisager un système de « seconde chance » (voir partie 6), permettant aux robots qui sont revenus à leur point de départ avant la fin de l'exploration de recommencer en suivant la trace de l'un des autres robots jusqu'à ce qu'il trouve une frontière.

La dispersion des robots est également rudimentaire et ne repose que sur le choix d'une cible aussi loin que possible de toute trace laissée. Lorsque plusieurs robots sont proches (dans leurs rayons de perception respectifs), et en particulier si d_{sync}est élevé, l'ajout d'un système de dispersion supplémentaire pourrait également augmenter de manière significative les performances de l'algorithme sans augmenter consi-

dérablement sa complexité.

7.3 Conclusion

Dans la version proposée ici, l'algorithme donne de bonnes performances (nombre d'étapes pour compléter la couverture) par rapport aux deux autres approches, et au bénéfice d'une complexité constante (en temps et en mémoire) pour un robot donné, ce qui permet d'utiliser un très grand nombre de robots. En effet, dans le cas de l'approche par frontières globales, les stratégies les plus efficaces, telles que *MinPos*, nécessitent de calculer autant de distances que de paires robot/frontière, ce qui peut poser problème s'il y a un trop grand nombre de robots ou si la carte est trop grande. De plus, notre approche locale est plus facilement distribuable au prix d'une perte de performance possible lorsque les robots sont proches, et ne nécessite pas de synchronisation temporelle entre les robots

Ces résultats montrent également qu'il existe des approches intermédiaires entre les algorithmes locaux et les algorithmes globaux, pouvant prendre en compte les cellules de la grille dans un rayon de perception déterminé et garantissant une exploration complète.

Références

[1] M. Andries et F. Charpillet : Multi-robot taboolist exploration of unknown structured environments. *In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5195–5201, 09 2015.

[2] M. Araya-Lopez, V. Thomas, O. Buffet et F. Charpillet : A closer look at momdps. *In 2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 197–204, Oct 2010.

[3] Antoine Bautin, Philippe Lucidarme, Rémy Guyonneau, Olivier Simonin, Sébastien Lagrange, Nicolas Delanoue et François Charpillet : Cart-O-matic project : autonomous and collaborative multi-robot localization, exploration and mapping. 5th Workshop on Planning, Perception and Navigation for Intelligent Vehicles. *In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page x, Tokyo, Japan, novembre 2013.

[4] Antoine Bautin, Olivier Simonin et François Charpillet : Minpos : A novel frontier allocation algorithm for multi-robot exploration. *In International conference on intelligent robotics and applications*, pages 496–508. Springer, 2012.

[5] W. Burgard, M. Moors, D. Fox, R. Simmons et S. Thrun : Collaborative multi-robot exploration. *In Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No.00CH37065)*, volume 1, pages 476–481 vol.1, 04 2000.

[6] João Paulo Lima Silva de Almeida, Renan Taizo Nakashima, Flávio Neves-Jr et Lúcia Valéria Ramos de Arruda : Bio-inspired on-line path planner for cooperative exploration of unknown environment by a multi-robot system. *Robotics and Autonomous Systems*, 112:32 – 48, 2019.

[7] Jan Faigl, Olivier Simonin et Francois Charpillet : Comparison of task-allocation algorithms in frontier-based multi-robot exploration. *In* Nils Bulling, éditeur : *Multi-Agent Systems*, pages 101–110, Cham, 2015. Springer International Publishing.

[8] Ettore Ferranti, Niki Trigoni et Mark Levene : Brick & mortar : an on-line multi-agent exploration algorithm. *In ICRA*, pages 761–767, 2007.

[9] Ettore Ferranti, Niki Trigoni et Mark Levene : Rapid exploration of unknown areas through dynamic deployment of mobile and stationary sensor nodes. *Autonomous Agents and Multi-Agent Systems*, 19(2):210–243, 10 2009.

[10] Enric Galceran et Marc Carreras : A survey on coverage path planning for robotics. *Robotics and Autonomous Systems*, 61(12):1258 – 1276, 2013.

[11] S. Garnier, F. Tache, M. Combe, A. Grimal et G. Theraulaz : Alice in pheromone land : An experimental setup for the study of ant-like robots. *In 2007 IEEE Swarm Intelligence Symposium*, pages 37–44, April 2007.

[12] Elizabeth A. Jensen et Maria Gini : Effects of communication restriction on online multi-robot exploration in bounded environments. *In* Nikolaus Correll, Mac Schwager et Michael Otte, éditeurs : *Distributed Autonomous Robotic Systems*, pages 469–483, Cham, 2019. Springer International Publishing.

[13] N. Mahdoui, V. Frémont et E. Natalizio : Cooperative frontier-based exploration strategy for multi-robot system. *In 2018 13th Annual Conference on System of Systems Engineering (SoSE)*, pages 203–210, 06 2018.

[14] Q. V. Nguyen, F. Colas, E. Vincent et F. Charpillet : Localizing an intermittent and moving sound source using a mobile robot. *In 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1986–1991, Oct 2016.

[15] O. Simonin, T. Huraux et F. Charpillet : Interactive surface for bio-inspired robotics, re-examining foraging models. *In 2011 IEEE 23rd International Conference on Tools with Artificial Intelligence*, pages 361–368, Nov 2011.

[16] Sebastian Thrun et Arno Bücken : Integrating grid-based and topological maps for mobile robot navigation. *In Proceedings of the National Conference on Artificial Intelligence*, pages 944–951, 1996.

[17] E. Vincent, A. Sini et F. Charpillet : Audio source localization by optimal control of a mobile robot. *In 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5630–5634, April 2015.

[18] B. Yamauchi : A frontier-based approach for autonomous exploration. *In Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97. 'Towards New Computational Principles for Robotics and Automation'*, pages 146–151, 1997.

Combiner Optimisation Stochastique et Frontières pour l'Exploration 3D avec une Flotte de Drones

A. Renzaglia[a,b] alessandro.renzaglia@inria.fr
J. Dibangoye[b] jilles-steeve.dibangoye@insa-lyon.fr
V. Le Doze[b] vincent.le-doze@inria.fr
O. Simonin[b] olivier.simonin@insa-lyon.fr

[a]Univ. Grenoble Alpes, Inria, Chroma, F-38000 Grenoble, France.

[b]INSA Lyon, CITI Lab, Inria, Chroma, Lyon, France.

Résumé

Ce papier adresse le problème de l'exploration de terrains inconnus par une flotte de drones aériens coopératifs. Nous présentons une nouvelle approche décentralisée qui alterne exploration locale par optimisation stochastique et exploration par frontières. L'approche permet à chaque robot de générer une trajectoire en fonction des données qu'il collecte et de la carte locale qu'il construit par intégration des données partagées entre agents. Dès que l'agent arrive dans un minimum local, correspondant à une position où il est entouré d'espaces déjà explorés, alors l'algorithme identifie la plus proche frontière où il se rend avant de reprendre l'optimisation locale. Avec un faible coût calculatoire, une capacité à gérer les contraintes, et une prise de décision décentralisée, l'approche est particulièrement adaptée aux applications multi-robot en environnement complexes 3D. Les résultats en simulation montrent que l'approche génère des trajectoires sures et valides qui guident les robots pour une exploration complète de l'environnement. Par ailleurs, en terme de temps d'exploration, notre approche est significativement meilleure que la méthode des frontières proches. Elle fournit des temps équivalents à la méthode gloutonne centralisée tout en étant bien moins côuteuse en calcul.

Mots-clés : *Exploration multi-robot, cartographie 3D, optimisation stochastique locale, approche frontière.*

Abstract

This paper addresses the problem of exploring unknown terrains with a fleet of cooperating aerial vehicles. We present a novel decentralized approach which alternates gradient-free stochastic optimization and frontier-based approaches. Our method allows each robot to generate its trajectory based on the collected data and the local map built integrating the information shared by its teammates. Whenever a local optimum is reached, which corresponds to a location surrounded by already explored areas, the algorithm identifies the closest frontier to get over it and restarts the local optimization. Its low computational cost, the capability to deal with constraints and the decentralized decision-making make it particularly suitable for multi-robot applications in complex 3D environments. Simulation results show that our approach generates feasible and safe trajectories which drive multiple robots to completely explore realistic environments. Furthermore, in terms of exploration time, our algorithm significantly outperforms a standard solution based on closest frontier points while providing similar performances compared to a computationally more expensive centralized greedy solution.

Keywords: *Multi-robot exploration, 3D mapping, local stochastic optimization, frontier-based approach.*

1 Introduction

L'exploration d'environnements inconnus est une des tâches où l'usage de robots mobiles peut apporter une aide cruciale aux hommes dans des scénarios complexes et critiques. En particulier, le développement rapide des drones autonomes (UAV en anglais) permet d'imaginer des missions de plus en plus complexes, comme l'exploration de vastes environnements inaccessibles pour les véhicules terrestres standards. Cependant, les solutions pour de tels scénarios se heurtent aux fortes contraintes des plateformes de petits drones : la limitation des capacités de calcul embarquées et des batteries. Cela nécessite d'explorer la définition de solutions très légères en calcul.

L'objectif de l'exploration autonome est de visiter (ou cartographier) complètement un envi-

1. Unmanned Aerial Vehicle

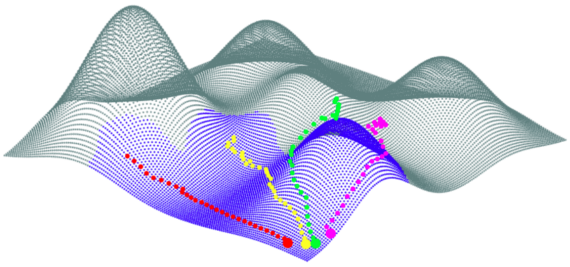


FIGURE 1 – Quatre drones débutant leur mission d'exploration d'un environnement inconnu

ronnement initialement inconnu en un minimum de temps (cf. Fig. 1). Ainsi, chaque robot doit décider de sa prochaine position afin de maximiser le gain sur la zone explorée. Comme dans toute mission multi-robot, la coordination entre agents joue un rôle crucial [19]. La problématique est de réduire les informations redondantes entre agents pour accélérer le processus d'exploration. L'approche standard pour le problème de l'exploration est fondée sur le concept de frontières, c'est-à-dire les cellules ou lignes séparant une région explorée d'un région encore inexplorée [18]. Toutefois, les solutions par frontières ne satisfont pas la contrainte pré-citée des petites plate-formes pour l'exploration aérienne d'environnements 3D. L'assignation des frontières aux robots requière une optimisation centralisée complexe et, même en considérant des stratégies heuristiques, elle demande à répéter régulièrement l'identification des frontières et le calcul des distances robot-frontières. Ce sont des opérations très lourdes en particulier pour des environnements 3D.

Pour dépasser ces limites, nous proposons une nouvelle approche où les robots sont guidés par une optimisation locale exploitant les données collectées pour maximiser l'espace nouvellement exploré, et une stratégie basée sur les frontières pour garantir la complétude de l'exploration. L'optimisation repose sur un algorithme d'optimisation stochastique exécuté indépendamment sur chaque robot et exploitant les informations de la carte construite par la flotte. L'optimisation étant strictement locale, les robots peuvent toutefois rester piégés dans des minima locaux avant de terminer leur tâche. Quand un agent détecte une telle situation, il exploite la carte globale pour identifier une frontière qu'il va rejoindre avant de reprendre son exploration locale. Il s'agit donc d'une approche hybride, combinant optimisation locale et frontières, qui vise à dépasser les limites de chaque technique et à construire une solution plus efficace fondée sur leurs avantages respectifs.

Le reste du papier est organisé comme suit. La prochaine section présente brièvement les travaux existants, avec une attention spéciale pour l'exploration multi-robot d'environnements non planaires. En Section 3, nous formalisons le problème de l'exploration multi-robot comme un problème d'optimisation. La Section 4 présente en détail l'approche proposée, en décrivant l'optimisation locale adoptée et la méthode fondée sur les frontières. Enfin, en Section 5 des expérimentations en simulation sont présentées pour évaluer l'algorithme proposé.

2 Travaux existants

Le problème de l'exploration autonome a été considérablement étudié ces dernières années, rendant difficile la synthèse d'une bibliographie riche. Une tentative de synthèse proposée par [9], offre une analyse comparative étendue des principales méthodes existantes. Une autre revue de la litérature proposée par [1] s'est focalisée sur les aspects liés à la communication.

La pluspart des méthodes existantes pour l'exploration autonome sont basées sur le concept de frontières. Ce concept a été introduit initialement par Yamauchi d'abord dans le cadre mono-robot [17] puis dans le cadre multi-robot [18]. Cette idée simple mais efficace a donné lieu à de nombreuses variantes et améliorations dans les années suivantes, en particulier dans le cadre des environnements 2D. Une approche centralisée, efficace mais coûteuse, a été introduite par [5] afin de résoudre un problème multi-critère considérant à la fois les coûts et les gains associés aux trajets. L'algorithme MINPOS [3] propose une approche d'allocation décentralisée des points frontières suivant un ordre sur les membres de la flotte de robots. Cet ordre, basé sur la distance à chaque frontière, permet d'obtenir une distribution spatiale homogène des robots dans l'environnement. Dans [7], les auteurs proposent une reformulation du problème d'exploration par frontières comme plusieurs problèmes de voyageur de commerce, c'est à dire un pour chaque robot et les points frontières encore disponibles. Bien que cette stratégie d'exploration conduise à un temps d'exploration plus court par rapport à d'autres stratégies, son coût calculatoire exorbitant rend son application limitée, contrairement à notre proposition. Une approche alternative, pour faire face au problème d'allocation des

points frontières de façon distribuée, repose sur une reformulation en un problème de satisfaction de contraintes distribuée (DSCP), comme proposée dans [13]. Au-délà du problème d'allocation des points frontières, un problème tout aussi important est celui portant sur le besoin de communication durant la tâche. Cette problématique est notamment addressée dans [2]. Les auteurs y présentent une stratégie visant à coordonner les robots entre eux afin de former un réseau capable de satisfaire les contraintes de connectivité.

Dans le cadre des environnements 3D, une extension directe de l'approche basée frontières a été proposée dans [6]. Comme mentionné précédemment, en trois dimensions, le processus d'identification des points frontières est bien plus complexe, et a donné lieu à quelques travaux, eg. [20]. Dans un cadre simulé, [12] présente une approche par frontières centralisée visant à explorer des scènes tridimensionnelles avec une flotte de micro-véhicles aériens munis de caméras. Notre approche se distingue par une solution distribuée, et par des communications entre robots qui se limitent à l'échange des points frontières et non des cartes.

La solution que nous proposons dans cet article a pour objectif de dépasser le recours systématique au calcul des points frontières à chaque pas de décision. Il s'agit de réduire le coût total de calcul. Le recours aux points frontière est significativement réduit par la combinaison avec un algorithme d'optimisation locale décentralisé. La solution résultante guide les robots vers une position maximisant l'espérance du gain d'information, tout en assurant que la tâche d'exploration sera complètement réalisée.

3 Exploration 3D multi-robot

Le problème de planification multi-robot considéré dans ce papier porte sur l'exploration autonome, par une flotte de drones aériens, d'un terrain inconnu représenté par une surface $\mathcal{S}$ dans un environnement 3D, en un temps minimum. Formellement, ce problème est défini par un tuple $(N, \mathcal{S}, \mathcal{P}, \mathcal{C}, E)$ où :

- N est le nombre de robots $r \in \{1, 2, \dots, N\}$;
- $\mathcal{S}$ est un ensemble *inconnu* de points 3D (voxels) notés $s \in \mathcal{S}$;
- $\mathcal{P}$ est l'ensemble des positions $\mathbf{P}$, où $\mathbf{P}_t^{(r)} \doteq [x_{1,t}^{(r)}, x_{2,t}^{(r)}, x_{3,t}^{(r)}]$ est la position du drone r à l'étape t ;
- $\mathcal{C}(\mathbf{P}^{(r)}) \preceq 0$ défini les contraintes *partiellement inconnues* que les positions $\mathbf{P}^{(r)}$ de chaque drone doit satisfaire afin d'éviter les collisions drone-obstacle et drone-drone ainsi que la satisfaction des contraintes portant sur l'altitude minimale et maximale de vol ;
- $E_t(S)$ est la portion de la surface S explorée jusqu'à l'instant t. Un point s est considéré exploré par un agent si : (1) ils sont connectés par la ligne de vue et (2) ils sont à une distance plus petite que la portée maximale du champs de vision.

Soit $e_t(s)$ le sous-ensemble d'agents qui explore pour la première fois le voxel s à l'instant t, c-à-d

$$e_t(s) = \{r \in \mathcal{R} \mid s \in E_t(S) \backslash E_{t-1}(S), O(s, \mathbf{P}_t^{(r)}) = 1\}$$

où $O(s, \mathbf{P})$ est vrai lorsque le point s est observé par un agent positionné en $\mathbf{P}$. De plus, les points frontières sont définis comme les voxels déjà explorés $s \in E(\mathcal{S})$ qui sont adjacents aux régions non-explorées, c-à-d $\mathcal{S} \backslash E(\mathcal{S})$.

L'objectif de ce problème est de trouver les trajectoires des robots $(\mathbf{P}_t^{(r)})_{r=1,\dots,N;t=1,\dots,t}$ qui permettent d'explorer la totalité de la surface S en un temps minimum T, tout en satisfaisant les contraintes $\mathcal{C}$. Ce problème est impossible à résoudre hors-ligne étant donné que l'environnement est initialement inconnu ; ainsi une solution en ligne, même sous-optimale, est nécessaire. Pour y parvenir, nous définissons pour chaque robot r une fonction d'optimisation comme suit :

$$J_t^{(r)} = \sum_{s \in E_t(\mathcal{S})} \frac{\delta_s^r}{|e(s)|}, \quad \delta_s^r = \begin{cases} 1 & \text{if} \quad r \in e(s) \\ 0 & \text{if} \quad r \notin e(s) \end{cases} \tag{1}$$

Afin de maximiser cette fonction, chaque drone tente à chaque itération de maximiser la quantité de nouveaux voxels s (composante δ_s^r) explorés, tout en minimisant les informations redondantes (c-à-d l'exploration simultanée de nouvelles régions par plus d'un drone). Ceci est dû au terme $|e(s)|$, qui représente la pénalisation utile pour réduire l'importance des voxels qui ont été explorés simultanément par plusieurs drones. En d'autres mots, chaque voxel a une récompense fixée qui est répartie sur l'ensemble des agents qui l'ont vu en premier. Cette récompense a pour valeur maximale 1, lorsqu'un seul agent l'a exploré. Cette stratégie d'attribution des récompenses vise à forcer la répartition des agents, afin de minimiser la redondance des informations collectées. À noter que la somme sur l'ensemble des drones des fonctions $J_t^{(r)}$ est simple-

ment le nombre de voxels exploré jusqu'à l'instant t, c-à-d

$$\sum_{r\in N} J_t^{(r)} = \sum_{r\in N}\sum_{s\in E_t(\mathcal{S})} \frac{\delta_s^r}{|e(s)|} = |E_t(\mathcal{S})|\,. \quad (2)$$

4 L'algorithme FCAO

Dans cette section, nous présentons un nouvel algorithme d'exploration multi-robots combinant une optimisation stochastique basée sur une approximation locale de la fonction objectif avec une approche basée sur les frontières. Nous nommons cet algorithme Frontier-based Cognitive Adaptive Optimization (FCAO) (Algorithme 1). L'approche décentralisée qui en résulte a pour objectif de surmonter les faiblesses des deux méthodes utilisées tout en préservant leurs points forts.

Algorithm 1: FCAO

```
function FCAO(w, α, E)
    Initialiser t, δt = [t − w : t] and P_δt.
    repeat
        CAO(t, P_δt, E, α)
        P_{t+1} ← ClosestFrontier(P_t, E)
        MoveTo(P_{t+1})
        t ← t + 1
    until P_t = P_{t−1};
function CAO(t, P_δt, E, α)
    repeat
        Maj J(E) avec nouveaux points
        Maj ϑ_t selon équation (5).
        Générer points (ζ^(m))_{m∈{1,...,M}}
         aléatoirement.
        Sélectionner m* selon équation (7)
        P_{t+1} ← P_t + αζ^(m*)
        MoveTo(P_{t+1})
        t ← t + 1
    until P_t = P_{t−1};
function ClosestFrontier(P_t, E)
    F = CandidateFrontiers(E)
    return SelectClosestFrontier(F, P_t)
```

4.1 Optimisation stochastique basée sur une approximation locale

Trouver un vrai optimum de (1) est impossible en pratique, car l'environnement est initialement inconnu et l'optimisation doit être poursuivie en ligne pendant le processus d'exploration. Cependant, chaque robot peut calculer les valeurs numériques de (1) à chaque pas de temps en fonction des données collectées. Dans ce cadre, les algorithmes d'optimisation stochastique basés sur une approximation locale de la fonction objectif d'origine sont des outils puissants pour contourner ce problème. C'est le cas, par exemple, de l'algorithme CAO, développé et analysé à l'origine par Kosmatopoulos dans [11], qui a récemment été proposé, dans diverses versions, pour trouver des solutions à des problèmes multi-robots, tels que : le déploiement optimal de plusieurs robots [14], [15], la détection de cible [16] et la localisation et cartographie multi-robots [10]. Dans cette section, nous présentons une version modifiée de cet algorithme d'optimisation pour le rendre approprié au problème considéré.

La solution proposée représente un moyen efficace de résoudre en temps réel des problèmes d'optimisation sous contrainte où la forme exacte de la fonction objectif $\mathcal{J}$ n'est pas disponible, mais dont les valeurs numériques peuvent être récupérées en exploitant les informations collectées. L'algorithme CAO est basé sur deux étapes principales : *i)* à chaque itération, une fonction d'approximation est construite pour avoir une estimation locale de la fonction objectif inconnue J ; *ii)* la position suivante du robot est sélectionnée de manière à maximiser la fonction d'approximation en respectant les contraintes du problème. Ces deux étapes sont détaillées ci-dessous.

1. A chaque instant t, la fonction J est estimé comme suit :

$$\hat{J}_t^{(r)}\left(x_{1,t}^{(r)}, x_{2,t}^{(r)}, x_{3,t}^{(r)}\right) = \vartheta_t^T\,\phi\left(x_{1,t}^{(r)}, x_{2,t}^{(r)}, x_{3,t}^{(r)}\right) \quad (3)$$

où ϑ_t désigne le vecteur des paramètres estimés, calculés aux instants t, et ϕ le vecteur des fonctions de régression. Dans notre cas, nous avons choisi cette fonction ϕ comme un polynôme de troisième degré des variables d'état, c'est-à-dire :

$$\phi = 1 + \sum_{i=1}^{3} x_{i,t}^{(r)} + \sum_{i=1}^{3}\sum_{j\geq i}^{3} x_{i,t}^{(r)} x_{j,t}^{(r)} + \sum_{i=1}^{3}\sum_{j\geq i}^{3}\sum_{k\geq j}^{3} x_{i,t}^{(r)} x_{j,t}^{(r)} x_{k,t}^{(r)}\,. \quad (4)$$

Ce choix est arbitraire et d'autres solutions peuvent également être adoptées. Cependant, il convient de noter que cette approximation est strictement locale et qu'elle n'a aucune ambition

de reproduire la fonction originale dans tout l'espace d'exploration. Son rôle est exclusivement de fournir une estimation de J pour permettre à chaque robot de sélectionner sa prochaine position. Le vecteur des paramètres d'estimation ϑ_t, de dimension 20 dans notre cas, est ensuite calculé à chaque instant t sur la base d'un ensemble limité de mesures antérieures comme suit :

$$\vartheta_t = \underset{\vartheta}{\operatorname{argmin}} \frac{1}{2} \sum_{\ell=\ell_t}^{t-1} \left(J_\ell^{(r)} - \vartheta^T \phi \left(\mathbf{P}_\ell^{(r)} \right) \right)^2 \quad (5)$$

où $\ell_t = \max\{0, t - L - T_h\}$, avec T_h étant un entier non négatif défini par l'utilisateur, et $J_\ell^{(r)}$ la valeur numérique de la fonction objectif au temps ℓ. Des algorithmes standard d'optimisation des moindres carrés peuvent être utilisés pour résoudre (5). Notez que l'utilisation d'un ensemble limité de valeurs antérieures présente l'avantage de réduire considérablement le temps de calcul nécessaire pour obtenir le vecteur ϑ_t, mais la fonction résultante est une approximation fiable uniquement localement et ne peut pas être utilisée pour une optimisation globale.

2. Dès que l'estimateur $\hat{J}_t$ est construit selon (3) et (5), la nouvelle position du robot est obtenue par une recherche aléatoire. Un ensemble de nouvelles positions admissibles est généré en perturbant de manière aléatoire l'état actuel et directement testé sur $\hat{J}_t$. Plus formellement, un ensemble de M configurations d'états candidats est construit selon :

$$\mathbf{P}_t^{(r,m)} = \mathbf{P}_t^{(r)} + \alpha\, \zeta_t^{(r,m)}, \forall m \in \{1, \ldots, M\}, \quad (6)$$

où $\zeta_t^{(r,m)}$ est un vecteur aléatoire unitaire de moyenne zéro avec une dimension égale à celle de $\mathbf{P}_t^{(r)}$ et α représente le pas d'exploration pour chaque itération. Notez que ce paramètre, dans les algorithmes d'optimisation, dépend généralement du temps pour assurer la convergence vers un optimum local (voir, par exemple, [4]). Dans notre cas, nous n'avons pas cette nécessité puisque dans une tâche d'exploration, il n'y a aucun intérêt à se déplacer avec un pas inférieur au pas maximal réalisable et il n'y a pas de convergence tant que la totalité de l'environnement n'est pas explorée. Parmi toutes les M nouvelles configurations candidates $\mathbf{P}_t^{r,m}$, celles qui correspondent à des positions non réalisables, c'est-à-dire celles qui violent les contraintes $\mathcal{C}$, sont négligées et la nouvelle position du robot sera $\mathbf{P}_{t+1}^{(r)} = \mathbf{P}_t^{(r,m^*)}$ où m^* est calculé comme suit :

$$m^* = \underset{m \in \{1,\ldots,M\}}{\operatorname{argmax}} \hat{J}_t^{(r)} \left(\mathbf{P}_t^{(r,m)} \right). \quad (7)$$

Le choix aléatoire des candidats est essentiel et crucial pour l'efficacité de l'algorithme, car ce choix garantit que $\hat{J}_t$ soit une estimation fiable et précise de la fonction inconnue J. Enfin, précisons que, dans cet article, nous nous concentrons uniquement sur le problème de planification de haut niveau sans prendre en compte explicitement les contraintes de dynamique pour les robots. Ce choix peut également être justifié en supposant que les nouveaux points de déplacement sont suffisamment éloignés pour être toujours accessibles par le robot. Cependant, des hypothèses plus restrictives, y compris sur la dynamique du robot, réduisant l'espace des prochaines positions possibles n'affecteraient que la définition de $\mathcal{C}$, mais pas l'algorithme.

4.2 Plus proche frontière (closest frontier)

Comme indiqué précédemment, l'optimisation proposée est strictement locale et, même si elle peut trouver très efficacement une solution admissible à un problème de génération de trajectoire sous contrainte, elle nécessite de mesurer des variations dans les valeurs de la fonction d'optimisation pour fournir de meilleures solutions à chaque itération. En d'autres termes, comme chaque méthode d'optimisation locale, elle ne peut pas surmonter les optima locaux sans informations externes. Dans une tâche d'exploration, cela signifie que si les mouvements du robot apportent de nouvelles zones explorées, l'algorithme fournit une direction claire à suivre, mais lorsqu'il reste entouré de régions déjà explorées, la méthode d'optimisation ne peut fournir qu'un mouvement aléatoire. Ce n'est que dans ce cas-là que nous calculons l'ensemble des frontières disponibles et que le robot doit se déplacer à la plus proche. Cela nécessite de calculer chaque distance drone-frontière dans l'espace 3D, puis de sélectionner la plus petite (adaptation de l'approche standard [17] aux espaces 3D). Une telle attribution de frontière permet au robot de surmonter efficacement les blocages, acquérir de nouvelles informations et relancer l'optimisation locale à partir de celle-ci. Plus globalement, alterner explorations locales et assignations de frontières permet de garantir l'exhaustivité de l'exploration, comme pour les autres méthodes fondées sur les frontières.

Cette étape utilise une carte globale que chaque

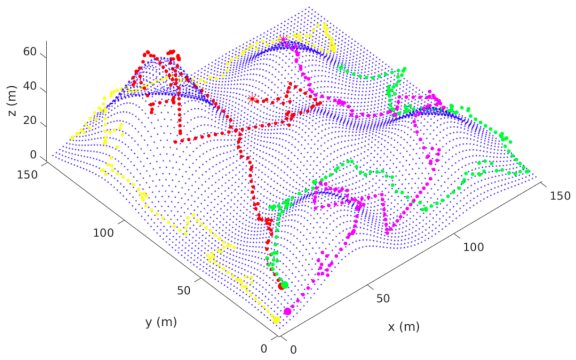


FIGURE 2 – Trajectoires réalisées par quatre robots lors de l'exploration d'un terrain inconnu.

robot construit en intégrant les informations reçues de ses coéquipiers. Nous nous référons à [18] pour une description plus détaillée de cette stratégie et pour les détails sur la communication nécessaire à la construction de cette carte.

L'aspect important de cette approche est que, même si l'information est partagée, la décision reste décentralisée. Cela garantit en particulier la robustesse vis-à-vis de l'échec des robots car, si un robot cessait de communiquer, les robots restants continueraient à explorer tout l'environnement. De plus, en tant que système complètement asynchrone, les robots n'ont pas besoin d'attendre d'autres robots et la perte d'un ou de plusieurs robots ne provoque aucune interruption de l'algorithme.

Le désavantage de cette approche est la possibilité d'une redondance dans l'exploration, c'est-à-dire que plusieurs robots peuvent être affectés à la même frontière. Cet inconvénient existe dans notre approche mais il est largement limité par le fait que les frontières ne sont utilisées que rarement et de manière asynchrone par les agents.

5 Résultats de simulations

L'approche proposée a été testée à partir de simulations puis comparée aux solutions standards de la littérature. Les terrains à explorer sont générés selon une répartition aléatoire de courbes gaussiennes variants en hauteur et en écart type (cf. exemple Fig. 2). Les robots doivent respecter une hauteur minimale relative au sol au cours du vol ainsi qu'une distance de sécurité avec les autres robots afin d'éviter toute collision. Nous avons fait comme hypothèse de simulation que chaque robot dispose d'une capacité de détection omnidirectionnelle avec une portée limitée.

La figure 2 illustre un exemple de mission d'exploration réalisée par 4 robots en montrant les trajectoires réalisées. Pour tous les résultats de simulation répertoriés dans cette partie les positions initiales des robots peuvent légèrement varier, elles sont cependant toujours proches les unes des autres dans un des coins de la zone à explorer.

Afin d'évaluer quantitativement les performances de l'approche proposée nous la comparons à deux autres techniques standards : *i)* un algorithme décentralisé d'exploration de la plus-proche frontière, dans lequel chaque robot sélectionne indépendemment de ses coéquipiers le point frontière le plus proche de lui-même (noté closest frontier) ; *ii)* un algorithme centralisé du type glouton, qui assigne - séquentiellement - a chaque robot un point frontière unique prélevé d'une liste (noté greedy). Dans ce dernier, les points frontières sont tout d'abord séparés en différents groupes pour réduire au maximum le chevauchement des trajectoires des robots, puis seuls les centroides de ces groupes sont pris en comptes.

Un premier résultat avec 8 robots, démarrant du même ensemble de positions initiales, est présenté figure 3. Dans un premier temps, nous pouvons observer que notre approche est bien plus performante que l'algorithme closest-frontier pour réaliser l'exploration complète. Ceci se justifie par le haut niveau de redondance de l'exploration guidée uniquement par la sélection des points frontières les plus proches. Dans notre approche, l'optimisation locale déploie efficacement l'équipe en évitant le recoupements des informations acquises. Dans un deuxième temps, nous pouvons voir que malgré son aspect décentralisé notre méthode est capable de réaliser une performance très proche de la méthode gloutonne qui est centralisée.

Dans le même scénario, nous étudions un autre aspect important : le temps de calcul nécessaire pour trouver la nouvelle position objectif d'un robot à chaque itération de l'algorithme. Comme indiqué précédemment, l'un des point fort de notre approche est le faible coût calculatoire. Celui-ci est clairement illustré par la figure 3 (en bas), avec un facteur temps non seulement significativement inférieur au temps requis par l'algorithme glouton mais aussi meilleur que celui de l'algorithme très simple de plus proche frontière. La différence principale se situe dans le besoin de l'approche fondée frontière de calculer à chaque itération les distances entre la position du robot et celles de tous les points

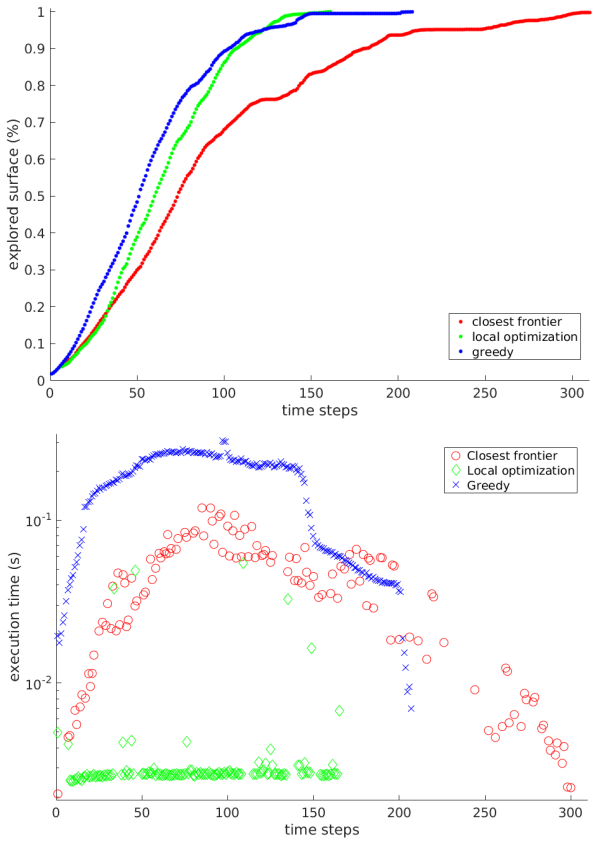


FIGURE 3 – L'approche proposée est comparée à deux alternatives : L'algorithme décentralisé de sélection de la plus proche frontière (closest) et l'algorithme centralisé d'affectation gloutonne des points frontières (greedy). Les résultats correspondent à un scénario avec 8 robots évoluant dans un environnement similaire à celui de la figure 2. En haut : Portion de l'environnement exploré en fonction du nombre de pas de temps. En bas : Temps de calcul mis pour sélectionner la position suivante à chaque itération.

frontières, ce qui peut devenir une opération très coûteuse en 3D. Dans notre cas l'effet est significatif, même si nous ne calculons pas la distance géodésique exacte entre deux points, ce qui rendrait les écarts encore plus grands. A la place nous calculons la ligne directe qui relie deux points en survolant le terrain avec une altitude respectant une contrainte de hauteur minimale par rapport au sol. Le coût de notre approche est vraiment indépendant du nombre de points frontières, comme le montre la courbe 3, où le temps d'exécution de l'optimisation locale est constant (sauf pour les rares cas ou les frontières sont exploitées) tandis que deux autres courbes varient au cours du temps à cause de leur dépendance au nombre de frontières présentes sur la

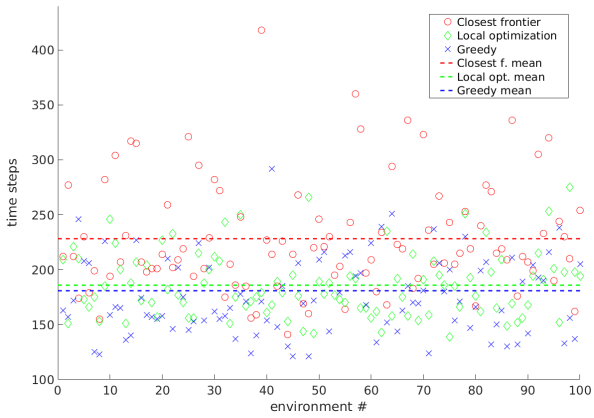


FIGURE 4 – Temps d'exploration des trois algorithmes obtenus sur 100 environnements générés aléatoirement. Les lignes en pointillés représentent la performance moyenne de chaque approche.

carte à chaque instant. De plus, en étant décentralisé, son coût est aussi indépendant du nombre de robots, contrairement à l'algorithme centralisé glouton. Le seul coût significatif de l'optimisation locale provient de la résolution de (5), qui dépend exclusivement de la dimension de la fonction ϕ et du nombre de mesures considérées dans l'historique, qui sont tout deux constants au cours de la mission.

Afin d'avoir des résultats statistiquement plus significatifs, nous avons comparé les trois méthodes sur 100 environnements générés aléatoirement (8 sommets avec des écarts-type égaux et centrés sur des positions tirées aléatoirement avec une distribution uniforme sur l'environnement). Les résultats présentés en figure 4 donnent le temps d'exploration final pour chaque essais ainsi que la moyenne de chaque algorithme. Comme dans la précédente étude, l'approche proposée par optimisation locale montre une performance très proche de celle de l'algorithme centralisé glouton. Ces deux dernières sont par ailleurs bien plus performantes que la méthode plus proches frontière. En plus d'avoir un fonctionnement moins performant en moyenne, cette dernière méthode présente aussi un écart-type plus important sur les résultats, alors que les deux autres montrent une bien meilleure robustesse par rapport aux différences entre les environnements.

5.1 Environement urbain simulé

Nous présentons maintenant les résultats obtenus avec une modélisation réaliste de l'environ-

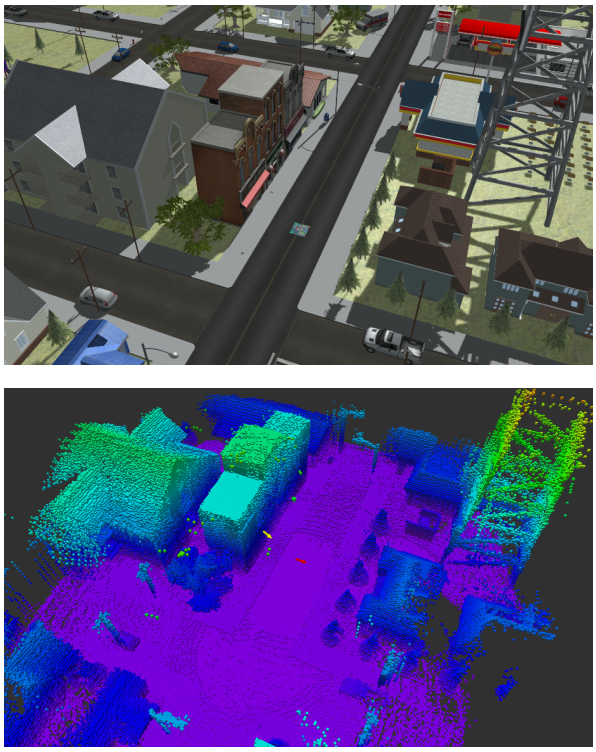


FIGURE 5 – Environnement à explorer (en haut) et carte 3D finale (en bas) construite par 2 drones exécutants l'algorithme FCAO.

nement et des drones afin d'illustrer la faisabilité et l'intéret de notre méthode pour des applications concrètes. Les simulations ont été réalisées avec les outils ROS et Gazebo, qui permettent de générer un contexte urbain riche et réaliste contenant des batiments, des infrastructures, des véhicules, etc. (cf. figure 5). La modélisation des drones a été réalisée en prenant comme référence les spécifications du quadrirotor Intel Aero. Chaque drone embarque un capteur lui permettant d'obtenir en temps réel une représentation spatiale sous forme de nuage de points de l'environement situé en dessous de lui dans une demi-sphère de rayon 20m. La cartographie de la surface explorée ainsi que la liste des coordonnées des frontières sont générés, lorsque cela est necessaire, à partir des nuages de points grâce au package ROS Octomap_server [8]. Les captures d'écran de la figure 5 illustrent l'exploration réalisée par deux drones simulés en montrant l'environnement tel que modélisé par Gazebo et la carte finale représentée par Octomap. La performance globale est illustrée par la courbe 6 où le nombre de noeuds de l'Octree est tracé en fonction du temps d'exploration des deux quadrirotors. Une vidéo explicative montrant le déroulement de la simulation est disponible au lien ci-dessous [2].

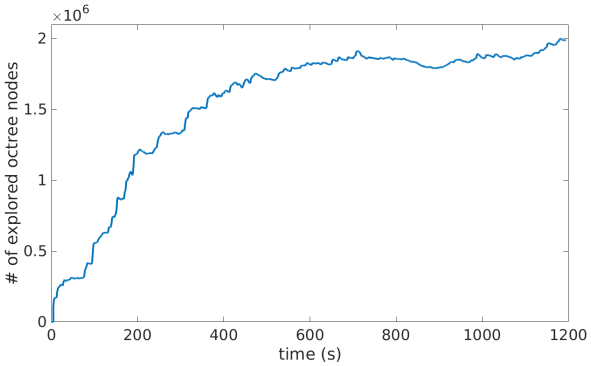


FIGURE 6 – Nombre de noeuds *octree* explorés au fil du temps dans le scénario décrit en Fig. 5.

6 Conclusions

Dans ce travail, nous avons présenté une nouvelle approche décentralisée pour l'exploration de terrains 3D avec une flotte de drones aériens, basée sur une optimisation locale de l'information acquise. L'optimisation est réalisée en adoptant un algorithme stochastique qui, avec un coût de calcul très bas, permet d'optimiser la fonction objectif uniquement sur la base des données collectées par les robots. Cette approche a été combinée avec une méthode fondée sur les frontières pour permettre de relancer l'optimisation lorsqu'un robot reste bloqué dans un optimum local. Des résultats en simulation ont prouvé l'efficacité de cette approche et une comparaison avec des approches standards basées sur les frontières a été réalisée. Cette analyse a montré que notre approche améliore considérablement les performances de algorithme décentralisé "plus proche frontière", tout en offrant des performances similaires en ce qui concerne une assignation de frontières centralisée et gloutonne beaucoup plus coûteuse. Une dernière expérimentation avec le simulateur Gazebo montre également l'applicabilité de notre approche dans un scénario plus réaliste.

Les travaux futurs porteront sur une validation plus approfondie de notre algorithme dans des environnements plus riches et variés, ainsi que sur un modèle plus réaliste des drones en prévision d'expériences dans des scénarios réels.

2. https://team.inria.fr/chroma/files/2019/05/3DMulti-UAVExploration.mp4

Références

[1] Francesco Amigoni, Jacopo Banfi, and Nicola Basilico. Multirobot exploration of communication-restricted environments : A survey. *IEEE Intelligent Systems*, 32(6) :48–57, 2017.

[2] Jacopo Banfi, Alberto Quattrini Li, Ioannis Rekleitis, Francesco Amigoni, and Nicola Basilico. Strategies for coordinated multirobot exploration with recurrent connectivity constraints. *Autonomous Robots*, 42(4) :875–894, 2018.

[3] Antoine Bautin, Olivier Simonin, and François Charpillet. Minpos : A novel frontier allocation algorithm for multi-robot exploration. In *International conference on intelligent robotics and applications*, pages 496–508. Springer, 2012.

[4] Dimitri Bertsekas and John Tsitsiklis. Gradient convergence in gradient methods with errors. *Journal on Optimization*, 10(3), 2000.

[5] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *IEEE Transactions on robotics*, 21(3) :376–386, 2005.

[6] Christian Dornhege and Alexander Kleiner. A frontier-void-based approach for autonomous exploration in 3d. *Advanced Robotics*, 27(6) :459–468, 2013.

[7] Jan Faigl, Miroslav Kulich, and Libor Přeučil. Goal assignment using distance cost in multi-robot exploration. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3741–3746. IEEE, 2012.

[8] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap : An efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, 34(3) :189–206, 2013.

[9] Miguel Juliá, Arturo Gil, and Oscar Reinoso. A comparison of path planning strategies for autonomous exploration and mapping of unknown environments. *Autonomous Robots*, 33(4) :427–444, 2012.

[10] Athanasios Kapoutsis, Savvas Chatzichristofis, Lefteris Doitsidis, Joao Borges de Sousa, Jose Pinto, Jose Braga, and Elias B Kosmatopoulos. Real-time adaptive multi-robot exploration with application to underwater map construction. *Autonomous robots*, 40(6) :987–1015, 2016.

[11] Elias B Kosmatopoulos. An adaptive optimization scheme with satisfactory transient performance. *Automatica*, 45(3), 2009.

[12] Nesrine Mahdoui, Vincent Frémont, and Enrico Natalizio. Cooperative frontier-based exploration strategy for multi-robot system. In *2018 13th Annual Conference on System of Systems Engineering (SoSE)*, pages 203–210. IEEE, 2018.

[13] Pierre Monier, Arnaud Doniec, Sylvain Piechowiak, and René Mandiau. Comparison of dcsp algorithms : a case study for multi-agent exploration. In *Advances on Practical Applications of Agents and Multiagent Systems*, pages 231–236. Springer, 2011.

[14] Alessandro Renzaglia, Lefteris Doitsidis, Agostino Martinelli, and Elias B Kosmatopoulos. Multi-robot three-dimensional coverage of unknown areas. *The International Journal of Robotics Research*, 31(6), 2012.

[15] Davide Scaramuzza et al. Vision-controlled micro flying robots : from system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robotics & Automation Magazine*, 21(3), 2014.

[16] Kuo-Shih Tseng and Bérénice Mettler. Near-optimal probabilistic search via submodularity and sparse regression. *Autonomous Robots*, 41(1), 2017.

[17] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Computational Intelligence in Robotics and Automation, 1997. CIRA'97., Proceedings., 1997 IEEE International Symposium on*, pages 146–151. IEEE, 1997.

[18] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM, 1998.

[19] Zhi Yan, Nicolas Jouandeau, and Arab Ali Cherif. A survey and analysis of multi-robot coordination. *International Journal of Advanced Robotic Systems*, 10(12) :399, 2013.

[20] Cheng Zhu, Rong Ding, Mengxiang Lin, and Yuanyuan Wu. A 3d frontier-based exploration tool for mavs. In *2015 IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 348–352. IEEE, 2015.

Diagnostic décentralisé à l'aide d'automates cellulaires

Nicolas Gauville[a,c] nicolas.gauville@loria.fr
Nazim Fatès[a] nazim.fates@inria.fr
Irène Marcovici[a, b] irene.marcovici@univ-lorraine.fr

[a] Université de Lorraine, CNRS, Inria, LORIA, F-54000 Nancy, France

[b] Université de Lorraine, CNRS, Inria, IECL, F-54000 Nancy, France

[c] Safran Electronics & Defense

Résumé

Nous nous intéressons au problème du diagnostic de défaillances dans un réseau distribué. Lorsque les composants du réseau sont susceptibles de tomber en panne, comment détecter le moment où le taux de composants défaillants dépasse un certain seuil sans faire appel à une autorité centrale ? Notre objectif est d'avoir une estimation de l'état général du réseau par le seul biais d'interactions locales des composants avec leurs voisins. En particulier, nous souhaitons qu'un consensus émerge sous forme d'état d'alerte lorsque le taux de défaillance dépasse un certain seuil. Nous utilisons le modèle des automates cellulaires pour proposer des solutions dans le cas d'un réseau ayant une structure de grille. Nous comparons trois méthodes d'auto-organisation du réseau, en partie inspirées de phénomènes physiques ou biologiques. Comme domaine d'application, nous avons en vue les réseaux de capteurs ou tout système fonctionnant de manière décentralisée.

Mots-clés : *Émergence, auto-organisation, viabilité ; résolution collective de problèmes ; déploiement de SMA, résistance aux pannes, fiabilité*

Abstract

We address the problem of detecting failures in a distributed network. If some components can break down over time, how can we detect that the failure rate has exceeded a given threshold without any central authority ? Our aim is to have an estimate of the global state of the network, only through local interactions of components with their neighbours. In particular, we wish to reach a consensus on an alert state when the failure rate exceeds a given threshold. We use the model of cellular automata in order to propose solutions in the case of a network with a grid structure. We compare three methods of self-organisation that are partly inspired by physical and biological phenomena. As an application, we envision sensor networks or any type of decentralised system.

Keywords: *Emergence, self-organisation, viability ; collective resolution of problems ; deployment of MAS, reliability to failures, fiability*

1 Diagnostic décentralisé

Nous nous intéressons ici à un réseau d'éléments tous identiques en interaction avec leurs voisins immédiats. Nous supposons que les éléments du réseau ont un état normal de fonctionnement et un état défaillant, lequel survient de manière aléatoire et irréversible. Notre objectif est de détecter de manière décentralisée le moment où le taux d'éléments défaillants dépasse un seuil critique fixé à l'avance. L'idée est de provoquer un changement de comportement global brutal lorsque ce seuil d'éléments défaillants est atteint, soit en diffusant un état d'alerte spécifique, soit en créant un *consensus global* sur l'état des éléments.

Afin de traiter ce problème dans un cadre mathématique simple, nous explorons le cas où le réseau d'éléments en interaction peut être représenté par un automate cellulaire. Le réseau est donc formé par un ensemble d'agents immobiles, les *cellules*, qui ont un état discret et sont disposées sur une grille bidimensionnelle finie. Les cellules changent leur état à chaque pas de temps en fonction d'une loi, appelée *règle de transition locale*, qui est uniforme (les cellules sont toutes régies selon la même loi). La localité de la loi exprime le fait que chaque cellule met à jour son état en fonction de son propre état et de celui des ses voisines immédiates.

Le cadre formel que nous utilisons rejoint celui des systèmes multi-agents réactifs, où les agents prennent des décisions en disposant de très peu de mémoire [11]. Ici, la structure de grille que nous avons choisie contraint fortement les interactions entre agents. Néanmoins, comme nous le

verrons par la suite, nous avons sélectionné les règles que nous présentons pour leur robustesse à des modifications de topologies qui rendraient le réseau plus irrégulier. De plus, ce cadre très simple permet de faciliter la reproductibilité des expériences. Nous avons effectué les simulations à l'aide du logiciel FiatLux , mais le modèle peut être reproduit très facilement par d'autres moyens.

La conception d'un système qui réalise de manière *décentralisée et robuste* un diagnostic pour dire si un taux critique de cellules défaillantes a été franchi à un moment donné est un problème difficile. En effet, nous travaillons dans un cadre qui conjugue trois contraintes fortes : a) localité des échanges d'information, b) uniformité de la règle de transition, c) petit nombre d'états pour les cellules. Ces trois contraintes jointes imposent donc de ne pas calculer *directement* le ratio du nombre de cellules défaillantes sur le nombre de cellules total. En effet, un tel calcul imposerait une centralisation, totale ou partielle, qui contredirait les contraintes du problème et serait en désaccord avec notre objectif, qui est d'avoir un fonctionnement *robuste* du système : si l'information était centralisée dans des noeuds particuliers, la défaillance (ou l'attaque) de tels noeuds pourrait mettre en danger l'ensemble du système.

En somme, le défi est de concevoir un système qui puisse se diagnostiquer lui-même tout en subissant une détérioration : les cellules défaillantes ne peuvent changer d'état ou transmettre d'information. On peut comparer ce problème à d'autres tâches comme la classification de la densité [4], pour laquelle il s'agit de déterminer quel est l'état majoritaire sur une grille de cellules possédant un état binaire, en respectant les mêmes contraintes (localité, uniformité de la règle de transition, petit nombre d'états). En effet, il s'agit dans les deux cas d'obtenir de manière décentralisée un consensus sur une propriété globale du système. Ici, pour ne pas être trop éloignés des cas d'application réels, nous souhaitons également que notre système soit robuste à l'asynchronisme des mises à jour des composants et qu'il ne soit pas spécifique à une forme particulière du réseau. Ce type d'approche a jusqu'ici été très peu considéré ; un point de vue similaire a cependant été adopté par Bénézit, qui a proposé différents modèles d'obtention de consensus sur des réseaux où circule une information [1]. D'autres pistes de recherche se développent également en vue de proposer des algorithmes totalement décentralisés, par exemple pour la maximisation de la couverture des réseaux de capteurs [12].

Dans cette optique, pensons au cas concret suivant : on dissémine un ensemble de capteurs sur un grand territoire. Ceux-ci ont une durée de fonctionnement limitée du fait de l'épuisement de leur batterie. On souhaite pouvoir détecter de manière totalement décentralisée, c'est-à-dire par un consensus, le moment où une fraction donnée de ces capteurs ne fonctionne plus, en vue de savoir quand il sera nécessaire d'intervenir pour ajouter de nouveaux capteurs qui pourront les remplacer. Notre système de diagnostic vise à répondre à ce genre de problème.

Dans la partie 2, nous présentons le formalisme utilisé pour décrire nos modèles. Nous étudions alors le fonctionnement de ces modèles et évaluons leur adéquation au problème du diagnostic décentralisé (parties 3, 4 et 5). Enfin, dans la partie 6 nous établissons une synthèse rapide de nos observations et discutons des perspectives de ce travail.

2 Automates cellulaires

De manière informelle, on peut définir un automate cellulaire comme un système composé d'une grille d'automates tous identiques, les cellules, ayant les caractéristiques suivantes :

1. Chaque cellule possède un état choisi parmi un ensemble fini d'états ; cet ensemble est noté Q.
2. À chaque itération, une cellule met à jour son état en fonction de son état courant et de celui de son voisinage, à l'aide d'une règle de transition locale.
3. Toutes les cellules appliquent la même règle de transition locale à chaque itération.

Un automate cellulaire est un cas particulier de système dynamique discret. Nous nous restreignons ici au cas des grilles bidimensionnelles. L'espace des cellules est défini par $\mathcal{L} = \{1, \dots, n\} \times \{1, \dots, m\}$ pour une grille de taille (n, m) (avec $n, m \in \mathbb{N}^*$).

Une configuration est un élément de $\mathcal{C} = Q^{\mathcal{L}}$, qui représente l'état de toutes les cellules de la grille à un moment donné.

Le voisinage $V(c)$ d'une cellule $c \in \mathcal{L}$ est un sous-ensemble fini de la grille, qui représente l'ensemble des cellules dont l'état est

1. http://fiatlux.loria.fr/

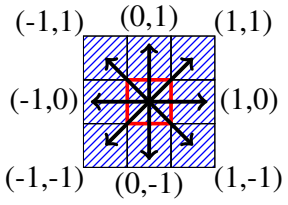


FIGURE 1 – Voisinage de Moore (cellule considérée entourée en rouge, voisines hachurées en bleu).

pris en compte par la règle de transition locale de l'automate cellulaire. Dans cet article, nous utiliserons le voisinage de *Moore*, avec la convention que la cellule centrale n'est pas incluse dans le voisinage (cf. figure 1) : on pose $V_M = \{-1, 0, 1\}^2 \setminus \{(0, 0)\}$ et le voisinage $V(c)$ de la cellule $c \in \mathcal{L}$ est défini par $V(c) = \{c + (i, j) : (i, j) \in V_M\} \cap \mathcal{L}$. Avec cette définition, les cellules du bord ont un voisinage de taille plus petite. Les fonctions locales que nous introduisons sont bien définies pour les cellules du bord, puisqu'elles dépendent seulement des nombres de voisines dans chaque état.

Règle de transition locale f

Dans cette étude, nous nous restreignons au cas où la règle de mise à jour des cellules est de la forme $f : Q \times \mathbb{N}^Q \longrightarrow Q$: la fonction de transition locale f prend en entrée l'*état courant de la cellule* considérée ainsi que le *nombre de cellules voisines se trouvant dans chacun des états de* Q. Les règles de cette forme sont dites *totalisantes-externes* (*outer-totalistic*).

L'application de cette règle de transition locale se fait en utilisant une fonction intermédiaire qui compte le nombre de cellules du voisinage qui sont dans un état donné :

$$\begin{array}{rccl} \eta : & \mathcal{C} \times \mathcal{L} & \longrightarrow & \mathbb{N}^Q \\ & (x, c) & \longmapsto & (|\{c' \in V(c); x_{c'} = q\}|)_{q \in Q}. \end{array}$$

Fonction de transition globale ϕ

Le système dynamique final est donc obtenu à l'aide de la fonction de transition globale $\phi : \mathcal{C} \longrightarrow \mathcal{C}$, qui associe une configuration de $\mathcal{C}$ à chaque configuration, après application de la règle de transition locale à toutes les cellules. Elle est définie par :

$$\begin{array}{rccl} \phi : & \mathcal{C} & \longrightarrow & \mathcal{C} \\ & x & \longmapsto & y \text{ tel que } \forall c \in \mathcal{L}, y_c = f(x_c, \eta(x, c)). \end{array}$$

Nous noterons x^0 la configuration initiale (correspondant au temps zéro), puis pour $t \geqslant 0$, $x^{t+1} = \phi(x^t)$.

3 Règle de *Greenberg-Hastings*

Notre point de départ est de chercher une règle locale simple dont la propriété principale soit de faire vivre une « onde » sur le réseau qui se fige si le nombre de cellules défaillantes est trop important (voir figure 3). De telles propriétés ont déjà été observées dans une version probabiliste de la règle de Greenberg-Hastings : ce modèle permet de modéliser simplement la propagation de vagues excitatrices à travers une grille de cellules (voir ref. [2] et les références ci-incluses).

L'ensemble des états est défini par $Q = \{\mathtt{N}, \mathtt{E}, \mathtt{R}\}$, où $\mathtt{N}$ représente l'état *neutre*, $\mathtt{E}$ l'état *excité*, et $\mathtt{R}$ l'état *réfractaire*. L'évolution est régie par une règle locale f qui prend en argument l'état de la cellule et le nombre $n_{\mathtt{E}}$ de cellules voisines dans l'état excité, et est définie de la manière suivante : une cellule neutre qui possède au moins un voisin dans l'état excité devient excitée avec probabilité p_τ, elle reste neutre sinon. Une cellule excitée devient toujours réfractaire et une cellule réfractaire devient toujours neutre. Cette règle est formalisée figure 2. Notons que les cellules défaillantes ne changent plus d'état et n'interviennent pas dans les transitions des autres états.

Il est connu que les propriétés de diffusion de cette onde dépendent fortement de la valeur de la probabilité de transmission p_τ : en dessous-d'un seuil critique, les ondes disparaissent brutalement [10, 2, 9]. C'est cette propriété dite de *transition de phase* qui nous permet ici de déclencher l'alerte : le changement brutal de comportement induit un changement qui nous permet de réaliser notre diagnostic décentralisé.

3.1 Mécanisme d'alerte

La problème central qui se pose à nous est donc de pouvoir distinguer, de manière locale, les deux phases du système, c'est-à-dire la présence ou l'absence d'une onde formée d'état excités sur la grille.

En utilisant l'automate cellulaire de Greenberg-Hastings, notre approche consiste à introduire un automate cellulaire ayant la propriété suivante : les cellules changent d'état d'autant moins souvent qu'il y a de cellules défaillantes. Plus précisément, s'il y a peu de cellules défaillantes, les cellules restent *actives*, ce qui signifie que leur état change fréquemment lorsque la règle locale s'applique. À l'inverse, la présence de cellules défaillantes dans une zone donnée de la grille

État $\mathtt{N}$ **:** **Si** $n_{\mathtt{E}} \geqslant 1$, *alors* $f(\mathtt{N}, n_{\mathtt{E}}) = \begin{cases} \mathtt{E} \text{ avec probabilité } p_\tau \\ \mathtt{N} \text{ avec probabilité } 1 - p_\tau \end{cases}$
si $n_{\mathtt{E}} = 0$, *alors* $f(\mathtt{N}, n_{\mathtt{E}}) = \mathtt{N}$
État $\mathtt{E}$ **:** $f(\mathtt{E}, n_{\mathtt{E}}) = \mathtt{R}$
État $\mathtt{R}$ **:** $f(\mathtt{R}, n_{\mathtt{E}}) = \mathtt{N}$

FIGURE 2 – Règle de Greenberg-Hastings

aura tendance à faire baisser l'activité des cellules. Nous utilisons donc un compteur d'inactivité, qui permet de faire en sorte qu'une cellule restée inactive durant un trop grand nombre de mises à jour se place dans un état spécifique d'alerte. Cet état se propage alors à son tour de proche en proche sur l'ensemble de la grille.

Cette méthode possède l'avantage de fonctionner en mode « tout ou rien » : l'état d'alerte est totalement absent de la grille tant qu'il n'a pas été produit en un lieu donné, puis il se propage en uniformisant les états des cellules.

Pour préciser les choses, notons $x_{i,j}^t$ l'état de la cellule $(i,j) \in \mathcal{L}$ au temps t. Les valeurs du compteur sont comprises entre 0 et une valeur maximale $T_{\max}$. À chaque pas de temps, lorsqu'une cellule change d'état, son compteur est remis à 0, tandis que si elle ne change pas d'état, son compteur augmente. Lorsque la valeur maximale $T_{\max}$ est dépassée, le compteur de la cellule prend un état de signalement spécial noté $\mathtt{S}$. Cet état $\mathtt{S}$ se maintient jusqu'à un éventuel changement d'état de $x_{i,j}^t$. Si une cellule a au moins quatre voisines dans l'état de signalement $\mathtt{S}$, elle passe dans l'état d'alerte $\mathtt{A}$, qui se propage ensuite sur la grille.

L'ajout de cet état intermédiaire de signalement $\mathtt{S}$ permet de s'assurer qu'il y a une zone contenant plusieurs cellules inactives contiguës avant de déclencher une alerte, ce qui permet d'être moins sensible à une anomalie localisée.

Considérons une cellule $(i,j) \in \mathcal{L}$, et notons $n_{\mathtt{A}}^t(i,j)$ et $n_{\mathtt{S}}^t(i,j)$ le nombre de cellules voisines de la cellule (i,j) qui sont au temps t dans l'état d'alerte $\mathtt{A}$ et de signalement $\mathtt{S}$, respectivement. L'état $\tau_{i,j}^{t+1}$ du compteur de la cellule (i,j) au temps $t+1$ est alors défini de manière formelle dans la figure 4.

3.2 Test du modèle

Nous commencerons par nous intéresser au cas de l'apparition progressive des défaillances au cours du temps, en mesurant l'évolution des différentes cellules. Ensuite, nous nous intéresserons au cas statique où les grilles sont initialisées avec un taux fixe de cellules défaillantes. Dans ces expériences, nous mesurerons la probabilité $p_{\mathtt{A}}$ qu'une grille initialisée avec une densité de cellules défaillantes donnée se place en état d'alerte après un certain nombre de pas de temps.

Dans un premier temps, nous nous plaçons donc dans le cas où les défaillances apparaissent de manière aléatoire uniforme : à chaque pas de temps, chaque cellule a une probabilité $p_{\text{def}} = 5.10^{-5}$ de devenir défaillante. La figure 5 présente l'évolution de la densité de cellules en état d'alerte et la densité de cellules défaillantes au cours du temps. Les densités des états d'alerte sont relatives au nombre de cellules non défaillantes.

On constate que la densité relative de cellules en état d'alerte passe brutalement de 0 à 1, du fait de la propagation de l'état d'alerte. Lorsqu'on reproduit l'expérience, on constate que la proportion de cellules défaillantes présentes au moment où ce saut se produit varie faiblement. Ce modèle répond donc au problème posé dans la mesure ou nous atteignons bien un état d'alerte qui se présente sous forme d'un consensus général sur l'état des cellules.

Examinons maintenant comment le seuil varie en fonction du paramètre du modèle p_τ. Pour cela, nous nous plaçons désormais dans un cas statique où les défaillances sont fixées une fois pour toutes. Initialement chaque cellule peut être défaillante avec probabilité $d_{\mathtt{D}}$, ou non défaillante auquel cas son état initial est tiré avec équiprobabilité parmi les états $\mathtt{E}$, $\mathtt{N}$, $\mathtt{R}$. Pour estimer la position du seuil en fonction du paramètre du modèle, nous effectuons, pour une valeur de $d_{\mathtt{D}}$ fixée, Z tests, dans lesquels nous faisons évoluer ces configurations initiales pendant T pas de temps. Nous notons ${p_{\mathtt{A}}}^T(d_{\mathtt{D}})$, la proportion des Z échantillons pour lesquels au bout de T pas de temps, la densité relative de cellules en état d'alerte dépasse 95%. Ce seuil peut être choisi arbitrairement proche de 100% mais nous avons souhaité éviter cette valeur limite de façon à ce

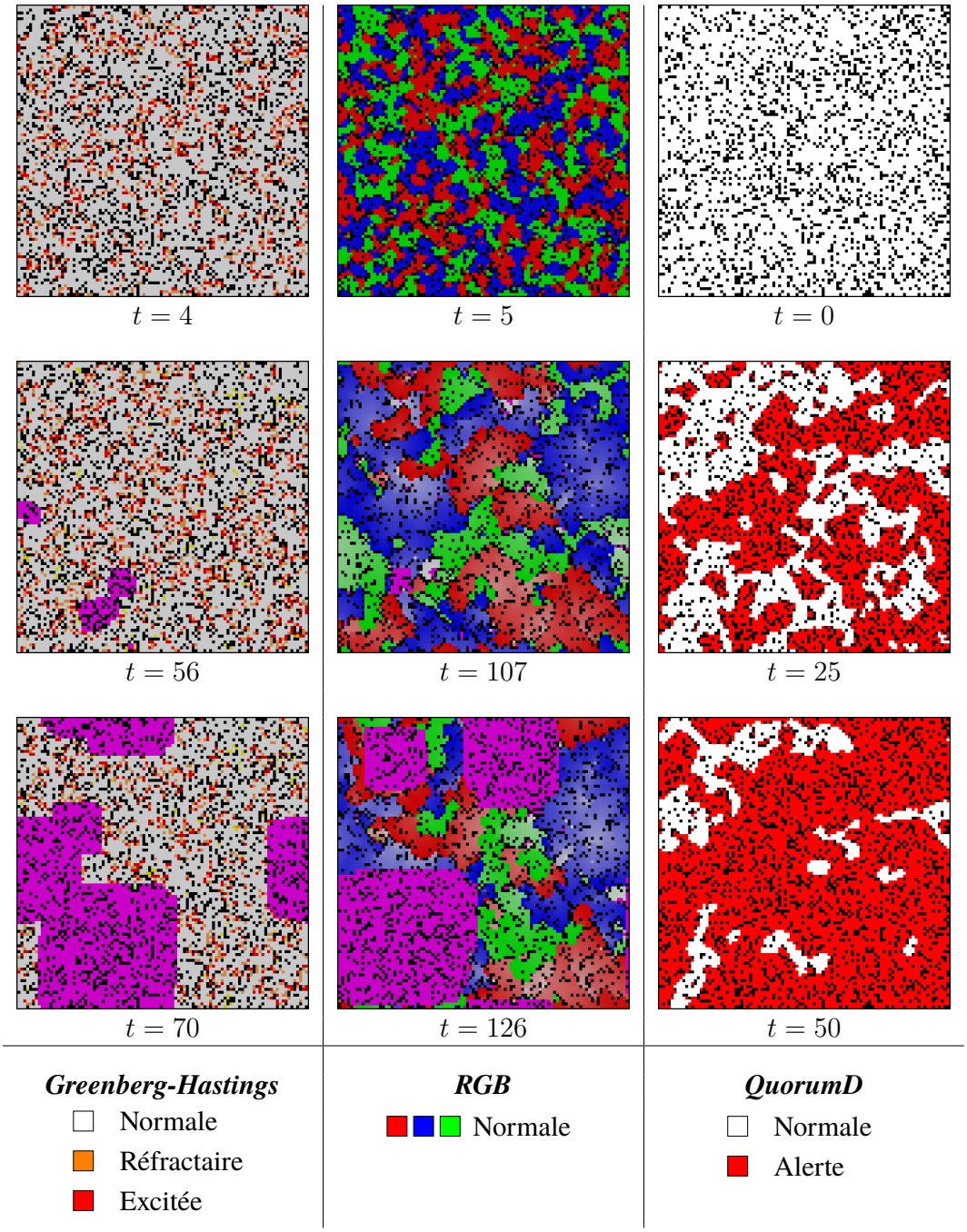


FIGURE 3 – Représentation de configurations typiques pour les trois modèles proposés. Les cellules défaillantes sont en noir. Sur les modèles Greenberg-Hastings et RGB, l'état d'alerte qui se propage est représenté en violet, l'état de signalement en jaune ; sur le modèle *QuorumD*, l'état d'alerte est en rouge.

que la présence de quelques cellules isolées restées à l'état normal ne perturbe pas la mesure.

Les résultats de l'expérience présentés figure 9 nous montrent que nous pouvons ajuster le seuil de cellules défaillantes détecté avec le paramètre p_τ.

3.3 Cas de l'asynchronisme

Nous avons supposé jusqu'ici que la règle locale s'appliquait de manière synchrone, c'est-à-dire, que toutes les cellules étaient mises à jour simultanément. Cependant, nous souhaitons que notre

Si $n^t_{\mathtt{A}}(i,j) \geq 1$ ou $\tau^t_{i,j} = \mathtt{A}$, **alors** $\tau^{t+1}_{i,j} = \mathtt{A}$ (propagation de l'état d'alerte),
sinon si $n^t_{\mathtt{S}}(i,j) \geq 4$, **alors** $\tau^{t+1}_{i,j} = \mathtt{A}$ (apparition de l'état d'alerte),
sinon si $x^{t+1}_{i,j} \neq x^t_{i,j}$, **alors** $\tau^{t+1}_{i,j} = 0$,
sinon si $x^{t+1}_{i,j} = x^t_{i,j}$, **alors** $\tau^{t+1}_{i,j} = \begin{cases} \tau^t_{i,j} + 1 & \text{si } \tau^t_{i,j} < T_{\max} \\ \mathtt{S} & \text{sinon.} \end{cases}$

FIGURE 4 – Règle de propagation de l'alerte

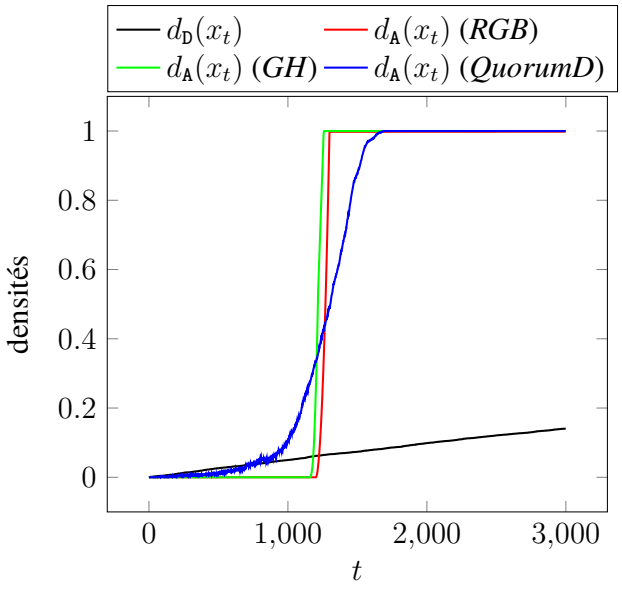


FIGURE 5 – Évolution des densités de cellules défaillantes et en alerte pour les modèles *RGB*, *Greenberg-Hastings* ($p_\tau = 0,31$) et *QuorumD* ($\lambda = 5$), sur des grilles de 100×100 cellules. Les paramètres des trois modèles ont été choisis pour aligner les seuils de cellules défaillantes détectés. Les trois modèles passent en état d'alerte après un certain nombre d'itérations.

méthode de diagnostic soit robuste à l'asynchronisme pour permettre au système de fonctionner en l'absence d'une horloge centralisée. En effet, dans le cas d'applications pratiques comme des réseaux de capteurs, il est préférable de ne pas supposer une synchronisation parfaite des transitions [3].

Notre prochaine expérience a donc pour but d'évaluer la robustesse du modèle à l'asynchronisme. Pour ce faire, nous appliquons une méthode de mise à jour dite α-asynchrone [6] : lors de chaque itération de la fonction de transition globale, chaque cellule est mise à jour avec une probabilité α, et laissée dans son état avec probabilité $1 - \alpha$.

La figure 10 reproduit l'expérience précédente (avec une valeur fixe des paramètres) en faisant varier le taux de synchronisme α. On constate que pour ce modèle, le seuil se déplace fortement. Par conséquent, si la façon dont les éléments sont mis à jour dans le système varie au cours du temps, il se produira aussi un déplacement de la valeur du seuil de détection. Ce modèle n'est donc pas robuste à l'asynchronisme. Nous allons donc chercher une règle robuste à l'asynchronisme.

4 Modèle *RGB*

Comme le modèle de *Greenberg-Hastings*, le modèle *RGB* que nous allons présenter a la propriété d'adopter deux comportements radicalement différents en fonction du taux de cellules défaillantes présentes sur la grille.

4.1 Présentation du modèle

Nous introduisons maintenant une nouvelle règle, également à trois états, mais pour laquelle la fonction de transition locale présente davantage de symétrie par rapport à ces trois états. Il s'agit d'un nouveau modèle, qui peut cependant rappeler la famille des automates cycliques, pour lesquels la règle locale fait changer les états des cellules en suivant un ordre pré-déterminé [7, 8].

La règle utilise trois états : $Q = \{R, G, B\}$. Soit q l'état courant d'une cellule, on note q^+ l'état suivant dans le cycle R-G-B (cf. figure 7), n_{q^+} le nombre de voisins dans cet état, et $n_{\mathtt{D}}$ le nombre de voisines défaillantes. La règle locale consiste à passer à l'état q^+ de manière certaine si on a au moins trois voisines dans l'état q^+, et avec une probabilité p_{tr} si on a seulement deux voisines dans l'état suivant. L'état courant est conservé dans les autres cas. La règle est donnée formellement dans la figure 6. À nouveau, les cellules défaillantes ne peuvent changer d'état et n'interviennent pas dans les transitions des autres états.

$$
\begin{array}{lrcl}
 & f : & Q \times \mathbb{N}^3 & \longrightarrow & Q \\
\textbf{Si} & n_{q^+} \geqslant 3, & f(q; n_R, n_G, n_B) & = & q^+ \\
\textbf{si} & n_{q^+} = 2, & f(q; n_R, n_G, n_B) & = & \begin{cases} q^+ \text{ avec probabilité } p_{tr} \\ q \text{ avec probabilité } 1 - p_{tr} \end{cases} \\
\textbf{si} & n_{q^+} \leqslant 1, & f(q; n_R, n_G, n_B) & = & q.
\end{array}
$$

FIGURE 6 – Règle *RGB*

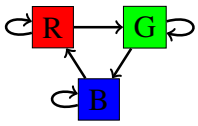


FIGURE 7 – Transitions possibles de l'automate RGB.

4.2 Expériences

De même que pour le modèle de *Greenberg-Hastings*, le modèle *RGB* voit son activité diminuer lorsque le nombre de cellules défaillantes augmente, ce qui permet de déclencher l'alerte au bout d'un certain temps (voir figure 5). Nous reprenons le même mécanisme que précédemment pour déclencher l'alerte, on peut donc à nouveau détecter un seuil de cellules défaillantes trop important. Ici, c'est p_{tr} qui va nous servir à régler le moment où l'alerte se déclenche. La figure 9 reprend le même protocole expérimental que pour *Greenberg-Hastings* (état initial équiprobable, même temps de mesure, même nombre d'échantillons) et donne des résultats similaires.

Qu'en est-il de la robustesse à l'asynchronisme ? Nous constatons sur les résultats présentés figure 10 que cette fois, le seuil détecté ne varie plus en fonction du taux de synchronisme α. Nous avons donc un net avantage à choisir cette règle dans un contexte où le mode de mise à jour des cellules n'est pas connu à l'avance.

4.3 Problème du passage à l'échelle

La contrepartie de ce fonctionnement en mode binaire est ce que nous appelons *le problème du passage à l'échelle* : si une petite partie de la grille reste inactive suffisamment longtemps pour déclencher l'état d'alerte, et ce même si le taux de cellules défaillantes à détecter n'est pas encore atteint, l'alerte sera déclenchée partout. Ceci peut se produire notamment dans le cas où des cellules défaillantes apparaissent très proches les unes des autres et coupent une zone de cellules du reste de la grille. La probabilité que ce cas de figure arrive est d'autant plus grande que la taille de la grille est grande, ce qui interdit donc un passage à l'échelle de la méthode. Nous proposerons donc d'autres solutions ne présentant pas ce problème.

5 Méthode *QuorumD*

Nous cherchons ici à trouver une solution dont les propriétés de déclenchement d'alerte soient indépendantes de la taille de la grille. Pour cela, nous abandonnons le mécanisme précédent de propagation d'alerte et nous considérons un état d'alerte qui peut apparaître et disparaître localement.

En dessous du seuil critique, l'état d'alerte est présent de manière partielle ; au-dessus du seuil critique, un *consensus global* s'établit et l'état d'alerte envahit totalement la grille.

5.1 Présentation du modèle

Idéalement, nous souhaitons obtenir une règle de transition possédant les caractéristiques suivantes :

- Une cellule dont toutes les voisines sont en état d'alerte ou défaillantes passe en état d'alerte (*c1*).
- Une cellule dont toutes les voisines sont en état normal passe à l'état normal (*c2*).
- Dans le cas contraire, la cellule passe en état d'alerte avec une probabilité d'autant plus grande que le nombre de cellules voisines en état d'alerte ou défaillantes est grand (*c3*).

Nous avons essayé différentes formes de fonctions pour définir la probabilité de passer à l'état d'alerte. Nous avons constaté que des règles reposant sur des combinaisons linéaires du nombre de cellules dans chaque état ne permettent pas d'obtenir de comportement satisfaisant. La règle que nous présentons ici est inspirée des modèles utilisés en physique statistique et en biologie. Par analogie avec le phénomène de détection de quorum (*quorum sensing*) présent dans les populations de bactéries, nous l'appelons *QuorumD*. L'obtention de l'état d'alerte est réglé par un paramètre λ qui permet de faire varier le taux de

$$
\begin{array}{lllll}
 & f : & Q \times \mathbb{N}^Q & \longrightarrow & Q \\
\textbf{Si } n_{\mathtt{N}} = 0, & & f(q; n_{\mathtt{A}}, n_{\mathtt{N}}, n_{\mathtt{D}}) & = & \mathtt{A} \qquad (c1) \\
\textbf{Si } n_{\mathtt{A}} = n_{\mathtt{D}} = 0, & & f(q; n_{\mathtt{A}}, n_{\mathtt{N}}, n_{\mathtt{D}}) & = & \mathtt{N} \qquad (c2) \\
\textbf{Sinon,} & & f(q; n_{\mathtt{A}}, n_{\mathtt{N}}, n_{\mathtt{D}}) & = & \begin{cases} \mathtt{N} \text{ avec probabilité } \frac{e^{\lambda \frac{n_{\mathtt{N}}}{S}}}{e^{\lambda \frac{n_{\mathtt{N}}}{S}} + e^{\lambda \frac{n_{\mathtt{A}}+n_{\mathtt{D}}}{S}}} \\ \mathtt{A} \text{ avec probabilité } \frac{e^{\lambda \frac{n_{\mathtt{A}}+n_{\mathtt{D}}}{S}}}{e^{\lambda \frac{n_{\mathtt{N}}}{S}} + e^{\lambda \frac{n_{\mathtt{A}}+n_{\mathtt{D}}}{S}}} \end{cases} \qquad (c3) \\
 & & & & \text{avec } S = n_{\mathtt{N}} + n_{\mathtt{A}} + n_{\mathtt{D}}.
\end{array}
$$

FIGURE 8 – Règle sans propagation (*QuorumD*).

cellules défaillantes à détecter. Une version formelle de cette règle est donnée sur la figure 8.

5.2 Test du modèle

La figure 5 présente l'évolution des densités de cellules défaillantes et de cellules en état d'alerte au cours du temps. Nous constatons que l'état d'alerte commence par coexister avec l'état normal en gardant une densité faible, puis lorsque le taux de cellules défaillantes devient trop grand, il prend le dessus pour envahir la grille. On peut donc parler ici d'un *comportement émergent*, au sens où l'état global d'alerte résulte ici d'une prise de décision véritablement décentralisée. Ceci rend la règle non seulement robuste à un changement de la taille de la grille, mais aussi à une concentration atypique de défauts à un endroit de la grille.

Nous souhaitons maintenant savoir dans quelle mesure nous pouvons fixer le seuil de cellules défaillantes à détecter en changeant la valeur du paramètre λ. La figure 9 nous montre que ce paramètre permet bien de régler le seuil de cellules défaillantes à détecter. De plus, le passage d'un état sans alerte à un état d'alerte se fait de manière plus précise en fonction du nombre de cellules défaillantes.

5.3 Résistance à l'asynchronisme

Comme pour les autres modèles, nous faisons varier le taux de synchronisme. La figure 10 présente la variation du déclenchement de l'alerte en fonction de ce taux : on observe une excellente robustesse du système à ces variations. Ceci est particulièrement encourageant dans la perspective d'utiliser ce modèle dans des situations concrètes.

5.4 Robustesse au changement d'échelle

Rappelons que nous cherchons à trouver une règle qui permette de détecter un seuil de défaillance fixé, même lorsque la taille du réseau varie. Par exemple, dans le cas des réseaux de capteurs, il arrive que de nouveaux capteurs soient ajoutés au réseau après le déploiement initial de celui-ci.

La figure 11 montre que les modèles de *Greenberg-Hastings* et *RGB* ont tendance à déclencher l'alerte plus tôt pour une taille de grille plus grande. En effet, plus la grille est grande et plus la probabilité qu'une zone ou les défaillances sont localement nombreuses provoque l'alerte en dessous du seuil est grande. En revanche, le modèle *QuorumD* a l'avantage d'avoir un comportement indépendant de la taille de la grille : les mesures de $p_{\mathtt{A}}$ ne varient pas lors du passage à l'échelle, ce qui montre que le passage dans l'état d'alerte généralisé résulte bien d'un consensus décentralisé.

6 Synthèse

Nous avons présenté une méthode de diagnostic décentralisé à l'aide d'automates cellulaires ayant des règles de transition simples et utilisant peu d'états. Ceci nous permet d'envisager des applications concrètes où cet automate cellulaire fonctionne en parallèle du système à diagnostiquer, tout en étant économe en quantité de calcul et en mémoire. Nous avons exploré deux approches différentes : la diffusion d'un état d'alerte (modèles *Greenberg-Hastings* et *RGB*), et l'obtention d'un consensus décentralisé (modèle *QuorumD*). Les trois modèles présentés semblent être des solutions adéquates au problème du diagnostic décentralisé, ayant chacune un comportement différent en réponse à différentes perturbations. Les modèles de *Greenberg-*

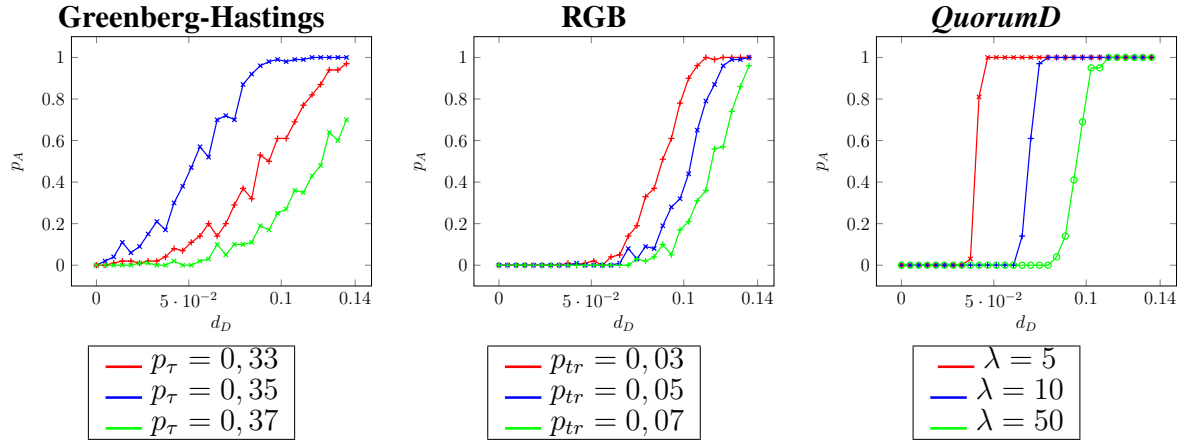


FIGURE 9 – Mesure de p_A pour différents paramètres de seuil d'alerte (mesures effectuées sur des grilles de 100×100 cellules sur une moyenne de $Z = 100$ tests).

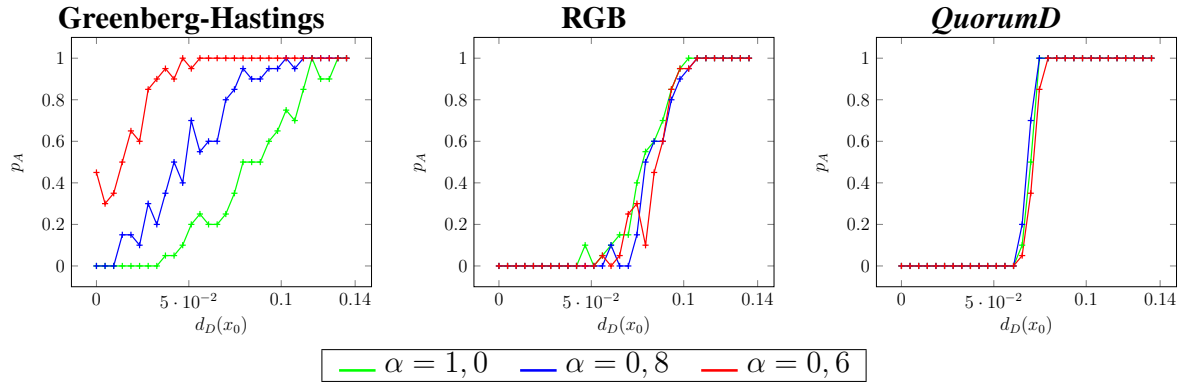


FIGURE 10 – Mesure de p_A pour différents taux de synchronisme (mesures effectuées sur des grilles de 100×100 cellules sur une moyenne sur 20 tests, $\lambda = 10, p_\tau = 0,35, p_{tr} = 0,02$).

Hastings et *RGB* ont l'avantage de ne pas faire apparaître d'état d'alerte en dessous du seuil à détecter, mais cette méthode interdit le passage à l'échelle direct (sans réajuster les paramètres). En revanche, même si le modèle *QuorumD* fait apparaître l'état d'alerte de manière plus progressive, il possède l'avantage d'être plus précis dans la détection du seuil de cellules défaillantes, d'être plus facile à paramétrer, et d'être plus robuste aux changements de mise à jour (asynchronisme). De plus, il est le seul des trois modèles à être robuste au changement d'échelle.

Le travail présenté d'ici est une première étude montrant la pertinence des automates cellulaires en vue de réaliser un diagnostic décentralisé. L'espace des règles étant gigantesque, notre travail d'exploration doit être poursuivi afin de mieux cerner les possibilités de solution au problème. Les expériences mériteraient également d'être approfondies, notamment en s'intéressant aux écart-types des valeurs obtenues.

Nous souhaitons désormais affiner notre compréhension des différents éléments entrant en compte dans la dynamique des automates cellulaires : l'influence du voisinage, du nombre d'états, de la forme des règles de transition, etc. En effet, il est difficile de prédire les différents comportements émergents observés et d'isoler les rapports entre les différents paramètres et le comportement global du système. Par ailleurs, d'autres types de défaillances sont envisageables et il serait intéressant de comparer l'effet de nos modèles d'auto-diagnostic distribué dans ces contextes.

Pour être en mesure d'appliquer ces méthodes sur le terrain, la prochaine étape sera d'examiner ces méthodes sur des topologies non régulières [5]. La forme des règles proposées ici a été choisie pour être aussi robuste que possible aux changements de topologie et des expériences préliminaires ont montré des résultats encourageants sur des topologies irrégulières [9]. En-

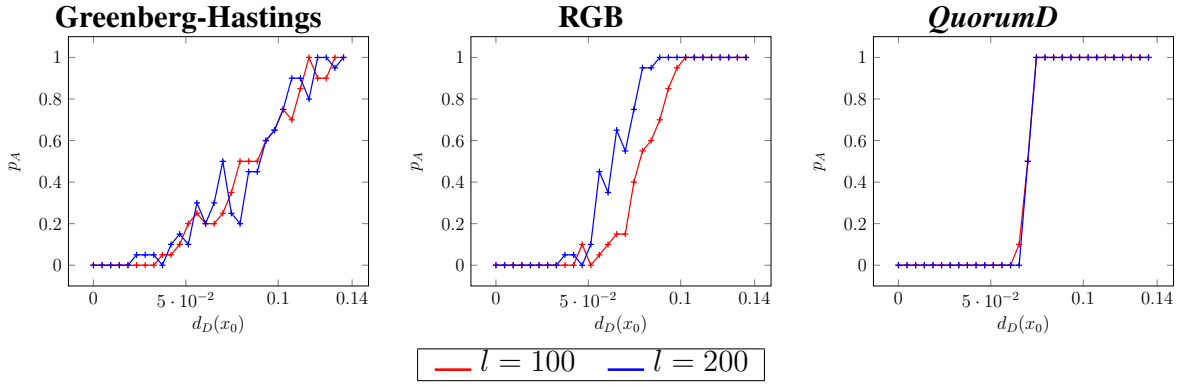


FIGURE 11 – Mesure de p_A pour différents taux de cellules défaillantes sur différentes tailles de grilles ($l \times l$) après 2000 itérations (moyenne sur 20 tests, $\lambda = 10, p_\tau = 0,35, p_{tr} = 0,02$).

fin, notons que les méthodes de transmission de l'information par automates cellulaires peuvent s'intégrer dans des systèmes multi-agents réactifs. On peut par exemple imaginer des situations où les agents tirent profit des informations de leur environnement pour localiser les zones où se trouvent le plus de défaillances et y engager des procédures de réparation.

Références

[1] Florence BÉNÉZIT, Patrick THIRAN et Martin VETTERLI : Interval consensus : From quantized gossip to voting. *In Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP'09*, pages 3661–3664, 2009.

[2] Hugues BERRY et Nazim FATÈS : Robustness of the critical behaviour in the stochastic Greenberg-Hastings cellular automatom model. *Int. Journ. of Unconventional Computing*, 7:65–85, 2011.

[3] Olivier BOURÉ : *Le simple est-il robuste ? une étude de la robustesse des systèmes complexes par les automates cellulaires*. Thèse de doctorat, Université de Lorraine, 2013. Université de Lorraine.

[4] Ana BUŠIĆ, Nazim FATÈS, Jean MAIRESSE et Irène MARCOVICI : Density classification on infinite lattices and trees. *Electron. J. Probab.*, 18:no. 51, 22, 2013.

[5] David W. CARMAN, Peter S. KRUUS et Brian J. MATT : Constraints and approaches for distributed sensor network security (final). *DARPA Project report,(Cryptographic Technologies Group, Trusted Information System, NAI Labs)*, 1(1), 2000.

[6] Nazim FATÈS : A guided tour of asynchronous cellular automata. *Journal of Cellular Automata*, 9(5-6):387–416, 2014.

[7] Robert FISCH : Cyclic cellular automata and related processes. *Physica D : Nonlinear Phenomena*, 45(1):19 – 25, 1990.

[8] Robert FISCH, Janko GRAVNER et David GRIFFEATH : *Cyclic Cellular Automata in Two Dimensions*, pages 171–185. Birkhäuser Boston, Boston, MA, 1991.

[9] Nicolas GAUVILLE : Système robuste de diagnostic décentralisé à l'aide d'automates cellulaires simples. Rapport technique, septembre 2018.

[10] Leonardo I. REYES : Greenberg-Hastings dynamics on a small-world network : the collective extinct-active transition. *arXiv preprint arXiv :1505.00182*, 2015.

[11] Antoine SPICHER, Nazim FATÈS et Olivier SIMONIN : Translating discrete multi-agents systems into cellular automata : Application to diffusion-limited aggregation. *In* Joaquim FILIPE, Ana FRED et Bernadette SHARP, éditeurs : *Proc. of ICAART 2009 : Agents and Artificial Intelligence*, pages 270–282. Springer Berlin Heidelberg, 2010.

[12] Antonina TRETYAKOVA, Franciszek SEREDYNSKI et Pascal BOUVRY : Graph cellular automata approach to the maximum lifetime coverage problem in wireless sensor networks. *SIMULATION*, 92(2):153–164, 2016.

De l'IoT à l'IoT-a : une approche pour des communications dynamiques

Alexandre Schmitt Valérie Renault Florent Carlier Pascal Leroux

Le Mans Université
Centre de Recherche en Éducation de Nantes (CREN)
Avenue O. Messiaen, 72085 Le Mans, France
prénom.nom@univ-lemans.fr

Résumé

Avec l'avènement des objets connectés et la mutation que vont connaître l'industrie et les particuliers, l'interopérabilité des communications de ces objets arrive au centre de nouvelles réflexions. Nous considérons ces objets comme des IoT-a (Internet of Things-agents) et proposons un ensemble de configurations permettant la mise en place d'un ou plusieurs agents au sein des niveaux matériels des objets. Nous concentrons notre étude sur la problématique de communications dynamiques des IoT-a dans le cas de la tolérance aux pannes. L'utilisation du protocole MQTT comme support des communications intra et inter-objets offre l'avantage de relier des objets connectés entre eux au moyen de passerelles appelées Bridges. Aussi, nous proposons une architecture SMA, reposant sur la plate-forme SMA embarquée Triskell3S. Elle permet la manipulation dynamique de Bridges dans l'objectif de minimiser la perte de messages dans le cas de la déconnexion d'un IoT-a du réseau. Enfin, nous présentons une mise en œuvre de notre architecture autour de deux expérimentations.

Mots-clés : *Objets connectés, IoT-a, Systèmes Multi-Agents Embarqués, Communications dynamiques*

Abstract

Due to the advent of IoT (Internet of Things) and the change that industry and individuals are about to know, the interoperability of communications among these objects is now at stake. We consider these objects as IoT-a (Internet of Things-agents) and we propose an ensemble of configurations allowing one or several agents to be among the material levels of the objects. The focus of our study is the problematic of dynamic communications among IoT-a in cases of fault tolerance. The advantage of using MQTT protocols as a communication support for intra and inter-objects is that IoT-a can be linked to each other through gateways called Bridges. Thus, we propose a MAS architeture, relying on the embedded MAS platform Triskell3S. It allows the Bridges to be dyanamically managed so as to minimise the loss of messages in case a IoT-a is disconnected from the network. Finally, we present an implementation of our achitecture with two experimentations.

Keywords: *Internet of Things-agent, Embedded multi-agents systems, Dynamic communication*

1 Introduction

Les objets connectés (Internet of Things : IoT) connaissent une évolution sans précédent depuis les années 1990. La volonté est d'agir sur le monde physique et/ou de collecter des données par l'utilisation d'un réseau global d'objets interconnectés. La question du lien constant de communication entre les objets se pose afin d'éviter les conflits opérationnels et de communication au sein de réseaux IoT. De plus, les limitations des ressources matérielles de certains objets entravent la gestion dynamique et intelligente d'un réseau d'IoT. Nous focalisons nos recherches sur la problématique des communications dynamiques entre IoT. Cela nous conduit à la question suivante : Comment assurer un lien constant de communication entre les objets ? Pour cela, nous avançons l'hypothèse que chaque IoT, disposant des ressources nécessaires, peut être une passerelle pour relayer les informations. Nous nous intéressons à rendre la gestion de ces passerelles dynamique afin d'apporter une meilleure tolérance aux pannes.

En parallèle, l'introduction des systèmes multi-agents (SMA) pour la gestion de processus industriels [11, 21] illustre leur adaptabilité à la gestion de systèmes réels complexes. Le concept d'IoT-a (Internet of Things-agent) [16] est une illustration des agents dans le domaine de l'IoT. Faisant suite à ces travaux nous nous focalisons dans cet article sur les échanges d'informations entre différents IoT-a. L'objectif est

de distribuer le traitement des informations dans les objets physiques et non de les déporter dans une infrastructure.

Dans cet article, nous commençons par positionner notre synthèse de l'architecture de l'IoT par rapport aux travaux effectués dans le domaine des Systèmes Embarqués et des Systèmes Cyber-Physique. Nous introduisons les différents paradigmes de l'IoT et les liens établis dans la littérature entre les IoT et le domaine des systèmes multi-agents. Nous présentons dans la section 3 les apports du modèle IoT-a avec ces différentes configurations possibles pour la mise en place d'agent(s) embarqué(s). En particulier, nous montrons comment les agents peuvent être repartis dans un système en fonction de leurs caractéristiques techniques. Nous formalisons dans la section 4 la problématique de communication dynamique des IoT-a et nous proposons une architecture SMA pour la résolution de cette problématique. Enfin, nous mettons en œuvre cette architecture dans la section 5, dans le cadre de deux expérimentations montrant les capacités d'adaptation aux tolérances aux pannes.

2 Positionnement

Travailler dans le domaine des systèmes embarqués nécessite une maîtrise des éléments matériels qui le composent et l'utilisation d'un environnement logiciel adapté aux ressources disponibles [10]. Les systèmes embarqués (SE) sont définis comme étant des systèmes de traitement de l'information intégrés dans des produits dédiés [10]. Ces solutions ont pour but de répondre de façon autonome (en calcul, en énergie et en mémoire) à un besoin spécifique. Lee [9] avance la notion de Cyber Physical Systems (CPS) et les définit comme étant un système embarqué auquel est ajouté la possibilité de capter des informations et/ou agir sur son environnement extérieur. Dans le domaine de l'industrie 4.0, dés lors que les CPS communiquent, nous pouvons les définir comme des IoT [13].

Dès la fin des années 1990, l'industrie prend la mesure des possibilités du domaine des Système Multi-Agents. Les réseaux électriques intelligents [5] ou encore la gestion de connectivité dans les plates-formes pétrolières [6] sont des champs d'application utilisant les SMA. Cette vision donne un système simple et tolérant aux pannes, en opposition aux systèmes centralisés complexes, puisqu'aucun des agents n'a besoin d'informations, a priori, sur les autres agents pour exécuter leurs taches primaires. Les CPS étant désormais capables de communiquer entre eux leurs résultats, ils deviennent des objets connectés répondant aux mêmes problématiques. L'approche multi-agents dans le domaine des IoT prend alors tout son sens. La mise en place d'agents dans les réseaux de capteurs [7] propose de réduire la consommation électrique des communications grâce à une distribution des agents. Cette vision est une première approche pour les communication dynamique entre objets autonomes. Les travaux [4, 16] avancent le concept d'IoT-a pour Internet of Things-agent. Les objets sont connectés et inter-agissent sur la base d'un langage commun. Les agents sont en capacités d'agir sur les éléments matériels et résoudre un problème collectivement [16] tel que la résolution collective d'un problème de taquin (N-puzzle) sur un mur d'écrans.

En termes de conception, la méthode de co-design DIAMOND [8] propose une solution pour la conception de systèmes embarqués intégrant des systèmes multi-agents. Cette approche permet de dimensionner au mieux les éléments matériels et logiciels nécessaires à un projet global. Le logiciel (les agents) et le matériel sont vu comme un tout modélisable et segmentable. À l'inverse, le concept d'IoT-a adapte les agents ou le SMA à des environnements matériels existants. Les agents sont conçus pour être compatibles avec différents matériels.

3 IoT-a et plate-forme SMA embarquée

La diversité des architectures présentes sur le marché des objets connectés nécessite un modèle pour l'intégration d'agents capables de s'adapter à différents niveaux matériels. Nous définissons le terme de *niveau matériel* comme étant une abstraction par rapport au jeu d'instructions du processeur du système. Plus un système est de haut niveau matériel, plus il s'affranchit des éléments électroniques qui le compose.

La mise en place d'agents au sein de différents niveaux matériels doit permettre de répondre à plusieurs types de besoins des systèmes (temps réel, événements bas niveaux, rapidité d'exécution, etc.). Aussi, deux systèmes, ayant l'un des contraintes temps réel et l'autre la nécessité d'une grande bande passante, n'auront pas les mêmes besoins en matière de mise en place d'agents. Nous proposons une évolution du modèle IoT-a pour répondre aux différentes archi-

tectures de l'IoT. Dans les travaux [4, 16], l'IoT-a a été présenté comme un concept intégrant la possibilité de faire interagir des agents au sein de systèmes embarqués. Les agents sont adaptés à un certain niveau matériel. Ce modèle ne détaille pas précisément l'emplacement du ou des agents au sein du matériel. Nous présentons dans la partie suivante les différentes configurations pour ce modèle.

3.1 Présentation du modèle IoT-a

Nous définissions le terme IoT-a comme l'ensemble des IoT-agents d'un réseau lorsque qu'il est utilisé au pluriel. Néanmoins un IoT-a est aussi considéré comme un seul objet-agent d'un réseau d'IoT-a. Nous proposons quatre configurations détaillées figure 1 pour la mise en place d'agents au sein d'un objet de l'IoT. Une fois agentifié cet objet est alors un IoT-a. Ces configurations peuvent être cumulées car elles sont indépendantes. Plus un objet est complexe, plus il peut intégrer de configurations d'agents différentes. A l'inverse, plus un objet est dédié à une tâche précise plus le nombre de configurations possibles est restreint. Un IoT-a disposant de plusieurs configurations est dans la possibilité d'échanger des informations entre différents composants matériels et logiciels.

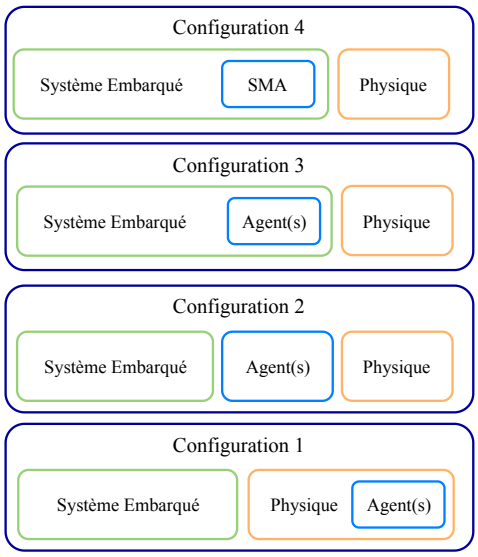


FIGURE 1 – Conception d'IoT-a

La **configuration 1** propose l'intégration d'un agent au niveau du matériel. Un objet dispose d'une architecture matérielle divisée en trois parties principales [12] : le processeur, la mémoire et les périphériques. L'ensemble est interconnecté au moyen d'un bus de données et d'un bus d'adresses. Un ou plusieurs périphériques agents peuvent être ajoutés au système et connectés sur les bus de données et d'adresses. Le logiciel du système embarqué peut bénéficier des comportements des périphériques agents situés dans la partie matérielle. Les agents sont équipés de leur propre bus de communication compatible avec une plate-forme SMA. Ils peuvent alors communiquer avec d'autres agents répartis sur d'autres niveaux matériels.

Cette première configuration est applicable à des objets connectés pouvant reposer sur une architecture matérielle de type micro-contrôleurs ou un ASIC (Microchip PIC, Atmel Atmega, Espressif Systems ESP32, etc.). Le composant intègre un agent matériel dans le silicium et devient une fonction supplémentaire et autonome. L'intégration d'un tel agent est aussi envisageable dans un composant programmable de type FPGA car l'agent peut être directement programmé dans le composant.

Dans la **configuration 2** un agent est présent en complément du système logiciel global (ex : système d'exploitation). A partir de cette configuration, il est admis que le système dispose de ressources matérielles suffisantes pour héberger un système d'exploitation. Le *Trusted Execution Environement* [17], disponible sur certaines architectures de processeurs, est une partie matérielle dédiée à l'exécution de solutions de sécurité. Cette zone est inaccessible sans une procédure de sécurité depuis le reste du système et permet d'analyser certains comportements du système ou de réaliser certaines tâches sécurisées. Un co-kernel [3] est un noyau léger ne s'occupant que de certaines parties bas-niveau spécifiques. Il laisse le noyau original s'occuper de tous les éléments habituels mais passe prioritaire lorsque certaines actions lui sont réservées. Un agent logiciel dans ces deux cas de figure serait très proche du matériel tout en offrant la possibilité de communiquer plus facilement avec le reste du système.

Cette seconde configuration peut être mise en place sur des objets reposant sur des architectures de type ARM, PowerPC, RISC-V ou x86 (Broadcom, Motorola, Intel, etc.). C'est la première configuration entièrement logicielle mais elle est au plus près du matériel et ne subit pas la latence d'un système d'exploitation. L'exécution des agents a la possibilité d'être temps réel et/ou sécurisée.

La **configuration 3** permet l'intégration d'un ou plusieurs agents au sein du système logiciel. Les

agents sont présents soit dans le noyau logiciel pour prendre en compte des événements bas niveau ou temps réel, soit dans l'espace utilisateur pour répondre à des problèmes plus complexes pouvant être multitâches. Ces agents, comme ceux cités dans les deux premières configurations sont autonomes et ont besoin d'une plate-forme multi-agents pour échanger avec d'autres agents présents sur des objets connectés différents.

Il est possible de mettre en place cette configuration sur tout type d'architectures compatibles avec un système d'exploitation (Raspberry Pi, Freescale Sabre SD, Intel UP, etc.). Le système d'exploitation peut être de différentes natures (Linux, RTOS, Windows, etc.). L'agent est compilé pour être compatible avec le système d'exploitation et subit les ralentissements induits par d'autres processus.

Enfin, la **configuration 4** propose que l'objet héberge une plate-forme multi-agents complète. Les agents présents dans l'objet peuvent interagir de façon autonome et peuvent être en grand nombre. La plate-forme sert de relais aux agents et/ou aux SMA repartis sur d'autres types de configuration matérielle ou sur d'autres objets connectés de son réseau. Cette configuration nécessite un objet avec des ressources matérielles suffisantes pour l'exécution de plusieurs agents et la prise en charge de différents protocoles de communication à différents niveaux matériels.

La configuration 4 nécessite les mêmes contraintes d'intégration que la configuration 3 mais demande un dimensionnement des équipements en lien avec la quantité d'agents à administrer.

3.2 Passerelles pour communications hétérogènes

La mise en place d'agents à différents niveaux matériels pose de nombreux problèmes de communication. Un agent présent à un bas niveau matériel ne dispose que de peu de solutions pour échanger des informations avec d'autres agents présents dans d'autres IoT-a. En effet, un agent bas niveau ne dispose que de certains types de bus de communication (I2C, SPI, UART, etc.) pour interagir avec le monde extérieur. Sans une adaptation logicielle, ces bus ne sont pas compatibles.

La mise en place d'une plate-forme SMA embarquée est la solution que nous proposons pour

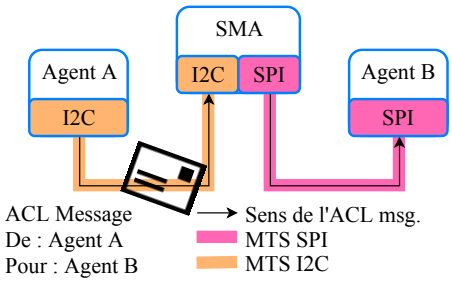


FIGURE 2 – Exemple de communications entre des agents bas niveaux

rendre compatible des agents bénéficiant de différents bus de communication. Le modèle FIPA [15] définit une plate-forme SMA comme un ensemble d'agents administratifs et d'actes de langage permettant à tous les agents d'interagir de façon cohérente. Chaque agent s'enregistre auprès de sa plate-forme SMA embarquée pour annoncer son existence et les services qu'il propose. La plate-forme est capable de relayer les informations entre différents niveaux matériels et bus de communication. Ce relais est effectué par la mise en place de plusieurs *Message Transport Services* (MTS). Un MTS permet à un agent de communiquer sur un bus spécifique. Un agent peut bénéficier d'un ou plusieurs MTS en fonction de son niveau matériel.

La figure 2 est un exemple de communication entre deux agents. Un agent A ne disposant que du bus I2C, utilise le MTS I2C pour interagir avec sa plate-forme. Il est capable d'envoyer un ACL message (Agent Communication Language, ACL) à sa plate-forme à destination d'un autre agent B utilisant le bus SPI. La plate-forme joue le rôle d'intergiciel pour relayer le message vers l'agent destinataire au travers du bus de communication adéquat. Étant donnée l'indépendance des configurations proposées, leur

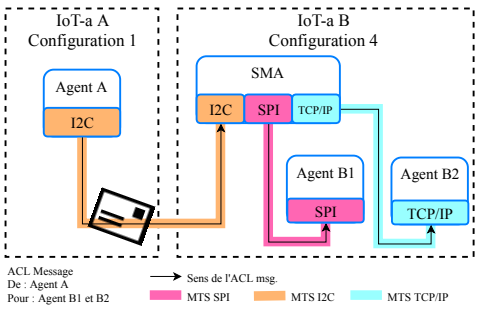


FIGURE 3 – Interaction entre deux IoT-a (A et B) avec différentes configurations 1 et 4

mise en place peut être effectuée sur des environnements différents. La figure 3 illustre un exemple de configuration pour plusieurs IoT-a. L'IoT-a A est compatible avec la configuration 1. L'IoT-a B est compatible avec la configuration 4. A se relie à B pour bénéficier d'une plate-forme SMA. Les IoT-a A et B peuvent alors échanger des ACL messages.

3.3 De l'IoT à l'IoT-a : intégration de Triskell3S

Dans le cadre de nos travaux, nous avons développé la plate-forme embarquée SMA Triskell3S [16, 14] adaptée aux IoT. Elle offre une interface de programmation permettant le développement d'agents sur différents environnements logiciel et matériel. Cette plate-forme est modélisée selon le standard FIPA [15]. Ainsi, chaque agent dérive d'une forme abstraite leur garantissant à tous de bénéficier des mêmes possibilités en matière de comportement, de communication et d'actes de langage. Dans le respect du modèle FIPA, des agents administratifs (Agent Management Service et Directory Facilitator) permettent l'enregistrement et le désenregistrement des agents disponibles et de leurs services associés.

Afin que les communications des agents mises en place avec les configurations 3 et 4 soient interopérables, la plate-forme Triskell3S utilisée pour notre expérimentation, dispose d'un protocole de communication commun entre tous les agents. Le protocole est compatible avec des intra-communications et extra-communications pour que tous les agents situés dans un IoT-a A puissent interagir avec les agents d'un IoT-a B.

L'ensemble des communications entre les agents compatibles avec les configurations 3 et 4 s'effectuent au travers d'un protocole de communication M2M (Machine to Machine) reposant sur le mécanisme *Publish/Subscribe*. Ce mécanisme permet aux réseaux d'IoT des échanges importants et légers sans les surcharger de requêtes. Parmi les différents protocoles M2M normalisés [2], le MQTT (Message Queuing Telemetry Transport) apparaît comme le plus adapté à notre modèle IoT-a [19]. Cette norme définit, entre autres, le *broker* comme étant le nœud central d'un réseau en étoile ou chaque client doit faire référence à son broker afin d'échanger avec le reste des clients. Chaque agent est alors un client de ce réseau. Le broker porte des *topics*, organisés en une arborescence à l'image d'une hiérarchie de dossiers. Un agent peut déposer une information sur un topic (mode *publisher*), ou écouter un ou plusieurs topics simultanément pour bénéficier instantanément des informations (mode *subscriber*). Un agent peut être à la fois *publisher* et *subscriber*. Ce mécanisme permet des échanges synchrones et asynchrones.

L'utilisation d'un moteur de communication MQTT offre de nombreux avantages pour la communication des agents au sein d'une seule plate-forme.

Cependant, comment faire communiquer des agents entre plusieurs plate-formes SMA ? Une solution consiste à ce que chaque agent bénéficie de plusieurs canaux MTS MQTT pour communiquer avec différents brokers. Cette solution est contraignante car elle nécessite de nombreuses configurations au sein de chaque agent. Nous proposons, dans la partie suivante, un mécanisme dédié à la communication entre brokers que nous adaptons pour le rendre dynamique.

4 Liens de communication dynamiques entre IoT-a

Nous nous focalisons dans cette section sur la communication entre plusieurs IoT-a en configuration 4. C'est à ce niveau de configuration que les IoT-a hébergent une plate-forme SMA. Un IoT-a ayant la capacité d'être en configuration 4 doit bénéficier d'un broker MQTT comme support pour les communications entre ses agents locaux. Aussi, chaque agent dispose d'un canal de communication MTS MQTT pointant vers un broker local. Cette solution permet à tous les agents de bénéficier de la même configuration de communication. Nous souhaitons disposer d'un mécanisme permettant la communication entre différentes plate-formes SMA tout en respectant ces contraintes de configurations.

4.1 Liens entre plate-formes

De nombreux projets [2] permettent la génération de passerelles entre différents brokers. Certains proposent l'échange de tout ou partie de leur arborescence de topics [1] ; cette fonctionnalité est alors qualifiée de *bridge*. Elle permet de créer un "miroir" d'informations entre plusieurs brokers.

La figure 4 illustre deux plate-formes Triskell3S reliées entre elles par un bridge. Une plate-forme est alors dans la capacité de se relier à

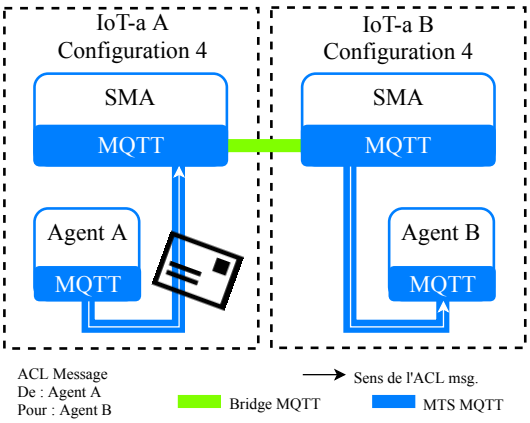


FIGURE 4 – Liens MQTT entre plusieurs IoT-a en configuration 4

différentes plate-formes et échanger des informations sur tout ou partie du réseau d'IoT-a.

Disposer de liens directs entre IoT-a, par l'intermédiaire des bridges, permet de nombreuses possibilités de gestion de l'information, comme l'organisation hiérarchique d'agents en fonction de leur emplacement physique, de leurs natures ou de leurs criticités. Ils permettent de mettre en place une redondance ou de disperser l'information sur différents sites géographiques. Cependant, pour que les plate-formes SMA puissent répondre aux critères de tolérance aux pannes, de dynamisme et d'autonomie, les bridges doivent pouvoir évoluer en fonction de différents indicateurs. Nous constatons que les différents projets de broker MQTT open source disponibles, proposent principalement l'utilisation d'un fichier de configuration où l'ensemble des options nécessaires à la mise en place de bridges y sont définies. Cette solution va à l'encontre de nos critères de dynamisme et d'autonomie car elle contraint la mise en place des bridges et la rend statique. Nous proposons une nouvelle approche ne modifiant pas le comportement de l'analyse des configurations initiales.

4.2 Bridges dynamiques

Par exemple, le redémarrage d'un broker MQTT induit de nombreuses pertes de configuration et de messages. Nous souhaitons disposer d'une solution de communication capable de s'adapter à son environnement en minimisant la perte de messages lors de redémarrage, pannes, coupure de réseau, etc.

La norme MQTT définit succinctement l'utilisation de topics réservés pour les besoins du broker appelé : $ topic. Ces topics permettent de mettre à disposition des informations concernant le serveur (le nombre de connexions actives, la charge réseau, le nombre de paquets transmis, etc.). Ces topics ont la possibilité d'interagir avec le broker et non pas seulement en récupérer de l'information. Ces topics permettent de modifier certaines configurations du serveur pour élaborer de nouveaux scénarios d'échange d'informations sans avoir à redémarrer le service.

Issu d'un travail collaboratif (Open Source) entre des chercheurs et des industriels, le projet Mosquitto [1] est adapté aux systèmes embarqués pour les interactions et actes de communication au sein de réseaux d'objets connectés. Mosquitto utilise le principe des topics réservés par le pattern $SYS. Le broker recense plus de 30 topics renvoyant périodiquement des informations. Il propose aussi la mise en place de bridges entre différents brokers via une configuration au démarrage du service. Nous avons ajouté au projet Mosquitto, la possibilité d'interpréter deux nouveaux topics pour la création et la suppression de bridges :

— $SYS/broker/bridge/new, topic pour la création.
— $SYS/broker/bridge/del, topic pour la suppression.

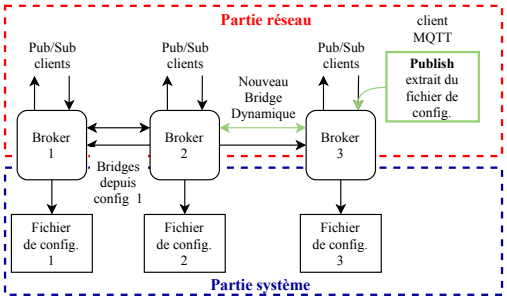


FIGURE 5 – Procédure de création d'un bridge dynamique

Notre principe s'appuie sur l'utilisation des paramètres disponibles pour la configuration de Mosquitto. Nous réutilisons le contenu nécessaire pour un fichier de configuration. Comme l'illustre la figure 5, la publication d'un message sous forme de texte sur les topics cités précédemment permet d'interagir avec le broker. Si le message est valide, alors les informations sont traitées et rappellent les mêmes mécanismes qu'au démarrage pour la création d'un bridge. Cette solution ne nécessite aucun accès au système et peut être animée par n'importe quel client MQTT.

Ce point d'entrée pour la modification de liens entre les plate-formes permet à des agents de modifier la topologie de liens des brokers avec la publication de messages aux topics adéquat [18]. Nous appelons cette fonctionnalité : *bridge dynamique*.

4.3 Réorganisation dynamique de bridges par une approche multi-agents

Nous venons de voir qu'il est possible grâce à la méthode de bridge dynamique de faire évoluer les liens entre plate-formes SMA à la volée sans redémarrage du service de communication ou intervention de la part d'un administrateur. L'administration de ces bridges, grâce à un ensemble d'agents présents dans chaque IoT-a, permet de prendre en compte la déconnexion et la connexion d'un nouvel IoT-a en configuration 4 sur le réseau. Les agents d'une plate-forme sont autonomes pour déterminer les liens qu'ils doivent établir pour minimiser la perte de messages depuis et vers les autres plate-formes présentes sur d'autres IoT-a. Aussi, nous proposons les trois familles d'agents suivantes pour la réorganisation dynamique de bridges MQTT : les agents Ping, Bridge et Saver.

L'**agent Bridge** est en capacité de créer et supprimer des bridges depuis le broker local vers un ou plusieurs brokers distants. Il a une connaissance limitée de son environnement. L'agent Bridge ne connaît que les bridges actifs entre son IoT-a et les IoT-a distants. Il n'a pas connaissance de l'état de connexion des IoT-a auxquels il est relié mais il est en attente d'un ordre de création de bridge depuis un autre agent. Pour chaque création d'un nouveau bridge, l'agent supprime le bridge précédemment ordonné. Nous avons observé que cette manipulation accélère la gestion des bridges car elle supprime un effet de bord appelé bridges fantômes [18].

L'**agent Ping** dispose de deux comportements. Le premier est d'interroger par requêtes réseau de type *Internet Control Message Protocol*. les IoT-a du réseau selon une stratégie de ping que nous détaillerons par la suite. Une stratégie de ping définit quel IoT-a devra être interrogé en fonction de l'adresse IP de l'IoT-a sur lequel est présent l'agent. Le second comportement est de pouvoir générer des stimuli dédiés à l'agent Bridge. Selon les stimuli générés, l'agent Bridge prend la décision de la création ou de la suppression d'un bridge avec un IoT-a. Ces stimuli se traduisent par la publication d'un message ACL de la part de l'agent Ping sur un topic écouté par l'agent Bridge.

Enfin, l'**agent Saver** est là pour sauvegarder les messages de sa plate-forme subissant la déconnexion des IoT-a auxquels il peut être connecté. Lorsque l'agent Ping détecte une déconnexion, il publie auprès de la plate-forme qu'elle est déconnectée du réseau. L'agent Saver prend alors le relais des communications non transmises durant une déconnexion. Lorsque les agents Ping et Bridge ont fini de coopérer pour trouver une solution à la re-connexion de leur plate-forme au réseau, l'agent Saver publie tous les messages publiés hors connexion. Actuellement, seules les communications entre les agents administratifs sont sauvegardées.

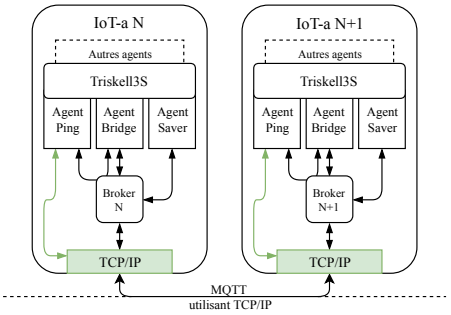


FIGURE 6 – Architecture multi-agents d'un IoT-a

La figure 6 détaille l'architecture des agents au sein de chaque IoT-a. Cette architecture est compatible avec plusieurs supports de communication physique proposant un réseau TCP/IP. Les agents dédiés à la réorganisation dynamique des communications font partie de la plate-forme Triskell3S. Ils s'enregistrent et publient les services qu'ils proposent auprès des agents administratifs.

Comme avancée dans la présentation du modèle IoT-a, d'autres agents (*Autres agents* sur la figure 6) peuvent être présents à différents niveaux matériels en plus des agents Bridge, Ping, Saver, AMS et DF. Nous avons implémentés d'autres agents pour la gestion dynamique d'informations et/ou le traitement distribué de tâches (agents haut niveaux) et la surveillances des consommations mémoire, CPU, bande passante et électrique (agents bas niveaux) de chacun de nos objets. Aussi, certains IoT-a peuvent être déconnectés ou reconnectés, les agents Ping et Bridge jouent leurs rôles pour assurer la meilleure inter-connexion dynamique de l'ensemble. Les autres agents adaptent alors leurs traitements en fonction du réseau d'IoT-a connectés.

5 Expérimentation

Notre expérimentation a pour but de montrer la capacité à la tolérance aux pannes de notre architecture SMA dans le cadre de la déconnexion d'IoT-a. Les résultats exposés sont issu de la mesure la plus représentative parmi un ensemble d'essais. Une différence de 5% des temps de réponse des agents apparaît en fonction des essais. Chaque mesure met en situation 16 IoT-a connectés par réseau TCP/IP en Ethernet. Au démarrage, les IoT-a ne sont pas reliés par bridge et ne peuvent échanger aucune information. Seul l'agent Ping a la capacité de découvrir les éléments du réseau et il notifie l'agent Bridge des liens devant être mis en place.

5.1 Protocole expérimental

Plusieurs topologies de liens sont possibles pour relier des IoT-a. L'étude des différentes topologies réseau pour la liaison d'éléments est très présente dans la littérature [20]. La technologie de bridges MQTT ne supporte pas pour le moment des topologies redondantes, comme par exemple Ring, Mesh or Spidergon [20]. C'est la raison pour laquelle nous proposons d'étudier deux topologies en accord avec les contraintes du bridge. Les agents Ping sont responsables de la mise en place de ces topologies. Pour nos expérimentations, les agents Ping ne peuvent disposer que d'un seul type de stratégie à la fois.

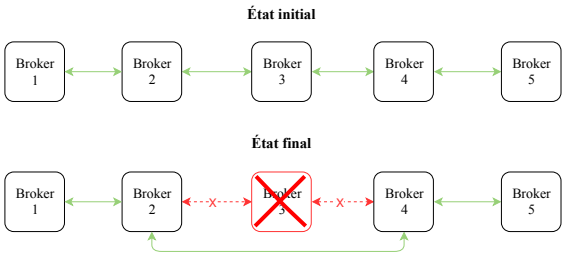


FIGURE 7 – Topologie en ligne

La stratégie en **ligne** illustrée par la figure 7, privilégie une meilleure répartition de la bande passante sur chaque élément du réseau. Cependant, elle est plus sensible en cas de déconnexion d'un des éléments.

La stratégie en **étoile** illustrée par la figure 8, privilégie une meilleure résistance à la déconnexion d'un élément. Néanmoins, elle supporte mal la déconnexion de son nœud principal. De plus, elle requiert une bande passante plus importante pour l'élément étant le nœud du réseau.

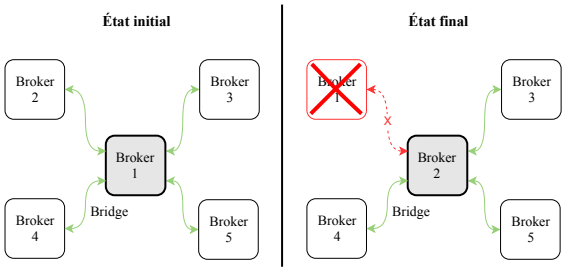


FIGURE 8 – Topologie en étoile

Les figures 7 et 8 détaillent les procédures suivies lors des deux expérimentations non synchronisées. Dans le cas de l'expérimentation de la stratégie en ligne, à l'état initial, chaque IoT-a essaie de créer un bridge avec son voisin inférieur le plus proche. Par exemple : broker 3 se connecte au broker 2. En situation finale l'IoT-a 3 est déconnecté et l'IoT-a 4 trouve une solution pour se connecter à un autre IoT-a.

L'expérimentation pour la stratégie en étoile débute lorsque chaque IoT-a essaie de créer un bridge avec l'IoT-a ayant l'adresse la plus faible. Les brokers 2, 3, 4 et 5 se connectent à 1.

En situation finale l'IoT-a 1, étant le nœud principal, est déconnecté, les IoT-a 3, 4 et 5 trouvent alors un nouveau nœud et se connectent à l'IoT-a 2.

5.2 Résultats

Deux mesures ont été effectuées pour caractériser chaque stratégie de ping disponible dans les agents Ping. Afin de simplifier les résultats, nous ne faisons apparaître que les résultats significatifs des IoT-a concernés par la déconnexion et la re-connexion. Au début de chaque mesure, aucun IoT-a n'avait connaissance de l'état de ses voisins et aucune configuration ne leur permettait d'établir un bridge.

Les figures 9 et 10 détaillent les résultats obtenus pour la stratégie en ligne puis en étoile. Nous constatons trois temps importants : la connexion de l'ensemble de IoT-a sur leur cible déterminée par leur stratégie (marqueur C), la déconnexion de l'IoT-a central (marqueur D) et sa re-connexion (marqueur R). Les agents Ping et Bridge réagissent en $\approx$100ms pour la création de bridges.

Dans le cas de la stratégie en ligne, on constate que l'agent Ping présent sur l'IoT-a 4 détecte la déconnexion de l'IoT-a 3 et propose instantanément à l'agent Bridge une solution de connexion

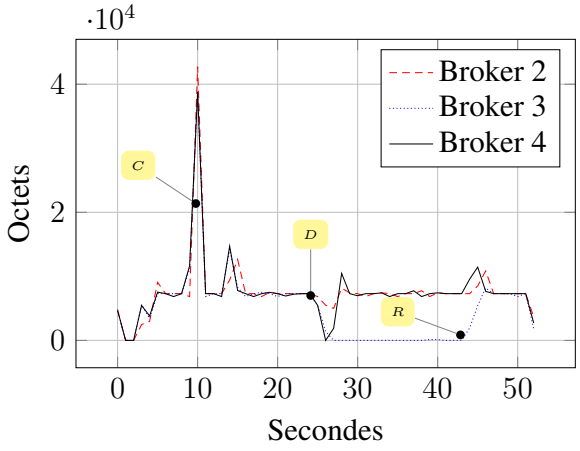


FIGURE 9 – Nombre d'octets reçus par broker pour l'expérimentation en ligne

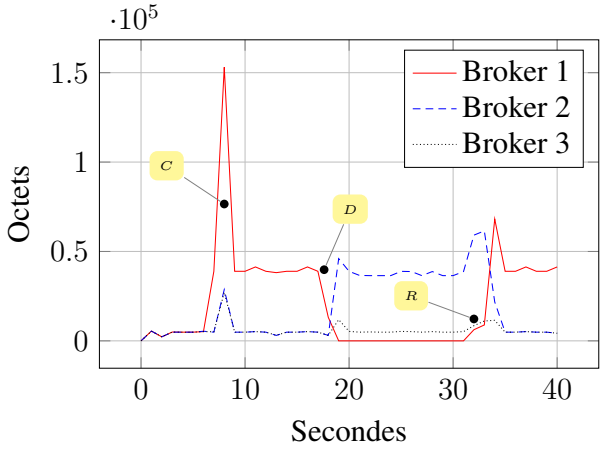


FIGURE 10 – Nombre d'octets reçus par broker pour l'expérimentation en étoile

à l'IoT-a 2. Il est possible de voir pour la stratégie en étoile, qu'au moment de la déconnexion de l'IoT-a 1, les agents Ping présents sur les IoT-a 3, 4 et 5 proposent un nouveau nœud : IoT-a 2. Dans les deux types de stratégie, la reprise des communications s'effectue en moins d'une seconde. Enfin, le recouvrement des courbes aux marqueurs R montre que la reconnexion ne provoque aucune perte de message dans les deux types de stratégies.

Les résultats que nous obtenons confirment le dynamisme des agents Bridge, Ping et Saver pour maximiser la tolérance aux pannes. Chacun de nos IoT-a s'adapte aux autres IoT-a présents sur le réseau. Bien que responsable d'un faible surcoût des communications, le coût en bande passante de création dynamique de bridges reste négligeable par rapport au reste des communications. Nous notons d'ailleurs la grande réactivité de notre solution qui réorganise l'ensemble du maillage du réseau en moins d'une seconde pour chaque événement. Enfin, notre approche SMA montre que les IoT-a déterminent de façon autonome comment se relier aux autres éléments de leur réseau sans configuration initiale préalable.

6 Conclusion et perspectives

Nous nous sommes intéressés dans cet article à l'intégration des agents et SMA au plus proche du matériel, formalisant les bases d'un modèle IoT-a compatible avec différentes architectures existantes sur le marché des IoT. Nous avons concentrés notre étude sur un niveau de configuration suffisant pour la prise en compte de la problématique des communications entre IoT-a hébergeant une plate-forme SMA.

La formalisation d'une architecture IoT-a offrant la possibilité de mise en place de bridges dynamiques et reposant sur la plate-forme Triskell3S permet de relier des plate-formes SMA de manière dynamique et autonome. Notre expérimentation a permis de montrer la capacité de notre architecture à s'adapter à deux cas particuliers de stratégies de topologie. Elle ouvre la possibilité d'implémentation à des topologies plus complexes.

Nous avons détaillé deux stratégies pour nos agents Ping : ligne et étoile. Seule l'étude des résultats de chaque stratégie indépendante est effectuée dans ce papier. Nous envisageons dans de prochains travaux, que nos agents Ping bénéficient des deux stratégies et puissent définir de façon autonome le choix de topologie en fonction des ressources mesurées par nos agents de surveillances bas niveaux et de l'architecture des IoT-a du réseau. Les IoT-a auraient alors la possibilité de créer des topologies hybrides mélangeant les stratégies ligne et étoile en toute autonomie.

Références

[1] R. A Light. Mosquitto : server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, 2(13), may 2017.

[2] A. Al-Fuqaha, K. Aledhari, G. Mohsen, R. Ammar, and M. Mehdi. Toward better horizontal integration among iot services. *IEEE Communications Magazine*, 53(9) :72–79, September 2015.

[3] A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio. Performance comparison of vxworks, linux, rtai and xenomai in a hard real-time application. *Nuclear Science, IEEE Transactions on*, 55 :435–439, 03 2008.

[4] F. Carlier and V. Renault. Iot-a, embedded agents for smart internet of things : Application on a display wall. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence, The First International Workshop on the Internet of Agents (IoA)*, pages 80–83, Omaha, Nebraska, USA, 10/2016 2016. IEEE Computer Society.

[5] S. Chouhan, J. Ghorbani, H. Inan, A. Feliachi, and M. A. Choudhry. Smart mas restoration for distribution system with microgrids. In *2013 IEEE Power Energy Society General Meeting*, pages 1–5, July 2013.

[6] L. Ioniţă and I. Ioniţă. Nm-mas : A multiagent system for network management in oil industry. In *2014 RoEduNet Conference 13th Edition : Networking in Education and Research Joint Event RENAM 8th Conference*, pages 1–6, Sept 2014.

[7] J.-P. Jamont and M. Occello. Une approche multi-agents pour la gestion de la communication dans les réseaux de capteurs sans fil. *Revue des Sciences et Technologies de l'Information - Série TSI : Technique et Science Informatiques*, 25(5) :661–690, 2006.

[8] J.-P. Jamont and M. Occello. Meeting the challenges of decentralized embedded applications using multi-agent systems. *International Journal of Agent Oriented Software Engineering*, 5(1) :22–67, 2015.

[9] E. A. Lee. Computing foundations and practice for cyber-physical systems : A preliminary report. Technical Report UCB/EECS-2007-72, EECS Department, University of California, Berkeley, May 2007.

[10] P. Marwedel. *Embedded System Design : Embedded Systems Foundations of Cyber-Physical Systems and the Internet of Things*. Springer Publishing Company, Incorporated, 3rd edition, 2018.

[11] V. Mařík and J. Lažanský. Industrial applications of agent technologies. *Control Engineering Practice*, 15(11) :1364 – 1380, 2007. Special Issue on Manufacturing Plant Control : Challenges and Issues.

[12] J. Nurmi. *Processor design : System-on-chip computing for ASIDs and FPGAs*. Springer, 2007.

[13] L. Pazzi and M. Pellicciari. From the internet of things to cyber-physical systems : The holonic perspective. *Procedia Manufacturing*, 11 :989 – 995, 2017. 27th International Conference on Flexible Automation and Intelligent Manufacturing, FAIM2017, 27-30 June 2017, Modena, Italy.

[14] P. Pico-Valencia and J. Holgado-Terriza. Agentification of the internet of things : A systematic literature review. *International Journal of Distributed Sensor Networks*, 14 :155014771880594, 10 2018.

[15] S. Poslad. Specifying protocols for multi-agent systems interaction. *ACM Trans. Auton. Adapt. Syst.*, 2(4), nov 2007.

[16] V. Renault and F. Carlier. Triskell3S, une plateforme embarquée multi-agents pour les IoT-a. In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2016)*, pages 181–190, Rouen, Saint-Martin-du-Vivier, France, Oct 2016. Fabien Michel and Julien Saunier.

[17] M. Sabt, M. Achemlal, and A. Bouabdallah. Trusted execution environment : What it is, and what it is not. In *2015 IEEE Trustcom/BigDataSE/ISPA*, volume 1, pages 57–64, Aug 2015.

[18] A. Schmitt, F. Carlier, and V. Renault. Dynamic bridge generation for iot data exchange via the mqtt protocol. *Procedia Computer Science*, 130 :90 – 97, 2018. The 9th International Conference on Ambient Systems, Networks and Technologies (ANT 2018).

[19] A. Schmitt, F. Carlier, V. Renault, and P. Leroux. Communication multi-niveaux pour des IoT-a. Interactions autour d'un mur d'écrans connectés. In *Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA 2017)*, Caen, France, Jul 2017.

[20] T. Takabatake. Simulations of noc topologies for generalized hierarchical completely-connected networks. In *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–5, June 2011.

[21] R. Unland. Chapter 2 - industrial agents. In P. Leitão and S. Karnouskos, editors, *Industrial Agents*, pages 23 – 44. Morgan Kaufmann, Boston, 2015.

Simulation multi-agent de l'autoconsommation collective en relation avec l'activité des foyers

J. Albouys-Perrois[a,b,c] albouys@limsi.fr
N. Sabouret[b] nicolas.sabouret@limsi.fr
Y. Haradji[c] yvon.haradji@edf.fr
M. Schumann[c] mathieu.schumann@edf.fr
C. Inard[a] christian.inard@univ-lr.fr

[a]Laboratoire des sciences de l'ingénieur pour l'environnement, Université de La Rochelle, France

[b]Laboratoire d'informatique pour la mécanique et les sciences de l'ingénieur CNRS, Université Paris Sud, France

[c]EDF R&D, France

Résumé

Cet article présente un modèle de simulation multi-agent de l'autoconsommation collective de l'énergie. Le modèle associe une simulation multi-agents de l'activité humaine, une simulation thermique du bâtiment et de l'eau chaude sanitaire (ECS), et une simulation de production locale d'énergie électrique photovoltaïque. Nous étudions différentes situations de consommation collective de l'énergie et nous montrons comment l'utilisation de cette énergie peut être optimisée à l'échelle du quartier en prenant en compte l'activité des occupants et les échanges d'énergie entre les foyers.
Cette article est une traduction et une réécriture d'un article soumis à la conférence Building Simulation 2019.

Mots-clés : *Simulation Multi-Agent, Consommation énergétique, Simulation de l'activité humaine*

Abstract

We present a framework and a model for the simulation of collective energy self-consumption strongly coupled to occupant activity in households. The model combines multi-agent simulation of human activity, a dynamic model of building energy including Domestic Hot Water (DHW) and all electrical appliances, and local generation of photovoltaic energy. A cooperative energy exchange policy between households was defined in order to maximise the use of locally generated PV electricity at the neighbourhood scale. In this first implementation, three households A, B and C were simulated, one being the PV energy producer. The DHW model was modified to benefit from PV electricity. Thanks to the energy exchange policy, a total of 83% of the energy generated by household A is used locally in the neighbourhood (compared to 38% when there are no energy exchanges with household B and C). The next step of this study is to add a model of energy storage in several self-consumption configurations.

Keywords: *Agent-Based Simulation, Energy Consumption, Human Activity Simulation*

1 Introduction

L'autoconsommation d'énergie peut être définie comme l'association d'une production locale d'énergie à l'aide de panneaux photovoltaïques ou d'une éolienne par exemple, et la consommation de tout ou partie de cette énergie [10, 11]. En pratique, seule une faible part de cette énergie est consommée par le producteur à savoir entre 25 et 40% d'après [10]. Différents phénomènes expliquent ces chiffres modestes d'autoconsommation. Pour commencer, une part importante de la consommation énergétique provient du bâtiment lui-même plus que de l'utilisation d'appareils électroménagers. Ainsi, selon [4] et [22], 57% de la consommation finale d'énergie des foyers en Europe est utilisée pour le chauffage. Mais ces périodes de consommation d'énergie pour le bâtiment ne coïncident pas nécessairement avec les périodes de production locale. Par exemple, en hiver, le chauffage doit fonctionner en permanence, même lorsque la production d'énergie par les panneaux photovoltaïques (PV) est nulle en journée. Il en va de même pour la consommation liée à l'activité des habitants qui sont plus souvent chez eux le matin ou le soir, c'est à dire lorsque la production des panneaux PV est plus faible ou nulle.

Dans ce contexte, plusieurs auteurs proposent de mettre en place des échanges d'énergie entre

des bâtiments auto consommateurs pour optimiser l'utilisation de l'énergie produite localement [18, 20]. Pour réaliser de tels échanges, il faut pouvoir anticiper les activités des foyers et les besoins de consommation associés. Prédire les activités humaines et les courbes de charge des foyers est donc d'une importance cruciale pour créer des solutions pratiques pour le contrôle de l'énergie dans le contexte de l'autoconsommation collective (ACC).

Dans cet article, nous proposons d'utiliser la simulation multi-agents des activités humaines couplée à un modèle d'énergétique du bâtiment pour étudier l'autoconsommation énergétique et les échanges d'énergie entre trois foyers. Le modèle que nous proposons est une simulation multi-niveau où chaque foyer, représenté par un SMA, aura des interactions avec ses voisins. Chaque SMA sera en plus co-simulé avec des moteurs thermiques. Ce couplage de simulation permet l'étude de différentes solutions techniques comme les échanges d'énergie [18], le stockage d'énergie [3] ou la gestion de la charge [19]. Le but général de notre travail est d'étudier l'efficacité de politiques et de solutions technique pour l'échange d'énergie à l'échelle du quartier.

L'article présente une première implémentation d'une politique spécifique d'échange d'énergie et est organisé comme suis : dans la section 2, nous discutons des travaux antérieurs portant sur l'autoconsommation d'énergie, les échanges d'énergie et l'activité humaine. La section 3 est dédiée à la présentation du simulateur SMACH (Simulation Multi-Agents de l'Activité Humaine) utilisé pour la simulation dynamique de la consommation énergétique dans les bâtiments. La section 4 décrit le modèle d'autoconsommation collective. Enfin, la section 5 présente le cas d'étude et les premiers résultats obtenus.

2 État de l'art

2.1 Simulation multi-agents de l'activité et des consommations énergétiques

La simulation du comportement humain lié à la prédiction de la consommation électrique a été un centre d'intérêt grandissant ces dernières années [7, 9, 2]. C'est pourquoi beaucoup d'études basées sur les systèmes multi-agents (SMA) proposent des modèles pour étudier le lien entre l'activité humaine et la consommation électrique.

Par exemple, les auteurs de [7] ont utilisé une méthode basée sur un modèle psychologique qui représente les habitudes des habitants d'un foyer. Cela a permis la production de courbes de charge à partir de l'utilisation des appareils électriques. Cependant, le comportement thermique du bâtiment n'est pas simulé alors que c'est la source principale de consommation d'énergie.

A contrario, les auteurs de [9] proposent une co-simulation entre le bâtiment et l'activité humaine. Le comportement humain est simulé grâce à la plateforme Brahms [17] qui permet une description fine des comportements humains et des appareils électriques. Cependant, ce modèle n'inclut pas l'autoconsommation énergétique et la possibilité de simuler des échanges entre les foyers.

L'approche proposée par le projet SMACH [2] est basée sur trois composantes principales. Premièrement, le simulateur SMACH permet la génération d'une population qui sera simulée. Cette population peut être construite à partir de critères spécifiques ou à partir de données statistiques de population. Le second composant de la plateforme SMACH est un SMA. Ce dernier simule l'activité humaine à l'aide de données issues d'études statistiques comportementales [14]. Le dernier composant est le modèle thermique multizone du bâtiment qui simule les températures intérieures et la consommation d'énergie. Le bâtiment est modélisé en utilisant la librairie BuildSysPro et le langage Modelica. Le SMA et le modèle d'énergie sont co-simulés au sein de SMACH en utilisant le standard d'interopérabilité FMI [13]. Cependant l'autoconsommation énergétique et les échanges d'énergie entre bâtiments n'étaient jusqu'à lors pas implémentés dans ce modèle.

La littérature ne propose donc pas aujourd'hui de modèle qui traite à la fois l'activité humaine, la production de courbes de charge et l'ACC.

2.2 Simulation de l'autoconsommation d'énergie

Les différentes recherches conduites sur l'ACC ont été centrées sur l'amélioration du taux d'autoconsommation des foyers. Ce taux est défini comme le ratio de l'énergie générée localement sur l'énergie consommée par le producteur sur une période donnée. Dans leur état de l'art, les auteurs de [10] ont étudié différentes méthodes pour améliorer l'autoconsommation de l'électricité produite par les panneaux photovoltaïques.

Ils proposent de mettre en œuvre deux solutions : la gestion de la demande et le stockage.

Les travaux de [3] ont traité l'impact de l'utilisation d'une batterie sur le taux d'autoconsommation. Les auteurs ont montré que la quantité d'énergie locale consommée peut atteindre jusqu'à 82% de la production pour un foyer individuel. Les auteurs ont également simulé la génération d'énergie avec un modèle de panneaux PV construit à l'aide de données réelles. Cependant, ils n'ont pas utilisé la simulation pour obtenir les données de la consommation d'énergie, mais se sont servi de profils moyens de consommation.

De même, dans [12], les auteurs utilisent des simulations pour montrer que l'utilisation de véhicules électriques comme moyen de stockage de l'électricité permet une amélioration du taux d'autoconsommation de 37%. Les auteurs ont utilisé un modèle stochastique basé sur les chaînes de Markov pour simuler l'activité humaine. Pour ces deux références ([3] et [12]), le modèle d'activité humaine n'est pas suffisamment précis pour rendre compte de la diversité des comportements humains.

D'autres travaux, comme [19], se sont intéressés à la gestion de la demande énergétique pour améliorer l'autoconsommation. Il s'agit de déplacer des activités pour qu'elles correspondent aux périodes de production d'énergie locale. En théorie, les résultats présentés sont excellents mais ils sont difficiles voire impossibles à mettre en œuvre en pratique. En effet, les individus ne peuvent pas toujours déplacer dans le temps certaines activités comme proposé par l'algorithme d'optimisation.

Les systèmes de stockage d'énergie et de gestion de la demande ne sont pas les seuls moyens d'améliorer le taux d'autoconsommation. Avec de l'énergie produite localement, il devient possible pour un foyer de partager sa production d'énergie locale avec son voisinage. La technologie Smart Grid, ou réseau intelligent, peut se définir comme un réseau avec des flux bidirectionnels d'informations et d'électricité entre les foyers et les fournisseurs[5]. Les recherches actuelles sur les échanges d'énergie se concentrent sur l'étude des algorithmes d'échanges. Ainsi les auteurs de [18] présentent des algorithmes basés sur la théorie des jeux pour réguler les échanges d'énergie. Dans [21], les auteurs ont utilisé un algorithme de jeux non-coopératifs pour échanger de l'énergie entre différents foyers. Cependant, les différents modèles décrits ne s'intéressent pas aux liens entre l'ACC et l'activité humaine. La plupart du temps, les modèles s'appuient principalement sur des données de la consommation d'énergie statique sans modèle d'activité humaine.

Concernant les échanges d'énergie lié à l'autoconsommation, l'étude de la littérature et nos premières enquêtes ont mis en avant un premier ensemble de politiques d'échanges entre les foyers :

- Politique de répartition : chaque foyer est autorisé à prend un taux fixe d'énergie sur la production
- Politique coopérative : L'énergie produite est échange entre les foyers selon des règles contractuelles
- Politique concurrentielle : L'énergie sera achetée et vendue sur un marché de l'énergie où les vendeurs et acheteurs interagiront.

Cette article se concentre sur une seule des politiques identifiées. Il s'agit d'une première implémentation pour étudier l'interaction entre l'autoconsommation collective et l'activité humaine. Dans ce cas d'étude les systèmes de stockage ne seront pas étudiés. Mais ils seront ajoutés dans les travaux futurs car comme cela a déjà été démontré, ils ont un impact important sur l'autoconsommation et les échanges d'énergie.

3 Simulation de l'activité humaine

Dans notre travail, la simulation de l'activité humaine liée à la consommation énergétique est réalisée en utilisant la plateforme SMACH dont les développements successifs ont déjà été présentés aux JFSMA [1, 16, 15]. Dans cette plateforme, les agents représentent les habitants d'un foyer. Chaque agent va sélectionner des tâches quotidiennes à réaliser. La séquence des actions, choisies par un agent à chaque minute, produit un diagramme d'activité. De plus, les diverses activités sont associées à des appareils qui consomment de l'électricité. Le simulateur est donc capable de générer une courbe de charge associée à un diagramme d'activité (Figure 1).

Dans les sections suivantes, nous présentons succinctement les deux dimensions principales de la simulation : le modèle d'activité humaine et le modèle d'agent.

3.1 Le modèle d'activité

La problématique de la plateforme SMACH est de produire des diagrammes d'activités, représentatifs des activités humaines au cours d'une

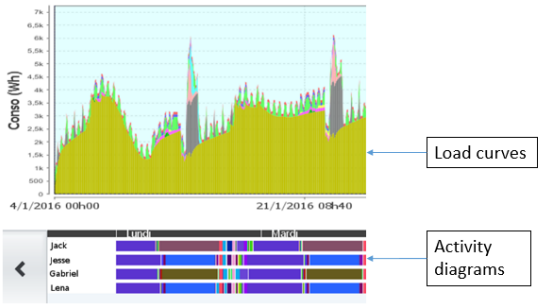


FIGURE 1 – Exemple de courbe de charge et de diagramme d'activité générés par le simulateur SMACH

journée, et cohérents avec les différents foyers. SMACH doit également générer de la variabilité dans les activités des occupants d'un jour à l'autre [6], afin de pouvoir simuler une année réaliste.

Pour atteindre ce but, la plateforme SMACH utilise une approche orientée données basée sur les résultats de l'Enquête Emplois du Temps de l'INSEE (TUS - Time Use Surveys) [14]. Cette enquête est composée de plusieurs dizaines de milliers de carnets où chaque participant a noté son activité courante toutes les 10 minutes sur une journée complète. Ces carnets ont été transformés en une base de données de profils (âge, emploi, catégorie socio-professionnelle...) liés aux activités décrites.

Le simulateur SMACH génère tout d'abord des foyers réalistes issus des bases de données statistiques nationales de la population. Chaque foyer est composé d'un ou plusieurs agents décrits par leur profil. Le bâtiment dans lequel ils vivent est caractérisé par la réglementation thermique selon son année de construction, et par les équipements du foyer (appareils électriques). Pour chaque profil d'individu et pour chaque tâche possible qu'un individu peut choisir d'effectuer, le simulateur SMACH extrait les informations statistiques suivantes de l'Enquête Emploi du Temps :

- Le nombre moyen de répétitions de l'activité par jour ou par semaine (rythme) ;
- Les périodes préférentielles de réalisation de la tâche au cours de la journée ou de la semaine ;
- La durée de la tâche.

Par exemple, le simulateur SMACH peut extraire de l'Enquête Emplois du Temps qu'un individu de classe moyenne, actif, entre 20 et 40 ans vivant dans une maison individuelle en ville fera ses courses une fois par semaine, aux alentours de 20h00, avec un écart type d'une heure et une durée de 30 minutes avec un écart type de 15 minutes.

Cet ensemble de tâches possibles extrait par le simulateur SMACH donne une description statique de la vie quotidienne d'un foyer.

3.2 Le modèle d'agent

La seconde partie du simulateur SMACH consiste en la transformation de cette description statique en une simulation dynamique.

Pour chaque pas de temps, les agents sélectionnent une tâche à réaliser. Cette sélection est faite grâce à un mécanisme basé sur un niveau de priorité dynamique. Ce niveau est modifié par des paramètres qui influencent le choix de l'activité. Par exemple, l'heure courante et le nombre de réalisations d'une activité, dépendant des périodes préférentielles et du rythme, impactent la sélection de la tâche. Le mécanisme de sélection prend aussi en compte l'activité courante pour éviter les phénomènes d'oscillations entre des tâches similaires, et considère aussi les tâches réalisées par les autres membres du foyer pour satisfaire des contraintes de dépendance entre les tâches et la réalisation des activités à plusieurs. Un autre paramètre important dans la sélection des tâches est le prix courant de l'électricité. Cela peut amener les agents à choisir des activités moins consommatrices durant les périodes où le prix est élevé, en fonction de la politique énergétique du foyer.

3.3 Évaluation

Les résultats des simulations ont été validés en suivant une approche de simulation participative par la comparaison des diagrammes d'activité et des consommations énergétiques des familles simulées avec des familles réelles [8]. Les participants pouvaient contrôler les différents agents du foyer en sélectionnant leurs activités manuellement pour ensuite valider le modèle d'agent en comparant le diagramme d'activité obtenu avec leur propre expérience. La courbe de charge générée par le simulateur a également été comparée aux courbes de charge réelles du foyer simulé.

Dans [14], les auteurs ont proposé une validation du modèle d'activité à un plus haut niveau. Ils ont généré une grande quantité de diagrammes d'activité grâce au simulateur SMACH et ont

comparé les résultats avec les carnets des TUS. Les auteurs ont montré que les diagrammes d'activité simulés sont similaires aux carnets d'un point de vue statistique avec quelques variabilités au niveau individuel. L'agrégation des courbes de charge produite par le simulateur SMACH a également été comparée à l'agrégation des données de courbes de charge obtenues sur le réseau démontrant une concordance statistique des résultats de la plateforme.

4 Modèle pour l'autoconsommation collective

Notre contribution vise à intégrer l'autoconsommation énergétique et les échanges d'énergie dans la plateforme SMACH. La figure 2 illustre le fonctionnement général du nouveau simulateur.

L'ajout de l'ACC est fait grâce à trois composants. Tout d'abord, nous avons implémenté un simulateur de panneaux PV. Deuxièmement, nous définissons des règles d'autoconsommation d'électricité pour les différents appareils. Enfin, nous définissons une politique d'échange d'énergie entre les foyers.

4.1 Ajout de panneaux photovoltaïques pour simuler l'autoconsommation

Le modèle thermique de bâtiment couplé avec le modèle d'activité du simulateur SMACH a initialement été développé dans le langage Modelica et la librairie BuildSysPro [13]. Pour l'implémentation du modèle de panneaux PV, nous avons décidé d'utiliser la même librairie pour sa simplicité à définir des interfaces physique-thermiques.

Le modèle simule des panneaux photovoltaïques de type silicium cristallin. Il prend en entrée un fichier météorologique et calcule la puissance électrique produite (en kW) en fonction de différents paramètres :

— La surface ;
— L'inclinaison ;
— L'azimut ;

Nous avons donc ici une co-simulation entre trois éléments : un modèle de panneaux PV, un modèle de thermique du bâtiment et un système multi-agent relatif à l'activité des habitants. Le modèle de panneaux PV est stocké (tout comme celui du bâtiment)dans un composant exécutable FMU("Fonctionnal Mock-Up Unit") qui permet des échanges d'informations entre les différents composants du simulateur. Cependant, bâtiment et panneaux PV sont stockés dans deux FMU différents. Cela permet de modifier les deux modèles de façon complètement indépendante et même de pré-calculer la génération de l'énergie des panneaux PV si nécessaire (pour faciliter le passage à l'échelle).

4.2 Évolution du modèle de consommation d'énergie

Nous avons modifié la façon dont les appareils électriques du foyer consomment l'énergie pour prendre en compte la production locale d'électricité.

Dans le modèle initial, le dispositif de production de l'Eau Chaude Sanitaire (ECS) dispose de deux modes de fonctionnement. Lorsque l'appareil est asservi aux heures creuses, il ne sera mis en fonctionnement que durant ces plages horaires où le coût de l'électricité est avantageux. S'il n'est pas asservi aux heures creuses, la production d'ECS obéit au respect de la température de consigne de l'eau. Dans notre nouveau modèle, nous avons ajouté une nouvelle règle sur la production d'ECS. Dès que les panneaux PV produiront une quantité suffisante d'électricité, la production d'ECS est autorisée. Asservir la production d ?ECS aux heures creuses permettra de lancer cette production à différents moments : Lors des heures creuses (électricité moins chère) où lorsque la production d ?électricité des PV est suffisante.

Le déclenchement de la production de l'ECS en fonction de la puissance produite par les panneaux PV n'est autorisée que si l'électricité produite est supérieure à un seuil. Nous avons fixé ce seuil à une valeur égale à la moitié de la puissance installée pour la production d'ECS.

Pour les autres appareils du foyer, premièrement, nous évaluons les besoins en électricité de l'ensemble des appareils hors ECS. Puis, nous déduisons de la puissance produite par les panneaux PV la consommation pour produire l'ECS. Si le solde d'électricité est supérieur à la puissance nécessaire pour alimenter tous les appareils nous utilisons la puissance produite par les panneaux PV et le surplus d'énergie pourra être redistribué vers autres foyers. Si ce solde est inférieur à la demande des autres appareils , le système achète au fournisseur d'énergie la différence entre la production et la consommation demandée.

Avec ce mode de fonctionnement, nous garantissons que la production d'ECS est toujours assu-

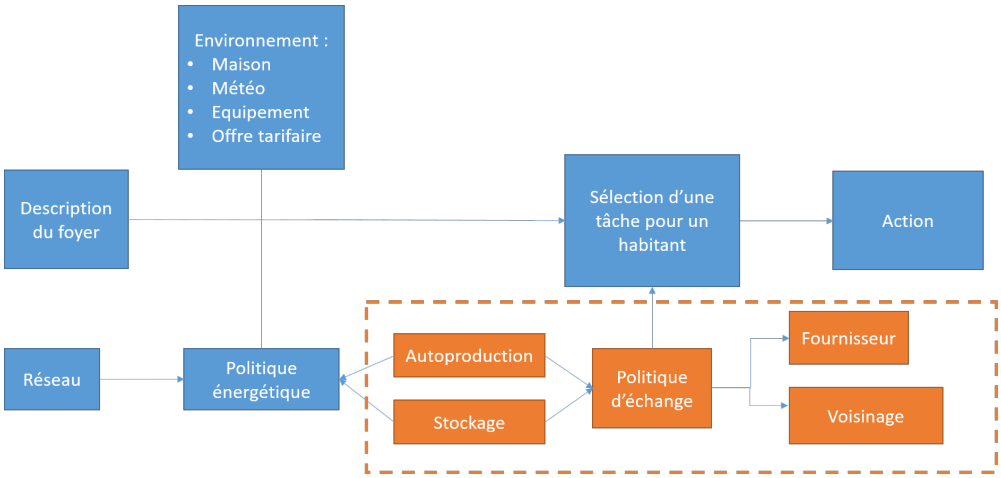


FIGURE 2 – Modèle SMACH avec l'ajout de l'autoconsommation collective

rée.

Dans la section suivante, nous nous concentrons seulement à l'impact de l'ACC sur la consommation d'énergie due à la production d'ECS, bien que l'ensemble des appareils soient pris en compte dans le modèle.

4.3 Modélisation des échanges entre les bâtiments

Dans le cadre de l'autoconsommation collective trois types de foyers peuvent être mis en avant en fonction de leurs équipements de production et de leur politique vis à vis des échanges :

- Des foyers producteurs d'énergie, qui peuvent fournir de l'énergie aux autres foyers ;
- Des foyers consommateurs d'énergie, qui peuvent seulement utiliser l'énergie des producteurs mais n'en produisent pas eux-même ;
- Des foyers à la fois producteurs et consommateurs d'énergie.

Ces trois types de foyers pourront également acheter de l'électricité à leur fournisseur d'énergie. Cinq états dans le processus de l'ACC ont donc été définis :

1. Le foyer consomme sa propre production ;
2. Le foyer consomme l'excès de production de ses voisins ;
3. Le foyer envoie l'excès d'énergie de sa propre production à ses voisins ;
4. Le foyer consomme de l'énergie du fournisseur institutionnel (qu'on appelle ici « réseau ») ;
5. Le foyer vend de l'énergie sur le réseau.

La décision d'échanger ou non de l'énergie dépend de la politique d'échange de chaque foyer. Cette politique peut aussi affecter la quantité et le prix de l'énergie échangée. En première hypothèse, nous avons adopté une politique d'échange coopérative dans laquelle le producteur donnera tout son excès d'énergie à ses voisins sans contrepartie. En plus, les voisins chercheront à consommer le maximum d'énergie pour répondre à leur besoin.

Dans cette première implémentation, nous nous concentrons sur les interactions entre un producteur et plusieurs consommateurs, en considérant les état 1, 2 et 4 énoncés plus haut.

Le modèle comporte deux parties. Premièrement, nous simulons un pas de temps avec uniquement le foyer producteur d'énergie. Il générera de l'énergie grâce à ces panneaux PV et consommera cette énergie en fonction de ses besoins. Puis, il enverra un message aux autres consommateurs indiquant la quantité d'énergie restante qu'il peut échanger. Dans un second temps, on simule le même pas de temps pour tous les foyers consommateurs. Ces derniers utilisent l'information transmise par le foyer producteur d'énergie pour leur propre consommation énergétique. Enfin, ils notifient au producteur la quantité d'énergie consommée. Tous les foyers utilisent le même modèle de consommation pour l'énergie échangée : d'abord l'ECS puis les autres appareils.

Nous avons donc une approche multi-niveau de l'ACC. Chaque foyer pris individuellement est un système multi-agent qui va simuler l'activité des

individus ainsi que leur consommation d'énergie. Cependant, en regardant à l'échelle d'un quartier, chaque foyer est aussi un agent qui va échanger son surplus d'énergie avec les autres foyers.

5 Cas d'étude et résultats

5.1 Contexte de simulation

Nous considérons 3 foyers individuels A, B et C dans le même quartier de la ville de Nice en France. Les trois foyers sont décrits dans le Tableau 1 et les familles dans le Tableau 2. Deux familles de personnes actives (foyers A et C) et un foyer de retraités (Foyer B) sont simulés. Les foyers ont été générés grâce aux enquêtes emplois du temps [16].

	Foyer A	**Foyer B**	**Foyer C**
Surface	$102m^2$	$110m^2$	$102m^2$
Réglementation Thermique	1974	1982	1974
Chauffage électrique	Non	Non	Non
Heures creuses	23h-07h	22h30-6h30	23h-07h
Panneaux PV	Oui	Non	Non
Ville	Nice	Nice	Nice
Nb. habit.	3	2	3

TABLE 1 – Paramètres des trois maisons

	Situation	**Genre**
Foyer A	Actif	Femme
	Actif	Homme
	Actif	Homme
Foyer B	Retraité	Femme
	Retraité	Homme
Foyer C	Actif	Femme
	Actif	Homme
	Actif	Homme

TABLE 2 – Paramètres des trois familles

Les diagrammes d'activité de chaque famille sont tous différents. Cependant, pour les foyers A et C, il y aura des activités similaires, comme le fait que les membres de ces foyers partent au travail durant la journée (Figure 3). La famille du foyer A est majoritairement à l'extérieur pendant la journée. C'est à dire là où est produite la plus grande partie de l'énergie alors que la famille B est chez elle sur cette période.

Le foyer A sera producteur d'énergie grâce à ses panneaux PV tandis que B et C ne seront que consommateurs d'énergie. Pour le producteur d'énergie, les panneaux PV possèdent les caractéristiques suivantes :

- Surface : $20m^2$;
- Inclinaison : 30° ;
- Azimut : Sud.

La puissance électrique des panneaux PV est égale à 2,5kWc (kilowatt crête). Nous avons dimensionné les panneaux PV afin que leur production d'électricité locale soit suffisante pour supporter, en plus de la consommation d'électricité du foyer A, une part de l'énergie consommée par le foyer B.

En utilisant ces modèles, nous évaluons l'impact de l'ACC sur la consommation d'énergie des trois familles.

5.2 Impact de l'autoconsommation

La Figure 4 montre la consommation d'électricité pour la production d'ECS et des autres appareils du foyer A pendant une journée et sans production d'énergie par les panneaux PV. La production d'ECS n'est autorisée que pendant les heures creuses du foyer A. La consommation d'électricité totale est égale à **165kWh** pour un mois complet.

L'ajout de l'autoconsommation (Figure 5) vise à diminuer la consommation d'électricité en provenance du réseau pour la production de l'ECS et à déplacer sa consommation d'électricité sur les périodes durant lesquelles les panneaux PV produisent de l'électricité. Ainsi, une part de la consommation électrique pour la production de l'ECS a été déplacée pendant la journée.

La production d'ECS consommera donc moins d'énergie en provenance du réseau. Cependant, si l'on regarde la consommation totale d'énergie, elle est égale à **180 kWh** pour un mois. Cette différence peut provenir de deux phénomènes :

- Tout d'abord, chaque simulation multi-agent d'un foyer produit un diagramme d'activité différent : c'est le principe de la plate-forme SMACH. Les consommations seront donc nécessairement différentes, à la marge ;
- Les panneaux PV peuvent conduire à une surconsommation du foyer car la production de l'ECS est sollicitée plus souvent sur les périodes de production d'électricité avec les panneaux PV.

La consommation d'électricité provenant du réseau pour la production de l'ECS a été diminuée pour atteindre la valeur de **104,9 kWh**. Cependant, le foyer A n'a consommé que 40% de l'énergie produite par les panneaux PV. Le reste de la production d'électricité a été rendue au réseau.Le principe de l'ACC pour le simulateur

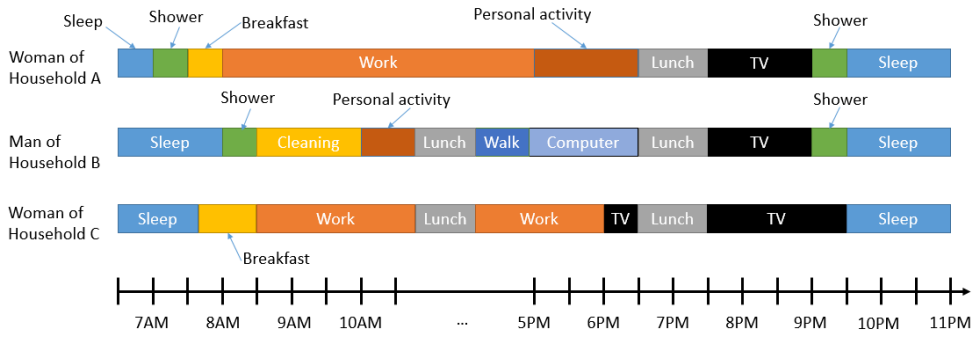


FIGURE 3 – Exemple de diagrammes d'activités pour un membre de chaque foyer

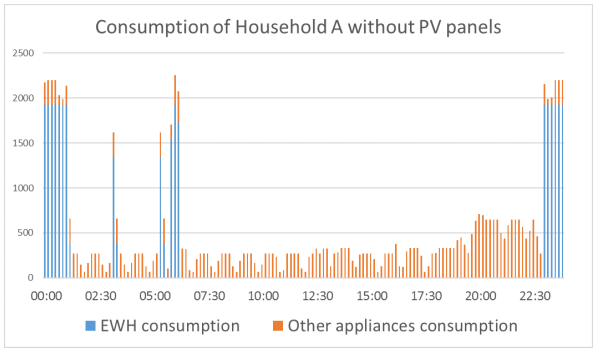


FIGURE 4 – Consommation électrique totale pour le foyer A et sans panneaux PV

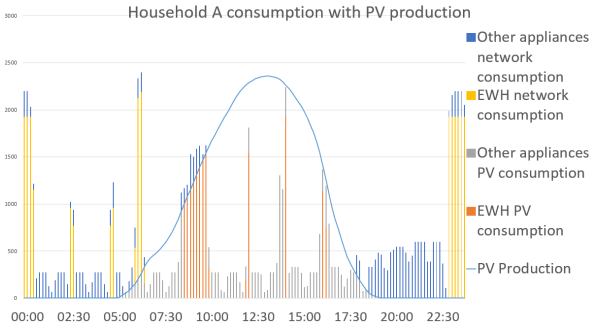


FIGURE 5 – Consommation totale d'électricité pour le foyer A avec production d'électricité par les panneaux PV

SMACH est de permettre d'utiliser cette énergie grâce aux échanges avec d'autres foyers.

La consommation d'électricité pour la production d'ECS reste élevée même en ajoutant l'électricité produite par les panneaux PV. En effet, 38% de l'électricité provient toujours du réseau car cette électricité est consommée majoritairement pendant la nuit.

5.3 Impact de l'autoconsommation collective

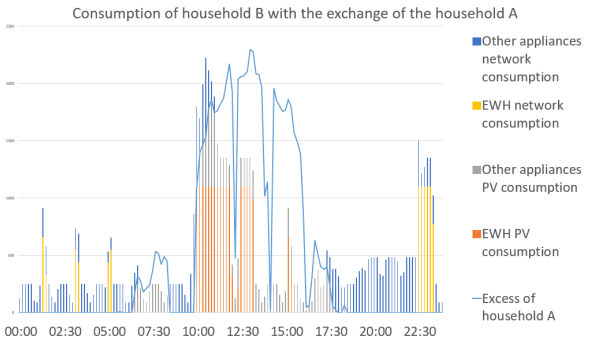


FIGURE 6 – Consommation d'électricité du foyer B avec des échanges

La Figure 6 montre la consommation d'électricité du foyer B lorsque le foyer A lui donne son surplus d'énergie. La consommation d'énergie du foyer B intervient majoritairement entre 10h00 et 17h00. La consommation d'électricité pour la production de l'ECS est la plus élevée. Enfin, la consommation d'électricité en soirée est essentiellement due à des activités telles que la cuisine et la vaisselle qui utilisent de l'ECS.

Les échanges d'électricité permettent de couvrir 38% de la consommation du foyer B. De plus cela permet au groupement de foyers A+B d'utiliser 68% de l'énergie produite par les panneaux PV de A. Donc, la production d'électricité du foyer A est suffisante pour couvrir une large partie de la consommation d'électricité du foyer B et ceci grâce aux échanges.

Cependant, encore 32% de la production d'électricité du foyer A reste inutilisée. C'est pourquoi nous avons décidé d'ajouter un autre foyer aux échanges à savoir le foyer C. Ce foyer consommera l'excès d'énergie du foyer B et plus préci-

sément 38% de la production électrique du foyer A.

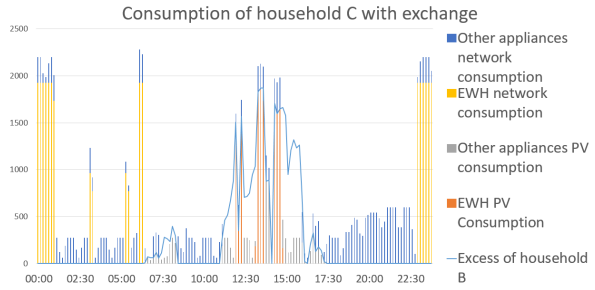


FIGURE 7 – Consommation d'électricité du foyer C avec des échanges

La Figure 7 montre la consommation du foyer C durant une journée. Le profil de la courbe de charge de ce foyer est similaire à celui du foyer A. Mais en regardant plus attentivement, seulement 24% de la consommation électrique totale de C provient des échanges effectués avec le foyer B. Cela s'explique par le fait que les foyers A et B consomment une large partie de la production d'électricité avant de la diriger ver le foyer C. De plus, il y a un toujours un excédent d'électricité que l'implémentation des échanges n'a pas permis de consommer. Ajouter plus de foyers consommateurs ne serait pas réaliste car cela ne ferait que diminuer la quantité d'énergie échangée entre chaque foyer. Il faut également noter que le gain du foyer C est faible. Cela est dû au fait que les membres de la famille sont absents durant la journée. Cet exemple montre que l'intégration de l'ACC est indissociable de la prise en compte de l'activité des habitants.

Avec l'intégration des échanges, seulement 17% de l'électricité générée par le foyer A est inutilisée (contre 62% sans les échanges). Cette énergie pourrait être utilisée pour alimenter d'autres appareils au sein des trois foyers à condition que les habitants intègrent la production d'électricité des panneaux PV dans le choix de leurs activités.

6 Conclusion

Dans cette étude, nous avons présenté une extension du modèle de simulation multi-agent SMACH pour prendre en compte l'autoconsommation collective. Ce modèle se décompose en deux niveaux : un ensemble de SMA simulant l'activité et la consommation d'électricité de chaque foyer, et un modèle de coordination entre les foyers permettant l'échange d'énergie. Nous avons illustré les possibilités sur un exemple simple d'ACC. Les résultats montrent que l'étude de l'autoconsommation d'électricité ne peut pas se faire sans intégrer l'activité humaine et qu'une part importante de l'énergie peut être partagée dans certaines configurations de foyers , par exemple une famille d'actifs et un couple de retraités.

Le modèle actuel ne prend en compte qu'une seule politique d'échange (coopérative). Nous travaillons actuellement sur une extension permettant d'intégrer différents types d'échanges. Nous devons aussi encore valider ce modèle dans des configurations plus riches intégrant plusieurs producteurs et des échanges avec le réseau (fournisseur d'énergie institutionnel).

Pour ce qui est des résultats de l'étude que nous avons présentée ici, l'autoconsommation de l'électricité pourrait être améliorée par différents moyens. Une première solution serait d'autoriser les habitants à déplacer certaines activités durant les périodes de production d'électricité par les panneaux PV en modifiant le modèle de décision du simulateur SMACH. Mais, certaines consommations d'électricité sont difficiles voir impossible à déplacer comme celle du réfrigérateur ou de l'éclairage. De plus, les activités déplaçables dans le temps représentent une faible part de la consommation totale d'électricité des foyers. Ainsi, le gain énergétique serait relativement faible.

Une deuxième solution serait d'introduire des systèmes de stockage d'électricité qu'ils soient collectifs ou individuels mais cela amènera une modification de la politique d'échange d'électricité entre les foyers.

Références

[1] Edouard Amouroux, Thomas Huraux, François Sempé, and Nicolas Sabouret. SMACH : Simuler l'activité humaine pour limiter les pics de consommation électrique. *Journées Francophones sur les Systèmes Multi-Agents (JFSMA)*, 2013.

[2] Édouard Amouroux, Thomas Huraux, François Sempé, Nicolas Sabouret, and Yvon Haradji. Simulating human activities to investigate household energy consumption. In *Proceedings of the 5th International Conference on Agents and Artificial Intelligence*, volume 2, pages 71–80, 2013.

[3] Martin Braun, Kathrin Büdenbender, Dirk Magnor, and Andreas Jossen. Photovoltaic self consumption in Germany using

Li-ion storage to increase self-consumed PV energy. *Fraunhofer IWES*, (34), 2009.

[4] Dorota Chwieduk. Towards sustainable-energy buildings. *Applied Energy*, 76(1-3) :211–217, sep 2003.

[5] Hassan Farhangi. The path of the smart grid. *IEEE Power and Energy Magazine*, 8(1) :18–28, jan 2010.

[6] Martha S. Feldman and Brian T. Pentland. Reconceptualizing Organizational Routines as a Source of Flexibility and Change. *Administrative Science Quarterly*, 48(1) :94, mar 2003.

[7] Eric Ferreri, Jean-Marc Salotti, and Pierre-Alexandre Favier. Prediction of electrical consumption using a bio-inspired behavioral model. *BauSIM2014*, 2017.

[8] Thomas Huraux, Nicolas Sabouret, Yvon Haradji, and François Sempé. Simulations multi-agents de l'activité humaine : application dans le contexte énergétique résidentiel français. *Applications Pratiques de l'Intelligence Artificielle*, 2015.

[9] Ayesha Kashif, Stéphane Ploix, Julie Dugdale, and Xuan Hoa Binh Le. Simulating the dynamics of occupant behaviour for power management in residential buildings. *Energy and Buildings*, 56 :85–93, jan 2013.

[10] Rasmus Luthander, Joakim Widén, Daniel Nilsson, and Jenny Palm. Photovoltaic self-consumption in buildings : A review. *Applied Energy*, 142 :80–94, 2015.

[11] Eduardo Matallanas and Felix Monasterio. Analysis of the Self-consumption Possibilities in Small Grid-connected Photovoltaic Systems in Spain. *26th European Photovoltaic Solar Energy Conference and Exhibition*, (1) :4619–4624, 2011.

[12] Joakim Munkhammar, Pia Grahn, and Joakim Widén. Quantifying self-consumption of on-site photovoltaic power generation in households with electric vehicle home charging. *Solar Energy*, 97 :208–216, 2013.

[13] Gilles Plessis, Aurélie Kaemmerlen, and Amy Lindsay. BuildSysPro : a Modelica library for modelling buildings and energy systems. *Proceedings of the 10th International Modelica Conference*, 2014.

[14] Quentin Reynaud, Yvon Haradji, François Sempé, and Nicolas Sabouret. Using Time-Use Surveys in Multi Agent Based Simulations of Human Activity. *Proceedings of the 9th International Conference on Agents and Artificial Intelligence - Volume 1 : ICAART*, (Icaart) :67–77, 2017.

[15] Quentin Reynaud, Nicolas Sabouret, Yvon Haradji, and François Sempé. Simulation de l'activité humaine. Une étude sur le réalisme multi-niveau. *Revue d'Intelligence Artificielle*, 32(2) :197–221, 2018.

[16] Quentin Reynaud, François Sempé, Yvon Haradji, and Nicolas Sabouret. Simuler l'activité humaine en combinant un système multi-agent avec des approches statistiques basées sur les "enquêtes emploi du temps". In *Journées Francophones sur les Systèmes Multi-Agents (JFSMA 2017)*, jul 2017.

[17] Maarten Sierhuis, William J. Clancey, and Ron J.J. Van Hoof. Brahms : a multi-agent modelling environment for simulating work processes and practices. *International Journal of Simulation and Process Modelling*, 3(3) :134, 2007.

[18] Wayes Tushar, Chau Yuen, Hamed Mohsenian-Rad, Tapan Saha, H. Vincent Poor, and Kristin L. Wood. Transforming energy networks via peer-to-peer energy trading : The potential of game-theoretic approaches. *IEEE Signal Processing Magazine*, 35(4) :90–111, 2018.

[19] Joakim Widén. Improved photovoltaic self-consumption with appliance scheduling in 200 single-family buildings. *Applied Energy*, 126 :199–212, 2014.

[20] Yuan Wu, Xiaoqi Tan, Liping Qian, and Danny H K Tsang. Optimal management of local energy trading in future smart microgrid via pricing. *Proceedings - IEEE INFOCOM*, 2015-Augus(1) :570–575, 2015.

[21] Naouar Yaagoubi and Hussein T. Mouftah. Energy trading in the smart grid : A distributed game-theoretic approach. *Canadian Journal of Electrical and Computer Engineering*, 40(2) :57–65, 2017.

[22] Jean-Paul Zimmermann, Jonathan Griggs, Nicola King, Les Harding, Roberts Penelop, and Chris Evans. Household electricity survey a study of domestic electrical product usage. Report, RAND Corporation, 2012.

Modélisation multi-agent des opérations semi-autonomes dans un système cyber-physique de forage pétrolier ou gazier

Yazan Mualla[a] yazan.mualla@utbm.fr
Igor Haman Tchappi[a,b] igortchappi@gmail.com
Amro Najjar[c,d] amro.najjar@uni.lu
Stéphane Galland[a] stephane.galland@utbm.fr
Robin Vanet[e] robin.vanet@telecom-st-etienne.fr
Olivier Boissier[e] Olivier.Boissier@emse.fr

[a]CIAD, Univ. Bourgogne Franche-Comté, UTBM, F-90010 Belfort, France

[b]Faculty of Sciences, University of Ngaoundere, Cameroon

[c]AI-Robolab/ICR, Computer Science and Communications, University of Luxembourg, Luxembourg

[d]Umeå University, 90736 Umeå, Sweden

[e]Univ. Lyon, IMT Mines Saint-Étienne, F-42023 Saint-Étienne, France

Résumé

Dans l'industrie pétrolière et gazière, après une certaine profondeur de forage, les températures augmentent suffisamment pour endommager les outils de forage et les processus d'atténuation ne sont plus suffisants. Dans cet article, nous proposons un système cyber-physique (SCP) où des agents sont utilisés pour représenter des entités en collaboration, à la fois en surface et en profondeur. Avec le SCP présenté, les outils souterrains répondent de manière autonome aux hautes températures avec une prise de décision démocratique décentralisée, basée sur un modèle de décision interne : chaque outil prend une décision en fonction de ses spécifications afin de supporter les hautes températures. Le SCP est implémenté en utilisant une simulation orientée-agent, et les résultats montrent qu'il est possible d'atténuer les conséquences des hautes températures en associant des mécanismes de vote et des modèles de refroidissement.

Mots-clés : *systèmes cyber-physiques, systèmes multi-agents, opérations de forage pétrolier et gazier*

Abstract

In Oil&Gas drilling operations and after reaching deep drilled depths, high temperature increases significantly enough to damage the down-hole drilling tools, and the existing mitigation process is insufficient. In this paper, we propose a Cyber-Physical System (CPS) where agents are used to represent the collaborating entities in Oil&Gas fields both up-hole and down-hole. With the proposed CPS, down-hole tools respond to high temperature autonomously with a decentralized collective voting based on the tools' internal decision model while waiting for the cooling performed up-hole by the field engineer. This decision model, driven by the tools' specifications, aims to withstand high temperature. The proposed CPS is implemented using a multiagent simulation environment, and the results show that it mitigates high temperature properly with both the voting and the cooling mechanisms.

Keywords: *cyber-physical systems, multiagent systems, oil and gas drilling operations*

1 Introduction

Dans son rapport publié en 2018, l'Agence Internationale de l'Énergie prévoie une augmentation continue de la consommation du pétrole durant les prochaines années. Cette augmentation serait essentiellement due a la hausse des demandes des industries pétrochimiques et des transports routiers et aériens [1]. Dans l'industrie pétrolière et gazière, une plate-forme est généralement utilisée pour forer un trou, appelé puits, dans la croûte terrestre à l'aide d'un outil, appelé le train de forage, afin de trouver des ressources naturelles. Lorsqu'elles sont découvertes, ces dernières sont extraites puis raffinées pour utilisation. Les puits peuvent être distingués selon qu'ils permettent un forage à basse ou à haute température. Les puits à basse température constituent une large majorité des puits existants de part le coût important des matériaux et de la technologie d'assemblage nécessaires à la réalisation d'un puits à haute température. L'augmentation du nombre de puits à basse température totalement drainés implique la progres-

sion croissante de la demande de puits à haute températures. Les températures élevées de ces derniers endommagent les outils. Ce qui génère des coûts supplémentaires dûs au temps nécessaire pour démonter, réparer et remonter l'outil endommagé. La limite de fonctionnement pratique pour les outils de forage existants est de 205 degrés Celsius. Au-dessus de cette limite, il est nécessaire d'investir dans des technologies permettant de gérer et d'atténuer les effets des hautes températures [2]. Par conséquent, les machines ont été équipées de capteurs de température, à la fois dans les outils à plusieurs kilomètres de profondeur et à la surface, afin de déterminer le taux de rafraîchissement à appliquer aux outils. Dans la plupart des systèmes existants, la gestion de la température depuis la surface repose sur un **refroidisseur de boue** qui permet de contrôler la température de la boue de forage qui est en contact avec les outils souterrains. Ce processus souffre de problèmes de réactivité. Tout d'abord, un délai est nécessaire pour que la boue refroidie atteigne le fond du puits. Ceci implique une latence entre l'instant où la température limite est atteinte et son réel refroidissement. De plus, la communication entre les composants sous-terrains et ceux en surface est basée sur un signal analogique transporté par la boue. Ce signal est non fiable (notamment dans des conditions difficiles de forage) : le signal analogique peut être bruité ou perdu.

En résumé, les outils souterrains peuvent détecter une situation problématique, voir critique, correspondant à une augmentation trop importante de la température. Toutefois, ces outils ne pouvant rien faire pour se protéger d'éventuels dommages à l'heure actuelle, il devient nécessaire de les équiper avec des mécanismes leur permettant d'éviter les cas mentionnés précédemment.

Dans le présent article [1], nous proposons un système cyber-physique (SCP), tel que défini par Baheti and Gill [5], qui vise à améliorer les outils souterrains avec un mécanisme de communication permettant d'atténuer de manière autonome les effets des hautes températures, tout en contrôlant un effecteur local à l'aide d'un système de vote. Les outils sont représentés par des agents qui contrôlent les capteurs et les effecteurs embarqués dans les-dits outils. Le paradigme agent fournit la possibilité d'une solution décentralisée au plus proche des outils, en déclenchant un cycle de vote pour démarrer un effecteur local. Le résultat de chaque cycle permettra aux agents de contrôler la montée des températures en attendant le refroidissement déclenché par l'ingénieur de forage depuis la surface. Pour implémenter fidèlement le SCP proposé, un modèle est défini avec tous les mécanismes et les paramètres du forage en situation réelle. Ces paramètres incluent la trajectoire du puits et les équations des températures de ces derniers.

Dans nos travaux antérieurs [3], nous avons proposé un modèle de contrôle multi-agent proche du temps-réel afin d'atténuer les effets des hautes températures durant le processus de forage. Nous avions alors ignoré le rôle de l'ingénieur de forage qui est responsable du refroidissement de la boue en surface. De plus, les règles de vote utilisées par les outils étaient des règles simples. En d'autres termes, notre précédent modèle de décision doit être étendu et complété par l'adjonction de fonctions d'utilité pour déterminer le niveau d'activation des effecteurs.

Cet article est organisé comme suit. La section 2 effectue un état de l'art. La section 3 inclut une description succincte des opérations de forage. La section 4 décrit le SCP proposé. La section 5 évalue le SCP proposé et discute les résultats obtenus. La section 6 tire des conclusions et propose des directions de recherche futures.

2 Synthèse de travaux antérieurs

Une synthèse de travaux antérieurs est présentée dans cette section. La littérature des SCPs et des Systèmes Multi-Agents (SMA) dans l'industrie pétrolière et gazière est détaillée dans la section 2.1. Les travaux liés à la théorie du choix social, et en particulier aux mécanismes de vote sont présentés dans la section 2.2.

2.1 SCP et SMA dans l'industrie pétrolière et gazière

Le rôle des SCPs dans des applications industrielles a été largement exploré dans la littérature [5, 6, 7]. Lee and Lapira [8] proposent d'augmenter l'interaction entre les entités d'un même système pour améliorer la compétitivité afin d'atteindre un processus manufacturier prédictif, transparent et efficace. En revanche, ces contributions sont limitées à la phase de conception, et aucune implémentation pratique n'est réalisée. Lee et al. [9] montrent comment utiliser des analyses avancées, des informations, et des réseaux informatiques afin de travailler de manière plus collaborative et résiliente. Les au-

1. Cet article est une synthèse de deux publications en anglais dans les ateliers RTcMAS-2018 [3] et CHARMS-2019 [4].

teurs proposent une architecture unifiée constituée de 5 niveaux afin de construire une implémentation du SCP. Toutefois, l'architecture proposée ne permet pas de supporter l'hétérogénéité des composants des SCPs. Dans l'industrie pétrolière et gazière, les SCPs se caractérisent par les composants physiques et des processus supervisés par la cyber-infrastructure du système, qui comprend des contrôleurs logiques programmables, des capteurs et un système de contrôle et d'acquisition de données (SCADA). Les données sur les propriétés du pétrole et du gaz, telles que la température, la pression, la densité et la vitesse, sont mesurées par des capteurs qui sont ensuite transmis au système de contrôle, qui applique des algorithmes pour détecter les anomalies et surveiller l'état du système. Une fois qu'une anomalie est détectée, des commandes de contrôle appropriées sont envoyées pour modifier le fonctionnement de l'infrastructure physique [10].

Dans la littérature, les SMA sont couramment utilisés pour coordonner les entités d'écosystèmes distribués, comme par exemple dans le domaine du « cloud computing » [11]. L'utilisation des SMA pour la modélisation des opérations dans l'industrie pétrolière et gazière réduit la quantité d'information à analyser par des opérateurs humains et offre un outil d'aide à la décision et une plate-forme qui permet d'effectuer des simulations de modèles. La majeure partie des travaux utilisant les SMA pour une étude de l'industrie pétrolière et gazière est d'ordre théorique. Il y a toutefois quelques applications concrètes. Par exemple, Mikkelsen and Jørgensen [12] discutent du passage à l'échelle des moyens de production de l'industrie pétrolière et gazière. Ils présentent une nouvelle application des SMA basée sur un contrôle multi-objectif intelligent. Malgré le fait qu'un grand nombre de travaux se soient concentrés sur différents aspects de la chaîne logistique et de commandement [13, 12], aucune étude (au meilleur de notre connaissance) ne s'est penchée sur les processus de forage et de production de pétrole. En dépit du fait que certains travaux ont montré que les SMA peuvent être utilisés effectivement pour la maintenance des outils [14], nous pensons que la perspective adoptée ne couvre pas la totalité des propriétés inhérentes aux SCP, notamment l'intégration des capteurs et des effecteurs physiques.

2.2 Systèmes de vote dans les SMA

La théorie du choix social propose un moyen permettant d'effectuer une décision collective basée sur des opinions possiblement divergentes entre les membres d'une même communauté [15]. Le vote est défini comme un plan général, un moyen permettant de considérer et éventuellement d'agréger les préférences des votants [16].

Dans le domaine des SMA, les systèmes de votes constituent une méthode capable de modéliser et implanter des décisions décentralisées [17]. Les agents représentent les votants avec leurs propres intérêts et objectifs. Cela veut dire que la stratégie la plus plausible pour un agent est de chercher à maximiser son utilité individuelle [18]. Quand différents agents du SMA ont des préférences différentes, voir contradictoires, il devient nécessaire d'avoir un mécanisme permettant aux agents de prendre une décision collective. Ainsi, chaque agent exprime ses préférences parmi les décisions possibles. Puis, un système de vote agrège ces préférences pour déterminer la décision collective [19].

Les règles de vote sont définies pour la gestion du processus de vote en tenant compte de propriétés telles que l'**anonymat** (les votes ne révèlent pas les votants) et la **neutralité** (chaque candidat est traité de la même manière) [20]. Les règles de vote varient selon les processus. Nous présentons ci-après les règles les plus utilisées dans la littérature des SMA :

1. **Pluralité :** Les votants effectuent un seul vote et le candidat qui obtient le plus grand nombre de votes gagne.
2. **Méthode Borda :** Chaque votant classe les candidats dans l'ordre selon ses préférences. Cet ordre donne des points à chaque candidat en fonction de la position que lui a attribué le votant. En d'autres termes, s'il-y-a m candidats, cela donne $m-1$ points au candidat classé au premier rang, $m-2$ points au second, et ainsi de suite. En conséquence, le gagnant est celui qui a le plus grand nombre de points [21].
3. **Méthode Condorcet :** Cette règle élit, le cas échéant, le candidat qui obtiendrait la majorité des suffrages lors du vote à l'unanimité contre chacun des autres candidats. Chaque fois qu'un tel candidat existe, il est appelé vainqueur Condorcet.

3 Technologies et outils de forage

L'assemblage de fond de trou (Bore-hole Assembly, BHA) est une partie du train de forage qui inclue un ensemble d'outils et des contrôleurs électroniques. Les composants du BHA, du moins profond au plus profond sont présentés dans la figure 1 :

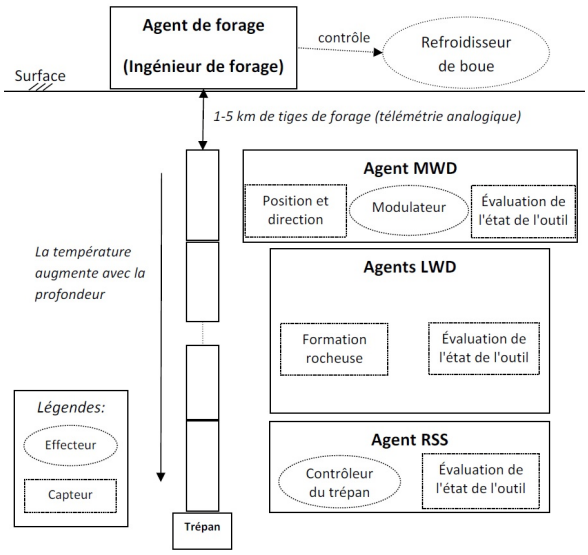


FIGURE 1 – Architecture multi-agent du SCP, adapté des travaux de Mualla et al. [3]

— **Technique de mesure pendant le forage (Measurements while Drilling, MWD) :** elle possède des capteurs qui mesurent la position et la direction de l'outil en utilisant le champs magnétique et gravitationnel de la terre. Il communique avec la surface en transmettant des données à travers un modulateur. Les méthodes de transmission varient, mais la plus courante consiste à envoyer des pulsations à travers le système de refroidissement par boue. Toutefois, le décodage du message en surface est grandement dépendant du rapport signal sur bruit, qui est affecté par les conditions de forage ;

— **Technique de journalisation pendant le forage (Logging while Drilling, LWD) :** elle est utilisée pour effectuer des mesures sur la formation rocheuse, sans arrêter le forage ;

— **Système de forage rotatif et orientable (Rotary Steerable System, RSS) :** il est utilisé pour diriger le BHA et lui appliquer une rotation par minute ;

— **Trépan de forage :** il est utilisé pour forer la formation rocheuse.

Tous les outils au sein du BHA ont des capteurs embarqués qui évaluent l'état de l'outil et mesurent les températures.

4 Modèle multi-agent du SCP de forage

Ce travail propose que les outils souterrains s'équipent des mécanismes permettant d'atténuer de manière autonome et collaborative les effets négatifs des hautes températures sur lesdits outils. Ces mécanismes s'appuient sur un processus de vote influencé par les spécifications des outils. Ce processus est mis en œuvre dans les intervalles existants entre les décisions de refroidissement de la part de l'ingénieur de forage en surface. Cette approche est rendue possible par les capacités des équipements récents qui permettent une communication entre les outils. Ainsi, ils mesurent et enregistrent toutes les données qu'ils transfèrent au MWD pour les envoyer à surface à l'aide d'un **modulateur**. De plus, le **contrôleur du trépan** est aussi un effecteur souterrain dans le RSS. Il est responsable du contrôle de la rotation du trépan, ce qui affecte la vitesse de forage. Cette réduction de la vitesse de forage retarde la montée de température des outils. Bien que cela cause une durée de forage plus longue, l'instant où les outils atteindront un état critique est également retardé et potentiellement évité. Ainsi, les dommages subis par les outils seraient limités et permettent à ces derniers de creuser plus profondément.

La section 4.1 offre une analyse des mécaniques et des paramètres du forage. La section 4.2 présente l'architecture de vote multi-agent. De plus, les caractéristiques et les modèles de décision des agents sont précisés.

4.1 Modèle des mécaniques de forage

Dans la littérature du domaine de l'industrie pétrolière et gazière, il n'y a pas de modèle concret des opérations de forage [22]. Le but principal de notre modèle est de représenter fidèlement les mécaniques de forage nécessaires pour calculer la vitesse de forage et la montée de la température pendant le forage.

Dans notre travail antérieur [3], nous avons défini une terminologie de forage

— **Profondeur mesurée (Measured Depth, MD) :** longueur du trou mesurée par des accéléromètres ;

— **Profondeur verticale réelle (True Vertical Depth, TVD) :** distance verticale entre la surface et le trépan. La TVD est particulièrement importante pour déterminer la température souterraine. Elle est calculée à partir de la MD et de l'angle de forage ;

— **Gradient de température :** taux de croissance de la température par unité de profondeur ;

2. Le premier auteur de ce manuscrit a cinq ans d'expérience dans l'industrie pétrolière et gazière.

— **Paramètres de forage :** définis par l'ingénieur forage en surface et utilisés pour contrôler le processus de forage. Trois paramètres cruciaux sont considérés dans notre modèle :

1. **Force :** représentée par le poids sur le trépan (Weight-on-Bit, WOB), elle est appliquée au train de forage, et est contrôlée uniquement en surface.
2. **Rotation :** représentée en Rotations-par-Minute (RPM) du train de forage, elle est contrôlée en surface et peut aussi être modifiée par les outils souterrains dans notre SCP.
3. **Débit de boue :** vitesse du flux de la boue qui circule dans le train de forage pour refroidir les outils et emporter jusqu'à la surface les éclats créés par le forage.

Selon Osgouei and Özbayoğlu [23], la vitesse de forage (ou taux de pénétration, Rate of Penetration, ROP) est définie par l'équation 1 .

$$ROP = K\frac{\overline{WOB}^K}{a^p}r \tag{1}$$

Où K est une constante liée à la dureté de la formation ; $\overline{WOB}$ est une fonction de WOB (équation 2) ; r est une fonction du RPM (équation 3) ; a^P est une fonction du débit de boue (équation 4).

Dans cette section, les fonctions $\overline{WOB}$, r et a^P sont détaillées à partir des propositions faites par Osgouei and Özbayoğlu [23]. Les caractéristiques du trépan ne sont pas abordées étant données qu'elles ne varient pas pendant le forage. Dans l'équation 2, d_b est le diamètre du trépan (en pouce) qui ne varie pas pendant le forage.

$$\overline{WOB} = \frac{7.88WOB}{d_b} \tag{2}$$

La fonction r (équation 3) est mesurée en cycle/minute. Les facteurs α et β permettent de modéliser la réaction du train de forage face à des formations rocheuses dures ($\alpha = 0.428$ et $\beta = 0.2$) et des formations rocheuses tendres ($\alpha = 0.75$ et $\beta = 0.5$).

$$r = \mathrm{e}^{\frac{-100}{RPM^2}}\, RPM^{\alpha} + \beta RPM(1 - \mathrm{e}^{\frac{-100}{RPM^2}}) \tag{3}$$

Dans l'équation 4, S est le nombre de coups de pompe, P est le débit de la pompe, et c est une constante reliée à l'efficacité de la pompe.

$$a^P = S\frac{P}{c} \tag{4}$$

4.2 Architecture multi-agent du SCP et coopération par votes

Dans notre SCP, un SMA est utilisé pour représenter les entités collaborantes d'un puits de forage au sein d'un système semi-autonome, en surface et sous terre (cf. figure 1).

Dans la partie inférieure de la figure 1 représentant l'espace sous-terrain, chaque outil programmable dans le BHA (MWD, LWDs et RSS) est représenté par un agent. Chacun de ces agents est responsable du contrôle des capteurs et des effecteurs embarqués dans l'outil. Chaque outil a des capteurs utilisés pour des mesures différentes : données de direction et de pilotage (MWD), de réparation et d'état (tous les outils), et de journalisation des caractéristiques de la formation (LWDs). Seul l'agent MWD contrôle le modulateur qui est responsable de la communication avec l'ingénieur de forage. De la même manière, seul l'agent RSS contrôle le trépan et sa rotation, en fonction des résultats des processus de vote.

Dans la partie supérieure de la figure 1 représentant la zone en surface, l'agent de forage représente l'ingénieur de forage. Cet agent est responsable du contrôle du processus de forage en ajustant les paramètres de forage en fonction des besoins afin d'atteindre la profondeur désirée. Additionnellement, il contrôle l'effecteur en surface, c'est-à-dire le refroidisseur de boue. Quand la boue refroidie redescend dans le puits et atteint le train de forage, elle est en contact direct avec tous les outils souterrains. Ainsi, elle refroidit les outils et transporte les déblais résultant du forage jusqu'à la surface.

L'intérêt de réduire la rotation du trépan est de ralentir la descente de l'assemblage et par conséquent de réduire l'augmentation de température. En parallèle, les outils souterrains envoient des requêtes à l'ingénieur de forage à travers le MWD afin d'augmenter la puissance fournie au refroidisseur de boue, afin de permettre un refroidissement ultérieur des outils. Cependant, entre-temps, les outils doivent prendre des mesures immédiates. Lorsqu'un outil atteint une température élevée et doit ralentir le perçage, il ne peut pas simplement demander au RSS de le ralentir car cela affectera tout le

système. Par conséquent, il doit commencer un cycle de vote pour obtenir l'opinion de tous les autres outils du système. Cela permettra une décision collective garantissant le bon fonctionnement de l'ensemble du système. Bien entendu, l'exécution de la décision est effectuée uniquement par le RSS qui contrôle l'actionneur.

Les agents du système proposé ont plusieurs propriétés : (i) autonomie, (ii) décentralisation, (iii) réactivité, (iv) flexibilité, et (v) sociabilité. Le lecteur intéressé pourra se référer à [4] pour une description détaillée de ces propriétés.

Modèle décisionnel des agents outils.

La température d'un outil à un instant donné est déterminée par la profondeur et le gradient géothermique [24]. C'est la donnée d'entrée du modèle de décision de l'agent outil. Trois niveaux de température sont définis pour chaque outil :

— **Niveau dangereux** : À l'atteinte de cette température, l'outil émet une requête auprès du MWD afin que le processus de refroidissement de la boue démarre depuis la surface.

— **Niveau critique** : Étant donné que la température à ce niveau approche la température critique, lorsqu'il détecte cette température l'agent réagit en demandant un vote entre les outils du BHA pour démarrer le contrôleur du trépan.

— **Niveau d'arrêt d'urgence** : L'outil est considéré comme définitivement endommagé et ne peut plus être utilisé.

Le modèle de décision détermine le niveau d'utilisation du contrôleur du trépan. C'est un nombre compris entre 0 et 100 représentant le pourcentage de la puissance du contrôleur du trépan. La décision d'activer le contrôleur du trépan ou de sélectionner le pourcentage de puissance est basée sur la nécessité de minimiser la consommation énergétique. En d'autres termes, si un pourcentage de puissance de 40% est suffisant pour contrôler la montée de température, le choix d'un pourcentage plus élevé doit être évité. La fonction utilisée pour déterminer la puissance désirée est formulée par l'équation 5 :

$$\frac{MaxP * (CR - TSL - TT)}{CR} \quad (5)$$

Où $MaxP$ est la puissance maximale du contrôleur de trépan ; CR est la soustraction de la moyenne des niveaux critiques de tous les outils avec la moyenne des niveaux d'arrêt d'urgence de tous les outils. Cette valeur est constante pour un forage, car les niveaux de température (spécifications de température) sont déterminés lors de la conception et de la fabrication de l'outil ; TSL est le niveau d'arrêt d'urgence de l'outil actuel ; TT est la température actuelle de l'outil.

La puissance désirée du contrôleur de trépan est utilisée pour déterminer la réponse définitive du modèle. La valeur calculée sera la valeur utilisée par l'agent dans le processus de vote. Le but ultime est d'atteindre un équilibre entre deux alternatives : garder un ROP élevé, et préserver l'intégrité des outils en régulant les températures.

5 Évaluation et résultats

Selon Al Meshabi et al. [25], l'un des défis rencontrés par les applications dans l'industrie pétrolière et gazière est la difficulté à gérer les éléments qui opèrent sur le terrain. Un environnement de simulation offre une alternative pratique pour tester de telles applications. Par conséquent, le SCP proposé est évalué en utilisant le simulateur AgentOil [22] qui est implémenté en utilisant Repast Simphony [26], qui est un environnement de simulation à base d'agents. Le choix de cet environnement est basé sur une comparaison des différents environnements de simulation qui montrent que RePast Simphony a des avantages opérationnels significatifs [27]. Le modèle des mécaniques de forage est implémenté dans cet environnement, ainsi que les modèles de décision et le système de vote.

La section 5.1 indique les paramètres initiaux utilisés durant les expérimentations. La section 5.2 analyse en détail les deux parties de notre SCP : le vote et le refroidissement, sur toute la durée de la simulation.

5.1 Initialisation et paramètrage

Une exécution de la simulation représente tout le processus de forage dans une application réelle, où le BHA est préparé en surface pour forer jusqu'à une profondeur précise avant d'avoir besoin de remonter en surface pour remplacer les vieux outils du BHA par de nouveaux. Tous les paramètres de simulation ont des valeurs initiales qui peuvent être changées par l'utilisateur au début de la simulation.

Une fois que la simulation a démarré, tous les agents sont créés dans l'environnement. Un pas

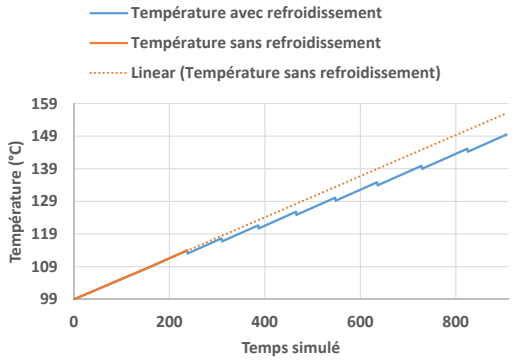


FIGURE 2 – Efficacité du refroidissement : augmentation des températures au cours du temps, avec et sans refroidissement

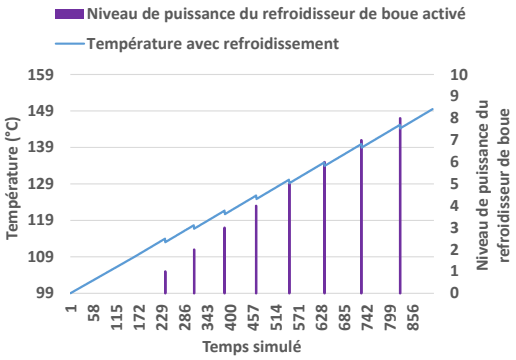


FIGURE 3 – Efficacité du refroidissement : impact du niveau de puissance du refroidisseur de boue sur les températures

de la simulation correspond à une minute afin de permettre la normalisation des résultats. À chaque pas, les paramètres de forage sont mesurés (équations 2, 3 et 4) et le ROP est calculé (équation 1). Le BHA exécute le forage et la température augmente avec la profondeur. Une fois que les outils atteignent une certaine température (caculée à partir de la TVD et du gradient géothermique), le modèle de décision de chaque outil est exécuté et les demandes de démarrage des effecteurs (en surface et en profondeur) sont envoyées de manière à atténuer les effets négatifs des hautes températures. Une simulation peut soit se terminer par une réussite (atteinte de la profondeur recherchée), soit par un échec (un outil est tombé en panne). Dans les deux cas, un historique complet de la simulation est fourni.

Pour normaliser les résultats, le même nombre d'agents est utilisé pendant toute la simulation : en surface, un seul agent représente l'ingénieur de forage, et en profondeur : un agent pour le MWD, trois agents pour les différents LWDs et un agent pour le RSS.

5.2 Éfficacité des mécanismes de refroidissement et de vote

Les comportements en temps réel des composants du SCP proposé sont présentés et analysés dans cette section, ainsi que leur impact direct dans l'atténuation de la température.

Il existe deux mécanismes dans notre SCP pour atténuer les hautes températures. Premièrement, le refroidisseur de boue est contrôlé par l'ingénieur de forage selon les requêtes des outils souterrains envoyées à l'agent de surface. Ensuite, en utilisant le contrôleur du trépan en profondeur pour ralentir le forage. Ce dernier est contrôlé par le RSS en fonction du résultat du vote de tous les agents du BHA. Le vote d'un agent est basé sur son modèle de décision interne, qui prend en compte la température actuelle et les spécifications de l'outil. Dans ce contexte, l'expérimentation est découpée en deux parties. Tout d'abord, pour prendre en compte l'impact du mécanisme de refroidissement, le système de vote est inhibé. Ensuite, la première partie de l'expérimentation est répliquée en activant le système de vote et en désactivant le refroidissement depuis la surface. De plus, toutes les exécutions sont effectuées avec un WOB fixe : 10k lbf qui est une valeur moyenne pour les opérations de forage.

Pour toutes les expériences, un graphique représentant la température d'un outil au cours de la simulation est construit. L'axe des abscisses représente le temps écoulé (en pas de simulation), et l'axe des ordonnées représente la température (en °C). Pour l'exécution présentée, la position de départ est de 3000m, et la température correspondant à cette profondeur est de 99°C.

Impact du refroidissement sur les températures.

Pour le cas où le mécanisme de vote est désactivé, la figure 2 compare l'évolution de température avec le refroidissement activé (courbe bleue), et avec l'évolution des températures quand le refroidissement est éteint (courbe rouge), lorsque la température augmente de façon linéaire.

Depuis le début de la simulation, la température augmente avec le forage. Notamment, la courbe bleue commence à décroître par rapport à la courbe rouge, lorsque le refroidissement s'active au pas 238, ce qui correspond à la tem-

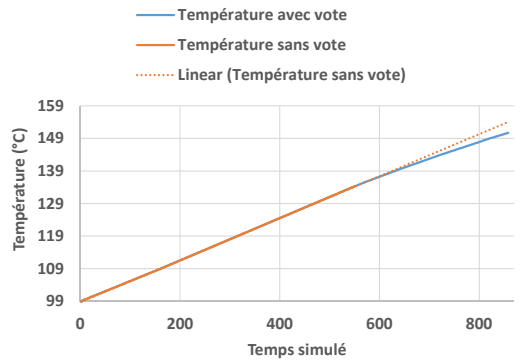


FIGURE 4 – Efficacité du vote : évolution des températures au cours du temps, avec et sans mécanisme de vote

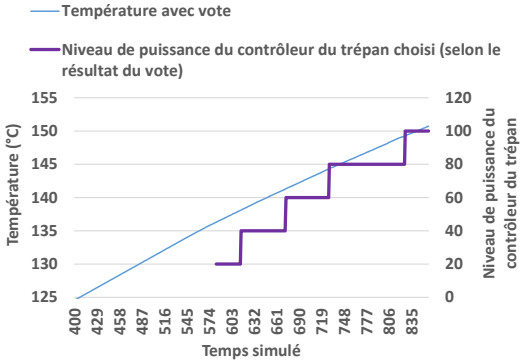


FIGURE 5 – Efficacité du vote : impact du niveau de puissance du contrôleur du trépan sur les températures

pérature de 113°C.

Dans la figure 3, l'axe des ordonnées représente la puissance fournie au refroidisseur de boue. Par moments, les outils souterrains requêtent l'activation du refroidisseur de boue en surface, lorsqu'ils atteignent leur niveau de température dangereuse. Une fois que le refroidisseur de boue n'est plus alimenté, la température chute (premier effet au pas 238). Il est important de se souvenir qu'il y a un délai entre les requêtes de refroidissement, étant donné le délai pour que le message atteigne la surface et que la boue refroidie atteigne le BHA. Ceci explique l'intervalle de temps entre deux chutes de température.

Impact du mécanisme de vote sur les températures.

Pour le cas où le mécanisme de refroidissement par boue est désactivé, la figure 4, représente la courbe de température avec vote (courbe bleue) et la courbe de température sans vote (courbe rouge). Les résultats montrent une chute de la tendance des températures grâce au mécanisme de vote.

La figure 5 fournit une représentation graphique plus détaillée sur la période $400 - 850$, c'est-à-dire approximativement à l'instant avant la chute de la tendance des températures. Dans cette figure, l'axe vertical à droite représente le niveau d'alimentation du contrôleur de trépan sélectionné par le vote. Le vote commence exactement au pas 583, où la décision de démarrer l'effecteur avec un niveau de puissance de 20% est prise. À la fin de l'expérimentation, et avant d'atteindre 150°C, le niveau de puissance maximale du contrôleur de trépan est utilisé, et après cela plus aucune atténuation n'est effectuée jusqu'à la panne d'un outil.

Il est important de noter que le mécanisme de vote (au niveau de la température critique) démarre après avoir déclenché le système de refroidissement (au niveau de la température dangereuse), car le refroidissement met plus de temps à agir que le mécanisme de vote.

5.3 Évaluation de l'efficacité du SCP proposé

Les figures 6 et 7 montrent respectivement les résultats en termes de temps et de profondeur en utilisant la simulation avec le système proposé (courbes colorées) et sans notre proposition (courbe grise). La figure 6 illustre le temps avant la panne d'un outil (en d'autres termes, combien de temps les outils ont survécu aux hautes températures). De même, la figure 7 montre la profondeur atteinte avant la panne (en d'autres termes, le niveau de profondeur foré avant la panne), ce qui constitue l'objectif principal d'un puits de forage. Dans les deux figures, trois règles de vote sont examinées : Condorcet (en vert), Pluralité (en rouge), et méthode Borda (en bleu).

La figure 6 permet de remarquer que toutes les courbes colorées sont au dessus de la courbe grise, ce qui signifie que tous les forages (avec différents systèmes de vote) sont plus efficaces que le forage sans le système proposé. Toutefois, ce résultat doit être mitigé car plus le temps de forage est grand, plus le délai pour atteindre l'objectif est long, et plus le coût financier de l'opération de forage est important. Il convient également d'analyser la figure 7 qui représente la profondeur avant la panne. Elle montre que les courbes colorées sont en dessous de la courbe grise (tendance accentuée avec un WOB élevé). Par exemple, la courbe verte (règle de Condor-

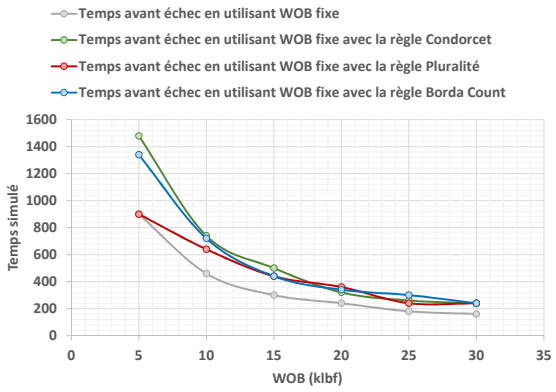


FIGURE 6 – Gains obtenus au cours du temps avec le système proposé

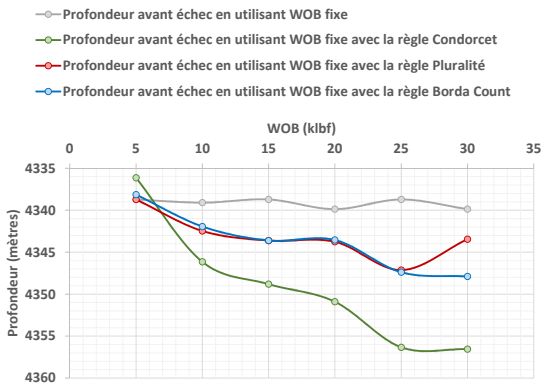


FIGURE 7 – Gains en profondeur avec le système proposé

cet) a atteint approximativement 4356m avec un WOB de 30k lbf, ce qui représente 16m de plus que le forage sans le système proposé (courbe grise). Cette différence est vitale dans des conditions de forage extrèmes car forer 16m aussi profondément prend plusieurs heures, et chaque heure correspond à un coût financier important.

6 Conclusion et Perspectives

Dans les opérations de forage pour la recherche du gaz et du pétrole, la température s'accroît avec la profondeur et par conséquent endommage les outils de forage. Pour corriger ce problème, cet article a proposé un SCP où les agents sont utilisés pour représenter les entités collaboratives qui contrôlent les capteurs et les effecteurs, à la fois en surface et en profondeur. Dans le SCP proposé, le processus qui décide de demander le refroidissement en surface à un ingénieur de forage ou de démarrer un vote pour démarrer l'effecteur souterrain est gouverné par le modèle de décision de chaque agent. Les résultats démontrent que notre SCP atténue la croissance des hautes températures avec la combinaison des mécanismes du vote et du refroidissement.

Le SCP proposé est principalement utile dans le forage de puits profonds et chauds, ou avec des conditions de forage difficile, où le signal analogique entre les outils souterrains et la station de forage en surface peut être bruyant ou perdu, les mesures des capteurs ne sont plus transmises fidèlement à la surface. Nous pensons que notre modèle n'offre pas de bénéfices significatifs dans le forage de puits peu profonds, et donc moins chauds.

Comme perspectives, nous pensons que l'« Explainable Artificial Intelligence » (XAI) peut jouer un rôle significatif dans notre modèle. Étant donné que nous opérons dans un système semi-autonome dans lequel l'ingénieur de forage possède un rôle moins complexe mais toujours décisionnel, définir un moyen de communication entre les humains et les autres entités du SCP (outils, capteur et effecteurs) est une clé de la compréhension des décision du SCP par l'ingénieur. De plus, comme certains outils sont plus importants que d'autres au vu des rôles qu'ils jouent, nous planifions d'étendre notre modèle pour prendre ce fait en compte et implémenter un système de vote pondéré [28]. En effet, le vote pondéré permettra à un agent de diviser son vote parmi les choix qui lui sont offert afin d'optimiser d'avantage le processus de prise de décision. Différents scénarios qui correspondent à différentes situations seront considérés. Par exemple, une distinction doit être faite entre le forage « offshore » et le forage terrestre, car dans le premier type de forage, les outils sont plus difficiles à obtenir, mais ils sont aussi plus avancés pour supporter de plus hautes chaleurs (avec un coût plus important).

Remerciements

Ce travail est supporté par le Conseil Régional de Bourgogne France-Comté dans le cadre du projet UrbanFly 20174-06234/06242. Ce travail est partiellement supporté par le programme Wallenberg AI, Autonomous Systems and Software Program (WASP) financé par la fondation Knut et Alice Wallenberg.

Références

[1] International Energy Agency, "World energy outlook 2018," https ://www.iea.org/weo2018/, 2018, accessed : 2018-11-27.

[2] G. DeBruijn, C. Skeates, R. Greenaway, D. Harrison, M. Parris, S. James, F. Mueller, S. Ray, M. Riding, L. Temple, and K. Wutherich, "High-pressure, high-temperature technologies," in **Oilfield Review**, vol. 20. Schlumberger, 2008.

[3] Y. Mualla, A. Najjar, R. Vanet, O. Boissier, and S. Galland, "Towards a real-time mitigation of high temperature while drilling using a multi-agent system," in **1st International Workshop on Real-Time Compliant Multi-Agent Systems**, vol. 2156. RTcMAS 2018, 2018, pp. 77–92.

[4] Y. Mualla, A. Najjar, O. Boissier, S. Galland, I. Haman Tchappi, and R. Vanet, "A cyber-physical system for semi-autonomous oil&gas drilling operations," in **5th Workshop on Collaboration of Humans, Agents, Robots, Machines and Sensors (CHARMS 2019)**, 2019, to appear.

[5] R. Baheti and H. Gill, "Cyber-physical systems," **The impact of control technology**, vol. 12, no. 1, pp. 161–166, 2011.

[6] J. Shi, J. Wan, H. Yan, and H. Suo, "A survey of cyber-physical systems," in **Wireless Communications and Signal Processing (WCSP), 2011 International Conference on**. IEEE, 2011, pp. 1–6.

[7] B. H. Krogh, "Cyber physical systems : the need for new models and design paradigms," **Presentation Report**, 2008.

[8] J. Lee and E. Lapira, "Predictive factories : the next transformation," **Manufacturing Leadership Journal**, vol. 20, no. 1, pp. 13–24, 2013.

[9] J. Lee, B. Bagheri, and H.-A. Kao, "A cyber-physical systems architecture for industry 4.0-based manufacturing systems," **Manufacturing Letters**, vol. 3, pp. 18–23, 2015.

[10] Y. Wadhawan and C. Neuman, "Evaluating resilience of oil and gas cyber physical systems : A roadmap," in **Annual Computer Security Application Conference (ACSAC) Industrial Control System Security (ICSS) Workshop**, 2015.

[11] A. Najjar, Y. Mualla, O. Boissier, and G. Picard, "AQUAMan : QoE-driven cost-aware mechanism for SaaS acceptability rate adaptation," in **Proceedings of the International Conference on Web Intelligence**. ACM, 2017, pp. 331–339.

[12] L. L. Mikkelsen and B. N. Jørgensen, "Application of multi-agent systems in offshore oil and gas production," in **Proceedings of IADIS Multi Conference on Computer Science and Information Systems**, 2012, pp. 158–163.

[13] V. L. C. de Oliveira, A. P. M. Tanajura, and H. A. Lepikson, "A multi-agent system for oil field management," **IFAC Proceedings Volumes**, vol. 46, no. 7, pp. 35–40, 2013.

[14] J. Barbosa, P. Leitão, E. Adam, and D. Trentesaux, "Dynamic self-organization in holonic multi-agent manufacturing systems : The adacor evolution," **Computers in Industry**, vol. 66, pp. 99–111, 2015.

[15] W. Gaertner, **A primer in social choice theory : Revised edition**. Oxford University Press, 2009.

[16] A. D. Procaccia and J. S. Rosenschein, "Junta distributions and the average-case complexity of manipulating elections," in **Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems**. ACM, 2006, pp. 497–504.

[17] J. Lang, "Logical preference representation and combinatorial vote," **Annals of Mathematics and Artificial Intelligence**, vol. 42, no. 1, pp. 37–71, 2004.

[18] J. Von Neumann and O. Morgenstern, **Theory of games and economic behavior**. Princeton university press, 2007.

[19] F. Brandt, V. Conitzer, and U. Endriss, "Computational social choice," **Multiagent systems**, pp. 213–283, 2012.

[20] F. Rossi, K. B. Venable, and T. Walsh, "A short introduction to preferences : between artificial intelligence and social choice," **Synthesis Lectures on Artificial Intelligence and Machine Learning**, vol. 5, no. 4, pp. 1–102, 2011.

[21] Y. Shoham and K. Leyton-Brown, **Multiagent systems : Algorithmic, game-theoretic, and logical foundations**. Cambridge University Press, 2008.

[22] Y. Mualla, R. Vanet, A. Najjar, O. Boissier, and S. Galland, "AgentOil : a multiagent-based simulation of the drilling process in oilfields," in **Demazeau Y., An B., Bajo J., Fernández-Caballero A. (eds) Advances in Practical Applications of Agents, Multi-Agent Systems, and Complexity : The PAAMS Collection**, ser. Lecture Notes in Computer Science, vol. 10978. Springer, Cham, 2018.

[23] R. E. Osgouei and M. Özbayoğlu, "Rate of penetration estimation model for directional and horizontal wells," in **16th International Petroleum and Natural Gas Congress and Exhibition of Turkey**, 2007.

[24] M. J. Jellison, D. R. Hall, D. C. Howard, H. T. Hall Jr, R. C. Long, R. B. Chandler, D. S. Pixton **et al.**, "Telemetry drill pipe : enabling technology for the downhole internet," in **SPE/IADC drilling conference**. Society of Petroleum Engineers, 2003.

[25] O. O. Al Meshabi, M. M. Khazindar, H. Orenstein **et al.**, "Attitude of collaboration, real-time decision making in operated asset management," in **SPE Intelligent Energy Conference and Exhibition**. Society of Petroleum Engineers, 2010.

[26] N. Collier, "Repast : An extensible framework for agent simulation," **The University of Chicago's Social Science Research**, vol. 36, 2003.

[27] Y. Mualla, W. Bai, S. Galland, and C. Nicolle, "Comparison of agent-based simulation frameworks for unmanned aerial transportation applications," in **7th International Workshop on Agent-based Mobility, Traffic and Transportation Models, Methodologies and Applications**. Procedia Computer Science, Elsevier, 2018.

[28] G. W. Cox, **Making votes count : strategic coordination in the world's electoral systems**. Cambridge University Press, 1997.

Modèle Multiniveau Dynamique basé sur la Densité: Application au Trafic Routier à Grande Échelle

Igor Haman Tchappi[a,b] igortchappi@gmail.com
Stéphane Galland[b] stephane.galland@utbm.fr
Yazan Mualla[b] yazan.mualla@utbm.fr
Amro Najjar[d,e] najjar@cs.umu.se
Vivient Corneille Kamla[c] vckamla@gmail.com
Jean Claude Kamgang[c] jckamgang@gmail.com

[a]Faculty of Sciences, University of Ngaoundere, Ngaoundere, BP : 454, Cameroon

[b]CIAD, Univ. Bourgogne Franche-Comté, UTBM, F-90010 Belfort, France

[c]ENSAI, University of Ngaoundere, Ngaoundere, BP : 454, Cameroon

[d]Umea University, 901 87 Umea, Sweden

[e]AI-Robolab/ICR, Computer Science and Communications, University of Luxembourg, Luxembourg

Résumé

De nos jours, avec l'émergence d'objets et de voitures connectés, les systèmes de trafic deviennent de plus en plus complexes et présentent des comportements hiérarchiques à plusieurs niveaux d'observation. La plupart des modèles de simulation multiniveaux utilisent un ensemble de niveaux prédéfinis. La commutation dynamique des niveaux lors de l'exécution de la simulation permet d'adapter le modèle aux contraintes liées à la qualité des résultats ou aux ressources de calcul disponibles. Cet article présente un nouveau modèle multiniveau basé sur la densité pour la gestion dynamique d'une holarchie représentant un système de trafic. La proposition étend l'algorithme DBSCAN dans le contexte des systèmes multi-agents holonique. Une méthode de commutation dynamique entre les différents niveaux d'abstractions est proposée. À cette fin, des indicateurs multiniveaux basés sur l'écart type sont proposés afin d'évaluer la cohérence des résultats de la simulation. Le modèle proposé est testé avec le modèle de poursuite de voiture Intelligent Driver Model.

Mots-clés : *Système MultiAgent Holonique, Modélisation et simulation multiniveau, DBSCAN, Trafic*

Abstract

Nowadays, with the emergence of connected objects and cars, the road traffic systems become more and more complex and exhibit hierarchical and multilevel behaviors. Most of the multilevel simulation models use a set of predefined levels. Dynamic adaptation of the levels during the simulation execution enables to adapt the model to constraints related to the quality of the results or the available computing resources. This paper presents a novel density-based multilevel model for the dynamic management of an holarchy that is representing a road traffic system. The proposal extends the DBSCAN algorithm in the context of holonic multiagent systems. A method for switching dynamically between the different abstractions levels is proposed. To this purpose, multilevel indicators based on standard deviation are proposed in order to evaluate the consistency of the simulation results. The proposed model is tested with the Intelligent Driver car-following model on typical traffic study cases.

Keywords: *Holonic Multiagent System ; Multilevel Modeling and Simulation ; DBSCAN ; Road Traffic ; Intelligent Driver Model*

1 Introduction

La croissance de la population et l'augmentation du nombre d'usagers sur les routes constituent aujourd'hui une source considérable de problèmes divers d'ordre sécuritaire, sanitaire, économique, énergétique et environnemental. Pour faire face à ces défis, il est nécessaire de fournir des systèmes de transport efficaces et durables. La modélisation et la simulation des flux de trafic est une des solutions importantes pour apporter des réponses à la problématique de l'amélioration des conditions de circulation des biens et des personnes en termes de réglementation et d'infrastructure. Il existe trois principales approches de modélisation et de simulation du trafic routier [14] : l'approche microscopique, mésoscopique et macroscopique. Les modèles microscopiques sont précis mais ne passent pas à l'échelle [21, 14], tandis que les modèles macroscopiques passent à l'échelle mais présentent un comportement grossier [21, 14]. Les modèles multiniveaux (combinaison de plusieurs niveaux de détail au sein d'un même modèle) quant à eux semblent être une approche appropriée pour modéliser le trafic à grande échelle [21, 2, 30]. La plupart des modèles multiniveaux de flux de trafic issu la littérature [2, 21, 13] divise premièrement le réseau routier en sections, de telle sorte que chaque représentation (micro, méso ou macro) soit associée à une section du réseau routier, et deuxièmement manage la transition à la frontière entre ces représentations. Ces modèles multiniveaux de trafic sont statiques car

les connexions entre les niveaux d'abstraction sont fixées à priori et ne peuvent pas être modifiées au cours de l'exécution [1]. Cependant, pour pouvoir observer certains phénomènes émergents tels que la congestion, une approche de modélisation multiniveaux dynamique est nécessaire [1, 27]. Contrairement aux modèles de trafic multiniveaux statiques, cet article propose un modèle multiniveau dynamique de trafic au travers des Systèmes MultiAgents Holonique (SMAH)

Les SMAH ont été appliqués avec succès à un large éventail d'applications [11, 10, 9, 29, 8, 26]. Les organisations holoniques font partie des modèles organisationnels réussis qui ont été introduit dans les Systèmes MultiAgents (SMA) [11]. Un holon, selon Koestler [18], est défini simultanément comme un tout et une partie qui peut se composer d'autres holons. La modélisation holonique est utilisée pour modéliser la nature hiérarchique intrinsèque des systèmes. Un SMAH est donc une structure récursive de holons [11]. Un SMAH permet de modéliser des systèmes complexes en définissant plusieurs niveaux d'abstraction du système [11]. Les SMAH sont des approches appropriées pour la modélisation et la simulation multiniveaux [10] de problèmes associés à plusieurs niveaux de détail, tel que le trafic [29].

Pour effectuer la modélisation et la simulation multiniveaux du trafic routier à grande échelle, le papier adapte l'algorithme DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [6] à la modélisation et à la simulation multiniveaux du trafic via son extension holonique H-DBSCAN [31]. DBSCAN est un algorithme de partitionnement qui partitionne une base de données en clusters disjoints de sorte que les objets similaires appartiennent au même cluster. De même, H-DBSCAN partitionne une population d'agents en groupes disjoints de telle sorte que les agents proches appartiennent au même groupe. Le principal avantage de DBSCAN et de son extension holonique H-DBSCAN est qu'ils ne nécessitent pas de prédétermination du nombre de clusters à priori [6]. Cet avantage permet de créer un modèle multiniveau basé sur la densité, de sorte que la population d'agents représente le niveau inférieur et que les groupes d'agents trouvés par H-DBSCAN représentent le niveau supérieur. Pour évaluer notre modèle multiniveau basé sur la densité, une application est présentée et discutée. Le principal atout du modèle multiniveau proposé est sa dynamique, c'est-à-dire qu'il n'existe pas de représentation fixé à priori pour les sections de la route et les représentations peuvent évoluer au cours de l'exécution.

La suite de cet article est organisée comme suit : la Section 2 présente les SMAH et le principe des modèles multiniveaux de trafic. La section 3 présente le modèle multiniveaux dynamique basé sur la densité proposé pour le trafic. La section 4 présente une application, l'évaluation et les performances de la proposition. Enfin, la section 5 conclut cet article et donne les axes de recherche futurs.

2 Travaux Connexes

2.1 Les Systèmes MultiAgents Holoniques

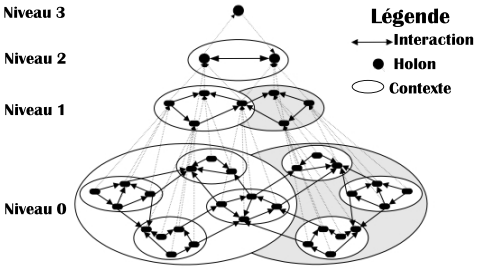


FIGURE 1 – Une holarchie imbriquée à quatre niveaux d'abstraction

Un SMAH est type particulier des SMA [22, 7, 15, 16] composé des holons. Le terme "holon" a été inventé par le philosophe hongrois Arthur Koestler en le basant sur le mot grec "holos" signifiant "tout" et le suffixe "-on" qui signifie "partie" [18]. Un holon est une structure fractale stable, cohérente et constituée de plusieurs holons agissant en tant que sous-structures. Un exemple biologique est donné par Koestler : un être humain est constitué d'organes, les organes à leur tour sont constitués de cellules qui peuvent être décomposées davantage et ainsi de suite. Aucun de ces composants biologiques ne peut être compris sans ses sous-composants ou sans le super-composant dont il fait partie. Dans le contexte des SMAH, un holon est assimilé à un agent qui peut se composer d'autres agents. Les holons sont des groupes autonomes et autosimilaires, imbriqués ou non. L'auto-similarité des holons permet de construire les SMAH sur la base de la composition des holons. Un SMAH est une structure récursive de holons [11]. Une holarchie représentée par la figure 1 est un schéma d'interconnexion entre les holons. Un SMAH permet deux types de communication [26] : (i) communication intra-niveau : holons communiquant avec d'autres holons au même niveau d'abstraction ("communication horizontale"), et (ii) communication inter-niveaux : lorsque des holons de deux niveaux différents communiquent (" communication verticale").

	SMA	SMAH
Approche de conception	réductionnisme	holisme (hiérarchique et récursif)
Composition	agent atomique	holons (agent composé d'autres agents)
Représentation	Fine	Fine et grossière
Propriété emergente	Oui	Oui
Conception	Hétérogénéité des agents	Homogénéité des holons
Mise à l'échelle	Non	Oui

TABLE 1 – Différences principales entre MAS et SMAH

La table 1 met en évidence les principales différences entre les MAS et les SMAH. Les SMA exhibent un comportement proche du comportement réel du système mais ne passe pas à l'échelle [25, 14]. Par contre, les SMAH peuvent exhiber un comportement grossier mais passent à l'échelle [10]. Pour modéliser et simuler des systèmes complexes à grande échelle, il faut gérer un compromis entre la précision de la simulation et la disponibilité des ressources computationnelles [25, 10]. Comme indiqué dans la Table 1, les SMAH semblent être approprié pour modéliser et simuler un système complexe à grande échelle [26, 10]. Puisque que les SMAH passent à l'échelle [9, 11] et qu'ils ont été appliqué avec succès dans la modélisation et la simulation du trafic [30, 26], cet article se focalise sur les SMAH avec communication intra et inter-niveaux pour la modélisation et la simulation du trafic à grande échelle basée sur la densité.

2.2 Concepts Basiques de la Modélisation Multiniveaux du Flux du Trafic

Comme indiqué précédemment, les approches microscopiques, mésoscopiques et macroscopiques permettent la modélisation et la simulation du trafic respectivement par des représentations fines, grossières et très grossières. Aucune approche n'est la meilleure car chacune d'elles possèdent ses avantages et ses inconvénients. En effet, les modèles microscopiques nécessitent un coût computationnel élevé et sont généralement appliqués avec une grande précision sur des petites zones urbaines, tandis que les modèles macroscopiques sont incapables de gérer la destination des véhicules et sont généralement appliqués sur des autoroutes avec un coût de calcul acceptable [30]. Les modèles multiniveaux ont été développés afin d'étudier les situations dans lesquelles les modèles microscopiques et macroscopiques ne conviennent pas, comme la modélisation et la simulation de trafic à grande échelle [21]. Les modèles multiniveaux combinent les avantages des modèles macro et micro, mais ils sont difficiles à réaliser [30]. En effet, comme le flux de trafic est modélisé par la vitesse, la longueur, l'accélération, la position, etc. des véhicules atomiques au niveau micro, tandis qu'il est généralement modélisé de manière macroscopique par la densité, le flux et la vitesse de l'ensemble du trafic, la gestion de l'intégration de ces deux représentations au sein d'un même modèle est une tâche difficile [30]. Plusieurs modèles multiniveaux de trafic ont été proposés dans la littérature [2, 21, 13]. La méthode la plus couramment proposée pour gérer la complexité des modèles multiniveaux de trafic est la division du réseau routier en parties de sorte que chaque niveau de détail soit associé à une partie du réseau routier. Le principe d'hybridation consiste donc à gérer la transition entre ces niveaux de détail à la frontière du réseau routier par l'agrégation/désagrégation [21] par exemple. La figure 2 présente la méthodologie courante des modèles multiniveaux de trafic.

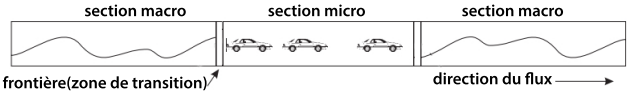


FIGURE 2 – Principe des modèles multiniveaux statiques [2]

Cette méthodologie possède plusieurs inconvénients :

— les niveaux de détail à interfacer sont choisis à priori soit micro-macro, micro-méso ou méso-macro. De plus, ces niveaux sont généralement deux. Nous soutenons qu'il peut être intéressant de prendre en compte plus de deux niveaux de détail notamment les représentations micro, méso et macro au sein d'un modèle multiniveau de trafic.

— manque de flexibilité et de réutilisabilité. En fait, les résultats du schéma de couplage macro-micro avec le modèle [19] de Lighthill et Whitham (LWR) pour la section macro et le modèle de vitesse optimale [24] pour la section micro présentée par Bourrel and Lesort [2], par exemple, ne peut pas être facilement étendu ni réutilisé pour modéliser un nouveau schéma de couplage du modèle LWR et du modèle Intelligent Driver Model (IDM) [32].

— le modèle multiniveau est statique et les niveaux de détail sont fixés à priori sur des sections routière et cette configuration ne peut pas évoluer au cours du temps. Cependant, pour pouvoir observer la formation de congestion ou pour trouver l'emplacement exact d'un bouchon dans une section macro, une approche de modélisation hybride dynamique est nécessaire [1].

Navarro et al. [23], Gaud et al. [10] ont proposé un modèle dynamique et multiniveau très intéressant. Cependant, ce modèle est dédié à la mobilité des piétons. Il existe très peu de travaux consacrés à la modélisation dynamique multiniveau du trafic routier [1, 12]. Tchappi et al. [30] dans leur revue identifie seulement deux travaux dans la littérature proposant de basculer dynamiquement et automatiquement entre les représentations du système : les travaux de l'équipe de H. Abouaïssa [12, 1] et les travaux de Sewall et al. [27]. Hassane et al. [12] propose SIMILAR/JAM-FREE pour gérer la modélisation dynamique multiniveau du trafic routier. SIMILAR est un framework utilisant le principe influence-réaction et la technologie à base d'agent pour la conception des simulations. JAM-FREE est un simulateur multiniveau de trafic à base d'agent. L'objectif des auteurs était de modéliser de manière dynamique et automatique le trafic multiniveau. L'idée et l'approche poursuivie par ces auteurs est très intéressante. Cependant, le modèle est très peu décrit,

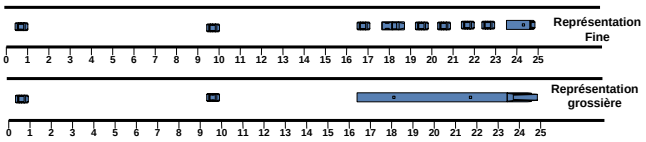


FIGURE 3 – Représentation fine/grossière (niveau inférieur/niveau supérieur)

plusieurs parties du modèle sont inachevées et le projet semble abandonné.

Sewall et al. [27] proposent un algorithme en temps réel pour la modélisation et la simulation hybride du trafic à grande échelle en utilisant à la fois une représentation continu et une représentation agent. Les auteurs proposent des techniques de couplage dynamique entre une représentation de véhicules discrets et une représentation continue. Ce travail est également intéressant, cependant, il se focalise sur la visualisation.

Cet article propose un modèle multiniveau dynamique et automatique du flux de trafic via les SMAH. Les exigences du modèle proposé sont :

— prendre en compte plusieurs niveaux de détail (micro, méso, macro) ;

— relier plusieurs niveaux de détail (micro, méso, macro) et permettre ainsi de faire ressortir les propriétés émergentes et la relation qui existe entre les niveaux d'abstraction ;

— gérer les transitions dynamique entre les niveaux de détail (micro, méso et macro) ;

— Pas de représentation fixe dédiée à priori pour un niveau particulier de détail ;

— migration dynamique (manque de ressources de calcul, visualisation, besoins de l'utilisateur, contraintes, etc.) entre les niveaux de détail (micro, méso et macro) ;

3 Modèle Multiniveau Dynamique basé sur la Densité

Cette section présente notre modèle dynamique multiniveau basé sur la densité pour gérer la simulation au cours du temps. L'idée du modèle proposée est inspirée du clustering de flux de données [28]. En effet, étant donné qu'un flux massif de données d'objets arrivent en permanence et à grande vitesse par des capteurs, il devient impossible de stocker toutes ces données en mémoire et même sur le disque dur [28]. De plus, le temps de traitement limité est un défi important pour les algorithmes de clustering de flux évolutif de données. Un moyen courant de résoudre ces défis consiste à conserver les résumés sur les données au lieu des données brutes d'origine et à les utiliser pour le clustering [4, 20]. Cette idée fonctionne assez bien pour clustering de flux évolutif de données. Notre proposition adapte cette

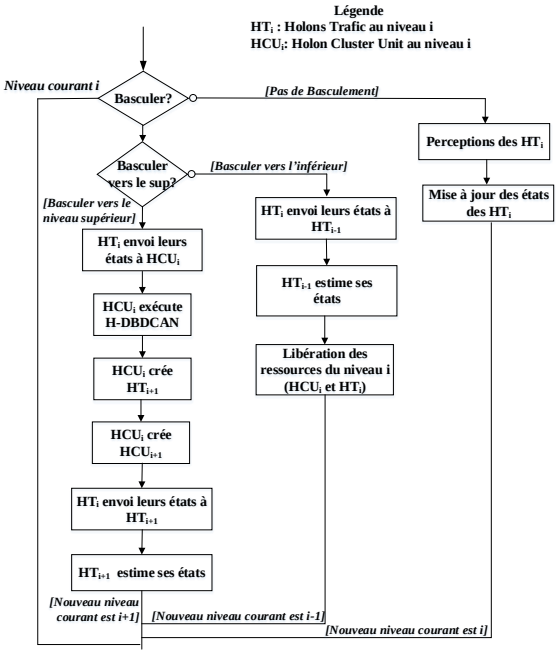


FIGURE 4 – Simulation Dynamique du comportement des holons

idée à la modélisation et à la simulation multiniveau à grande échelle du trafic. En effet, la simulation est effectuée à plusieurs niveaux d'abstraction, tel que le niveau micro représente des individus atomiques tandis que les niveaux supérieurs peuvent représenter des groupes d'individus. En d'autres termes, pour modéliser et simuler le trafic multiniveaux dynamique à grande échelle, certains véhicules de la représentation fine sont regroupés pour obtenir une représentation grossière. La figure 3 présente un exemple de l'idée de la proposition. Comme indiqué sur cette figure, la représentation fine affiche 9 véhicules et la représentation grossière affiche 3 véhicules tel que le groupe formé de 7 véhicules (véhicules en convoi) est représenté par un véhicule de moindre précision. Les véhicules en convoi sont des véhicules ayant à peu près la même vitesse, se déplaçant sur la même voie et ayant à peu près la même distance entre eux. Les véhicules en convoi mènent à un état stationnaire du trafic. Un état de trafic est stationnaire [14] quand il ne change pas au cours du temps (il conserve la même configuration spatiale).

Chaque entité atomique du système est représentée par un agent. Tous les agents représentent la population globale. Comme indiqué précédemment, un holon est assimilé à un agent qui peut être composé d'autres agents. Au niveau microscopique (niveau le plus précis), un agent est un holon non décomposé (atomique). Aux niveaux supérieurs, un holon est un groupe de holons, qu'il soit composé ou non. Un holon peut donc être considéré soit comme une entité individuelle, soit comme un ensemble de sous-holons. Cette dualité est généralement appelée

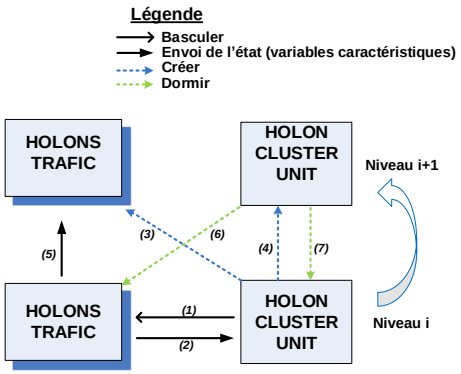


FIGURE 5 – Commuter vers une représentation grossière

effet Janus La proposition comporte deux types de holons :

— **Holons Trafic:** Les holons trafic sont des individus ou des groupes d'individus au sein du système de trafic. Les holons trafic peuvent être une paire véhicule/conducteur ou un groupe de paires véhicule/conducteur qui ont sensiblement la même vitesse, se déplacent sur la même voie et qui sont sensiblement équidistant.

— **Holons Cluster Unit:** Les holons cluster unit sont des holons particuliers créés à chaque niveau de la holarchie afin de simplifier le management de la holarchie au cours du temps. Le holon cluster unit n'affecte pas le comportement des holons trafic. Il est requis dans des circonstances particulières (lors d'une requête de commutation). Il y a toujours un holon cluster unit par niveau d'abstraction.

Les holons peuvent être simulés à plusieurs niveaux d'abstraction. La figure 4 décrit le modèle de comportement des holons. Il y a trois possibilités principales pour la simulation de holons :

— **Aucune commutation:** conserver la même représentation (statu-quo au niveau courant i). Sans requête de commutation, les holons trafic sont simulés au niveau courant i. Les holons trafic perçoivent l'environnement et mettent à jour leurs états à chaque pas de temps, comme indiqué dans la figure 4. Dans ce cas, la seule tâche du holon cluster unit est de vérifier si une requête de commutation vers le niveau supérieur ou inférieur est requise en fonction des insuffisances de ressources de calcul, de la visualisation, des contraintes, etc. Le nouveau niveau courant reste i et la simulation se poursuit au niveau i.

— **Commuter vers le niveau supérieur:** vers une représentation plus grossière (du niveau courant i vers le niveau $i+1$), comme illustré par la figure 5. Dans ce cas, le holon cluster unit au niveau i envoie une requête de commutation aux holons trafic au niveau i, comme indiqué par la flèche (1). Tous les holons trafic du niveau i envoient leurs états au holon cluster unit au niveau i indiqué par la flèche (2). Le holon cluster unit au niveau i rassemble les variables caractéristiques de tous les holons trafic au niveau i et applique l'algorithme H-DBSCAN [31] afin de construire le niveau $i+1$. H-DBSCAN est une extension de l'algorithme DBSCAN [6] que nous avons proposé dans nos travaux antérieurs. Comme DBSCAN, dans H-DBSCAN [31], la densité associée à un holon h est obtenue en comptant le nombre de holons dans une région d'un rayon spécifié autour du holon h (les voisins proches du holon h). Les holons dont la densité est supérieure à un seuil spécifié sont regroupés dans un super-holon situé au niveau immédiatement supérieur. En d'autres termes, le résultat de H-DBSCAN conduit le holon cluster unit à créer des super-holons trafic appropriés au niveau $i+1$ (indiqués par la flèche (3)) tels que les sous-holons trafic en convoi au niveau i soient regroupés dans un seul super-holon trafic au niveau $i+1$. Pour plus d'informations sur H-DBSCAN, le lecteur est renvoyé à [31]. Le holon cluster unit au niveau i crée également son propre super-holon cluster unit (holon cluster unit au niveau $i+1$) représentée par la flèche (4). Ensuite, les sous-holons trafic au niveau i envoient leurs états à leurs super-holons trafic respectifs au niveau $i+1$ indiqué par la flèche (5). Les super-holons trafic utilisent les états de leurs sous-holons trafic pour estimer leurs propres états. Tous les holons du niveau i deviennent inactifs, comme indiqué par la flèche (6-7), le nouveau niveau courant devient $i+1$ et la simulation se poursuit au niveau $i+1$. Notez que macro est le niveau le plus élevé : il n'est pas possible d'aller plus haut que le niveau macro.

— **Commuter vers le niveau inférieur:** vers une représentation plus fine (du niveau courant i vers le niveau $i-1$), comme illustré par la figure 6. Dans ce cas, le holon cluster unit au niveau i envoie une requête de commutation aux holons trafic au niveau i, comme indiqué par la flèche (1). Puisque tous les sous-holons du niveau $i-1$ sont inactifs, la première étape consiste à réinvoquer tous les sous-holons représentés par les flèches (2) et (3). Ensuite, chaque super-holon trafic au niveau i génère les états de ses sous-holons trafic selon une distribution probabiliste et envoie ces états à ses sous-holons trafic au niveau $i-1$ indiqué par la flèche (4). Étant donné que les niveaux supérieurs diminuent la précision de la simulation et ne sont requis que si nécessaire (besoins de l'utilisateur, manque de ressources de calcul, visualisation, etc.), les niveaux supérieurs sont libérés lors du passage vers un niveau inférieur indiqué par la flèche (5-6) . En effet, dans la mesure du possible, la

1. Janus est un ancien dieu romain décrit comme ayant deux visages, car il regarde à la fois vers le futur et vers le passé.

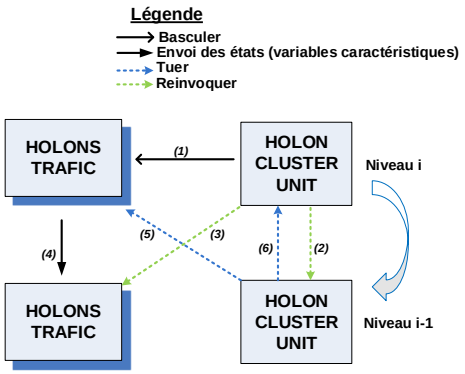


FIGURE 6 – Commuter vers une représentation plus fine

simulation a tendance à se dérouler autant que possible au niveau inférieur. Le nouveau niveau courant devient $i-1$ et la simulation se poursuit au niveau $i-1$. Notez que le niveau micro est le niveau le plus bas : il est impossible de descendre plus bas que le niveau micro.

Le modèle proposé est général et peut être adapté à plusieurs modèles de poursuite.

4 Application par le Modèle IDM (Intelligent Driver Model)

Le modèle d'accélération du holon trafic est basé sur le modèle IDM [32] et présenté dans la section 4.1. Notre modèle peut être adapté aux autres modèles de poursuite. Pour modéliser les holons trafic, la section 4.2 présente les paramètres décrivant les états internes des holons trafic. En cas de non commutation, les holons trafic perçoivent l'environnement (cf. section 4.3) et mettent à jour leurs états (cf. section 4.4). En cas de commutation vers le niveau immédiatement supérieur, les super-holons trafic estiment leurs états, présentés dans la section 4.5. Dans le cas d'une commutation vers le niveau immédiatement inférieur les sous-holons trafic estiment leurs états comme présenté dans la Section 4.5. Enfin, une étude de cas et une discussion sont présentées.

4.1 Intelligent Driver Model

IDM [32] est un modèle poursuite à temps continu pour la simulation du trafic sur autoroute et zone urbaine. Nous avons choisi IDM parce que sa mise en oeuvre est simple, que les résultats du calibrage sont disponibles et IDM tend à devenir un standard en matière de simulation microscopique du trafic. De plus, IDM a tendance à être un modèle de poursuite réaliste . IDM est formulé comme une équation différentielle ordinaire et, par conséquent, l'espace et

2. www.traffic-simulation.de

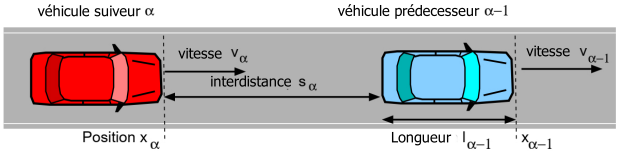


FIGURE 7 – Paramètres des modèles de poursuite. Les indices des véhicules α sont ordonnés de tel sorte que $(\alpha-1)$ dénote le véhicule prédécesseur. s_α est la distance entre pare-chocs définie par $s_\alpha = x_{\alpha-1} - x_\alpha - l_{\alpha-1}$. l_α est la longueur du véhicule et x_α la position. La vitesse relative est définie par $\Delta v_\alpha := v_\alpha - v_{\alpha-1}$ [17]

le temps sont traités comme des variables continues. IDM est caractérisé par une fonction d'accélération $\dot{v} := dv/dt$ qui dépend de la vitesse réelle $v(t)$, de l'interdistance $s(t)$ et de la différence de vitesse $\Delta v(t)$ comme indiqué par la figure 7. L'accélération IDM du véhicule α est donnée par Eq. (1).

$$\dot{v}_\alpha = \frac{dv_\alpha}{dt} = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right] \quad (1)$$

avec $s^*(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}}$.

v_0 est la vitesse désirée. s_0 est l'inter-distance minimale. T est l'écart temporel désiré. a est l'accélération maximale du véhicule et b est la décélération confortable. L'accélération IDM d'un véhicule α peut se diviser en un terme représentant le trafic libre donné par $\dot{v}_\alpha^{libre} = a(1-(v_\alpha/v_0)^\delta)$ et un terme représentant l'interaction donné par : $\dot{v}_\alpha^{int} = -a(s^*(v_\alpha, \Delta v_\alpha)/s_\alpha)^2$

4.2 Estimation des Paramètres des Holons Trafic

Les paramètres qui décrivent l'état d'un holon trafic α sont : la position x_α, la vitesse v_α, l'accélération $\dot{v}_\alpha$, l'inter-distance courante entre le véhicule leader et le véhicule suiveur s_α, la voie où se déplacent les holons trafic L_α, la longueur l_α et le poids (le nombre de sous-holons trafic composant un super-holon trafic) w_α. En plus, il existe des paramètres IDM fixes, définis par le biais de la calibration : l'écart temporel désiré T, l'accélération maximale a et décélération b.

Comme indiqué précédemment, la complexité conduit à un compromis [25]. La simulation au niveau micro est précise mais elle nécessite un coût de calcul élevé, tandis que la simulation aux niveaux supérieurs est grossière mais possède un coût de calcul acceptable. Dans notre modèle, la simulation peut basculer vers une représentation plus grossière ou vers une représentation plus fine et inversement, sachant que les représentations les plus fines sont les

plus désirées. Comme nous considérons plusieurs niveaux de détail dans le même modèle, la question de la transition entre ces niveaux devient cruciale [10]. Les niveaux supérieurs diminuent la précision de la simulation et leur intégration nécessite donc de quantifier l'erreur d'approximation. Cette action est effectuée par des indicateurs multiniveaux au sein des SMAH [8, 9]. Les indicateurs multiniveaux décrits dans ce document sont basés sur l'écart type. L'écart-type est un concept statistique utilisé pour mesurer la dispersion d'un ensemble de données. Lorsque la valeur de l'écart type est petite alors les données sont sensiblement homogènes tandis que lorsque la valeur de l'écart type est élevée alors les données semblent être hétérogènes. Dans cet article, l'écart-type est utilisé pour mesurer l'homogénéité des sous-holons au sein de leur super-holon. La faible valeur de l'écart type signifie que les sous-holons semblent homogènes et que leur regroupement au sein d'un super-holon semble être acceptable. Au contraire, la valeur élevée de l'écart type signifie que les sous-holons sont hétérogènes et que leur regroupement au sein d'un super-holon semble être mauvaise. L'écart-type s'applique aux variables vitesse σ^{vit} et inter-distance σ^{dist}. L'écart-type des vitesses mesure la dispersion de la vitesse des sous-holons et l'écart type des inter-distances mesure la dispersion des inter-distances entre les sous-holons. Notons que pour un holon atomique α, $\sigma^{vit}_{\alpha} = \sigma^{dist}_{\alpha} = 0$ et $w_{\alpha} = 1$.

4.3 Modèle de Perception : Mise à jour du voisinage

La simulation multiniveaux nécessite une distinction claire entre le modèle d'application agent et le modèle de l'environnement [10] afin de permettre une gestion indépendante et spécifique de leurs niveaux d'abstraction respectifs. Ce manuscrit se focalise sur le modèle d'application agent. Le modèle d'environnement multiniveaux pour la simulation de trafic basé sur les SMAH a déjà été proposé dans la littérature [9, 8]. Sur la base de ces travaux, un holon trafic α peut percevoir l'environnement à chaque pas de temps, ce qui lui permet de connaître la position et la vitesse du holon trafic prédécesseur $\alpha - 1$ donné par le modèle d'environnement. Cela permet au holon trafic de calculer l'inter-distance nette par rapport à son prédécesseur holon trafic s_{α} et à la différence de vitesse par rapport à son prédécesseur holon trafic Δv_{α}.

4.4 Modèle de comportement : Mise à jour des états

L'espace et le temps sont traités comme des variables continues par IDM. Notre modèle est un modèle à pas de temps. Nous avons donc besoin d'un schéma de comportement discret pour chaque holon trafic à chaque pas de temps. Dans les modèles de poursuite ou les modèles à base d'agents, il est naturel d'utiliser un schéma explicite supposant les *accélérations* constantes à chaque intervalle de pas de temps Δt [17], ce qui conduit donc aux règles de mises à jour numériques présentées par Éq. (2). Eq. (2) converge vers la solution exacte de Eq. (1) pour $\Delta t \to 0$ [17]. Eq. (2) décrit le comportement d'un holon trafic utilisé pour mettre à jour ses états au cours du temps

$$\begin{aligned} v_{\alpha}(t + \Delta t) &= v_{\alpha}(t) + \dot{v}_{\alpha}(t)\Delta t, \\ x_{\alpha}(t + \Delta t) &= x_{\alpha}(t) + v_{\alpha}(t)\Delta t + \frac{1}{2}\dot{v}_{\alpha}(t)(\Delta t)^2 \end{aligned} \tag{2}$$

4.5 Estimation des états du niveau inférieur vers le niveau supérieur

Soit h un super-holon trafic qui possède n de sous-holons trafic $h_{\alpha}, \alpha = 1, ..., n$. Les indices α sont ordonnés de telle sorte que $(\alpha - 1)$ dénote le holon trafic prédécesseur. Les états (variables caractéristiques) d'un super-holon trafic h sont estimés à partir des états de ses sous-holons trafic $h_{\alpha}, \alpha = 1, ..., n$ et donnés par Eq. (3) - (9). Dans la suite, nous supposons que les variables avec indices sont pour les sous-holons trafic et les variables sans indices sont pour les super-holons trafic.

— **Vitesse:** La vitesse d'un super-holon trafic est donnée par Eq. (3). C'est la moyenne des vitesse de ses sous-holons trafic.

$$v = \frac{1}{n}\sum_{\alpha=1}^{n} v_{\alpha} \tag{3}$$

— **Position:** La Position d'un super-holon trafic est donnée par Eq. (4). C'est la position du sous-holon trafic qui précède tous les autres sous-holons trafic.

$$x = x_1 \tag{4}$$

— **Voie:** La voie d'un super-holon trafic est donnée par Eq. (5). C'est la même voie que tous les sous-holons trafic.

$$L = L_{\alpha}, \alpha = 1, ..., n \tag{5}$$

— **Longueur:** La longueur d'un super-holon trafic est donnée par Eq. (6). C'est la longueur agrégée des sous-holons trafic ainsi que de l'inter-distance entre eux.

$$l = \sum_{\alpha=1}^{n} l_{\alpha} + \sum_{\alpha=1}^{n-1} s_{\alpha} \tag{6}$$

— **Poids:** Le poids d'un super-holon trafic est donné par Eq. (7). C'est la somme des poids de ses sous-holons trafic.

$$w = \sum_{\alpha=1}^{n} w_{\alpha} \tag{7}$$

— **Ecart type des vitesses:** L'écart type des vitesses est donné par Eq. 8. Il mesure la dispersion entre les vitesses des sous-holons trafic.

$$\sigma^{vit} = \sqrt{\frac{1}{n}\sum_{\alpha=1}^{n}(v_\alpha - \mu^{vit})^2} \qquad \mu^{vit} = v = \frac{1}{n}\sum_{\alpha=1}^{n} v_\alpha \tag{8}$$

— **l'écart type des interdistances:** L'écart type des inter-distances est donné par Eq. 9. Il mesure la dispersion des inter-distances entre les sous-holons trafic.

$$\sigma^{dist} = \sqrt{\frac{1}{n-1}\sum_{\alpha=1}^{n-1}(s_\alpha - \mu^{dist})^2} \qquad \mu^{dist} = 1/(n-1)\sum_{\alpha=1}^{n-1} s_\alpha \tag{9}$$

Plus la valeur de σ^{dist} et de σ^{vit}, est petite, plus le regroupement au sein d'un super-holon trafic est acceptable.

4.6 Estimation des états du niveau supérieur vers le niveau inférieur

Les états (variables caractéristiques) des sous-holons trafic $h_\alpha, \alpha = 1, ..., n$ sont générés à partir de l'état de leur super-holon trafic h.

— **Vitesses:** Nous supposons que les vitesses des sous-holons trafic au sein du super-holon suivent la loi normale avec la moyenne μ^{vit} et l'écart type σ^{vit}. La loi normale est le modèle le plus rependu pour caractériser la variation quantitative des données. La loi normale est une norme de référence pour de nombreux problèmes de probabilité et se rapproche à de nombreux phénomènes naturels [5]. Les données dans la loi normale sont résumées en utilisant la moyenne et l'écart type [5], ce qui est le cas dans notre modèle. Pour générer les vitesses des sous-holons trafic basées sur la vitesse et l'écart type de la vitesse du super-holon trafic, le modèle utilise la méthode de Box-Muller [3]. La transformation de Box-Muller est une méthode d'échantillonnage de nombres pseudo-aléatoire permettant de générer des paires de nombres aléatoires indépendants, standard et normalement distribués. La transformation Box-Muller standard génère des valeurs à partir de la loi normale standard $\mathcal{N}(0,1)$. De $\mathcal{N}(0,1)$, nous pouvons déduire $\mathcal{N}(\mu^{vit}, (\sigma^{vit})^2)$ tel que si $X \sim \mathcal{N}(0,1)$ et $Z = \mu^{vit} + \sigma^{vit} X$ alors $Z \sim \mathcal{N}(\mu^{vit}, (\sigma^{vit})^2)$. Les vitesses $v_{\alpha,\alpha=1,...,n}$ des sous -holons trafic sont données par Eq. 10.

$$v_{\alpha,\alpha=1,...,n} = \mu^{vit} + \sigma^{vit} Box\text{-}Muller(0,1) \tag{10}$$

— **Positions:** Nous supposons que les interdistances entre les sous-holons trafic au sein de leur super-holon suivent la loi normale avec la moyenne μ^{dist} et l'écart type σ^{dist}. La position des sous-holons trafic $x_{\alpha,\alpha=1,...,n}$ est donnée par Eq. (11).

$$\begin{cases} x_1 = x \\ x_{\alpha,\alpha=2,...,n} = x_{\alpha-1} - l_{\alpha-1} - s_\alpha \end{cases} \tag{11}$$

avec $s_\alpha = \mu^{dist} + \sigma^{dist} Box\text{-}Muller(0,1)$

— **Voies:** La voie des sous-holons trafic $L_{\alpha,\alpha=1,...,n}$ est la même voie que celle de leur super-holon trafic présenté par (12).

$$L_{\alpha,\alpha=1,...,n} = L \tag{12}$$

— **Longueur, poids:** d'un sous-holon trafic demeure le même qu'avant la requête de commutation vers les niveaux supérieurs.

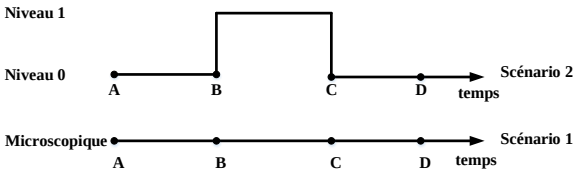


FIGURE 8 – Deux scenarios d'exécution

4.7 Résultats expérimentaux et discussion

Pour évaluer notre modèle multiniveau dynamique basé sur la densité, deux scénarios sont définis comme indiqué par la figure 8. Dans le scénario 1, la simulation est effectuée uniquement au niveau 0 du point A au point D. Dans le scénario 2, de A à B et de C à D, la simulation est effectuée au niveau 0 et de B à C, la simulation est au niveau 1. La figure 9 présente la durée d'exécution du modèle

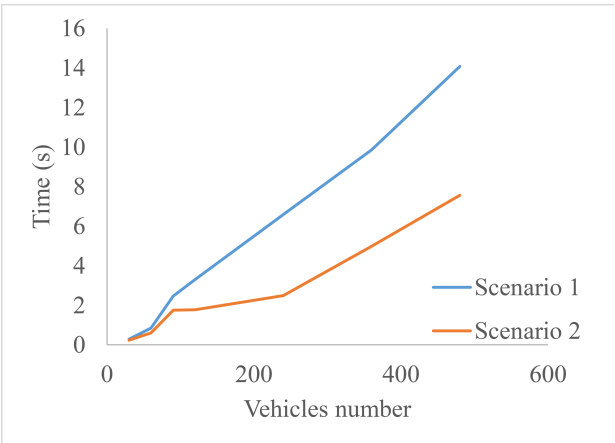


FIGURE 9 – Performance du modèle

multiniveau proposé (scénario 2) et du modèle entièrement microscopique (scénario 1) pour plusieurs véhicules avec une perte de précision maximale de précision toujours inférieure à 10 %. Le gain computationnel est compris entre 25% et 42%. Le gain computationnel étant supérieur à la perte de la précision rend notre approche multiniveau basée sur la densité très intéressante pour gérer la modélisation et la simulation du trafic routier à grande échelle.

Dans le trafic réel, notre modèle multiniveau peut être appliqué avec succès comme suit : la représentation grossière dans les sections de la route où le trafic est stationnaire et la représentation fine dans les sections où le trafic n'est pas stationnaire. Le trafic est stationnaire lorsque le trafic évolue en conservant la même configuration (toutes les trajectoires sont parallèles et équidistantes). Autrement dit, le trafic est stationnaire lorsque la différence de vitesse entre les véhicules est sensiblement nulle. Le principal avantage de la décomposition stationnaire / non stationnaire est qu'elle n'est pas statique à priori et dépend du flux. Plusieurs situations conduisent à un état de trafic stationnaire, tel que congestion, convoi, peloton, etc. En effet, la congestion, par exemple, est un phénomène récurrent. TomTom [3], dans son rapport de 2013, indique que, dans des villes comme Moscou, Rio de Janeiro, Mexico, Istanbul et Beijing, les gens dépensent en moyenne plus de 75% de temps supplémentaire en raison des bouchons. En outre, dans son rapport de 2009, le Texas Transportation Institute, qui mène une enquête sur la congestion du trafic aux États Unis, a déclaré qu'en 2007, la congestion avait occasionné un retard de déplacement estimé à 4,2 milliards d'heures. Les embouteillages mettent en évidence les groupes de véhicules, en particulier aux heures de pointe, ce qui rend le trafic stationnaire. Cependant, dans la vie réelle, le trafic est stationnaire pendant une période spécifique et devient ensuite stationnaire. Il est donc nécessaire de prendre en compte le temps maximum lorsque le trafic est stationnaire lors de l'exécution des niveaux supérieurs de notre modèle. En d'autres termes, nous devons évaluer le temps d'exécution maximal du niveau supérieur, ce qui permet de conserver un faible taux de perte de précision. Cela se fera dans le futur.

5 Conclusion

Le paradigme SMAH est un outil efficace pour modéliser des systèmes complexes multiniveaux à grande échelle. Cet article propose un modèle multiniveau dynamique du trafic à grande échelle basé sur la densité. Le modèle se focalise sur H-DBSCAN en tant qu'extension de l'algorithme de clustering bien connu DBSCAN pour créer la hiérarchie à plusieurs niveaux de détail. Une méthode pour basculer dynamiquement entre les niveaux supérieurs et inférieurs est présentée. L'évaluation du modèle présente des bons résultats pour faire face au compromis entre la précision de la simulation et la disponibilité des ressources de calcul. Les travaux futurs incluent la limitation de l'intervalle d'exécution des niveaux supérieurs pour conserver de bons résultats.

3. TomTom est une entreprise leader dans la fabrication de produits de navigation pour le trafic.

Remerciements

Ce travail est partiellement supporté par le Conseil Régional de Bourgogne France-Comté dans le cadre du projet UrbanFly 20174-06234/06242. Ce travail est également partiellement supporté par le programme Wallenberg AI, Autonomous Systems and Software Program (WASP) financé par la fondation Knut et Alice Wallenberg.

Références

[1] N Bouha, G Morvan, A Hassane, and Y Kubera. 2015. A first step towards dynamic hybrid traffic modelling. In *Proc. of 29th European Conf. on modelling and simulation (ECMS)*. 64–70.

[2] E Bourrel and J Lesort. 2003. Mixing micro and macro representations of traffic flow : a hybrid model based on the LWR theory. In *82th Annual Meeting of the Transportation Research Board*.

[3] G.E.P. Box and M.E. Muller. 1958. A Note on the Generation of Random Normal Deviates. *Ann. Math. Statist.* (1958). https://doi.org/10.1214/aoms/1177706645

[4] F Cao, M Ester, W Qian, and Zhou A. 2006. Density-Based Clustering over an Evolving Data Stream with Noise. In *SDM*.

[5] L Eckhard and Werner S.A. 2011. Problems with Using the Normal Distribution and Ways to Improve Quality and Efficiency of Data Analysis. *PLOS ONE* (2011). https://doi.org/10.1371/journal.pone.0021403

[6] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *2nd International Conference on Knowledge Discovery and Data mining*. 226–231.

[7] Jacques Ferber. 1999. *Multi-Agent Systems. An Introduction to Distributed Artificial Intelligence*. Addison Wesley.

[8] S Galland, F Balbo, N Gaud, S Rodriguez, G Picard, and O Boissier. 2015. A multidimensional environment implementation for enhancing agent interaction. In *14th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS15)*.

[9] S Galland and N Gaud. 2014. Holonic model of a virtual 3D indoor environment for crowd simulation. In *Proceedings of the International Workshop on Environments for Multiagent Systems (E4MAS14)*. Springer.

[10] N. Gaud, S. Galland, F. Gechter, V. Hilaire, and A. Koukam. 2008. Holonic multilevel simulation of complex systems : Application to real-time pedestrians simulation in virtual urban environment. *Simulation Modelling Practice and Theory, Elsevier* (2008).

[11] C. Gerber, J. Siekmann, and G. Vierke. 1999. Holonic Multi-Agent Systems. *Research Center for Artificial Intelligence (DFKI) Tech. Rep. 681, German* (1999).

[12] A Hassane, Y Kubera, and G Morvan. 2014. Dynamic Hybrid Traffic Flow Modeling. (2014). https://doi.org/arXiv:1401.6773v1

[13] MS EL Hmam, A Hassane, D JOLLY, and A Benasser. 2006. Macro-Micro Simulation of Traffic Flow. *IFAC Proceedings Volumes* (2006). https://doi.org/10.3182/20060517-3-FR-2903.00189

[14] Barceló Jaume. 2010. *Fundamentals of Traffic Simulation*. Springer.

[15] N.R. Jennings. 2001. An agent-based approach for building complex software systems. *Commun. ACM* 44, 4 (April 2001), 35–41.

[16] N. R. Jennings and M. J. Wooldridge. 1998. Applications of intelligent agents. *Agent Technology : Foundations, Applications and Markets* (1998).

[17] A Kesting, M Treiber, and D Helbing. 2008. *Agents for Traffic Simulation*. Technical Report arXiv :0805.0300. https://arxiv.org/pdf/0805.0300.pdf

[18] Arthur Koestler. 1967. *The Ghost in The Machine*. Hutchinson.

[19] M. J. Lighthill, F. R. S., and G. B. Whitham. 1955. On kinematic waves II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London A : Mathematical, Physical and Engineering Sciences* (1955). https://doi.org/10.1098/rspa.1955.0089

[20] L. Liu, H. Huang, Y. Guo, and F. Chen. 2009. rDenStream, A Clustering Algorithm over an Evolving Data Stream. In *2009 International Conference on Information Engineering and Computer Science*.

[21] S Mammar, S Mammar, and J Lebacque. 2006. Highway Traffic Hybrid Macro-micro Simulation Model. In *11th IFAC Symposium on Control in Transportation System*.

[22] Y Mualla, W Bai, S Galland, and C Nicolle. 2018. Comparison of agent-based simulation frameworks for unmanned aerial transportation applications. *Procedia Comput. Sci* 130 (2018), 791–796.

[23] Laurent Navarro, Vincent Corruble, Fabien Flacher, and Jean-Daniel Zucker. 2013. A flexible approach to multi-level agent-based simulation with the mesoscopic representation. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*. International Foundation for Autonomous Agents and Multiagent Systems, 159–166.

[24] G.F. Newell. 1989. Comments on traffic dynamics. *Transportation Research Part B : Methodological* (1989). https://doi.org/10.1016/0191-2615(89)90015-5

[25] D. E. O'Leary. 1998. Developing multiple-agent systems is more than top-down vs. bottom-up. *IEEE Intelligent Systems and their Applications* (1998). https://doi.org/10.1109/5254.671082

[26] S. Rodriguez, V. Hilaire, and A. Koukam. 2007. Towards a holonic multiple aspect analysis and modeling approach for complex systems : Application to the simulation of industrial plants. *Simulation Modelling Practice and Theory, Elsevier* (2007).

[27] J Sewall, D Wilkie, and M.C. Lin. 2011. Interactive Hybrid Simulation of Large-scale Traffic. *ACM Trans. Graph.* 30, 6, Article 135 (Dec. 2011), 12 pages. https://doi.org/10.1145/2070781.2024169

[28] M Stratos, E Ntoutsi, N Pelekis, and T Yannis. [n. d.]. An evaluation of data stream clustering algorithms. *Statistical Analysis and Data Mining : The ASA Data Science Journal* ([n. d.]). https://doi.org/10.1002/sam.11380

[29] IH Tchappi, S Galland, VC Kamla, and JC Kamgang. 2018. A Brief Review of Holonic Multi-Agent Models for Traffic and Transportation Systems. *Procedia Computer Science* (2018).

[30] IH. Tchappi, VC Kamla, S. Galland, and JC Kamgang. 2017. Towards an Multilevel Agent-based Model for Traffic Simulation. Procedia Computer Science, 887–892.

[31] IH. Tchappi, VC Kamla, S. Galland, JC Kamgang, CMS. Nono, and H. Zhao. 2018. Holonification Model for a Multilevel Agent-Based System. Application to Road Traffic. *Personal and Ubiquitous Computing* (2018).

[32] M. Treiber, A. Hennecke, and D. Helbing. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical Review E 62* (2000).

Un comportement déterministe pour une dynamique des prix réaliste

Philippe Mathieu
philippe.mathieu@univ-lille.fr

Remi Morvan
remi.morvan@ens-paris-saclay.fr

Univ. Lille, CNRS, Centrale Lille, UMR 9189 – CRIStAL (équipe SMAC), F-59000 Lille, France

Résumé

Depuis quelques années de nombreuses études en finance de marché se sont appuyées sur des agents artificiels, que ce soit pour l'évaluation d'une stratégie, l'étude de la dynamique des prix ou l'exécution efficace des ordres. Les comportements utilisés, bien souvent pour assurer de la liquidité sur les marchés simulés, s'appuient sur des comportements stochastiques, donc non déterministes, aussi bien dans le cadre de comportements chartistes que de comportements fondamentalistes. Le plus simple de ces comportements est le fameux Zero Intelligent Trader. Nous soutenons ici qu'un comportement rationnel et entièrement déterministe suffit à la fois pour reproduire les faits stylisés classiques du domaine, pour conserver la liquidité du marché mais aussi pour assurer que des agents qui ont des paramètres initiaux différents aient des possibilités différentes de s'enrichir. Pour illustrer cela nous introduisons les Deterministic Artifical Traders `DAT` *et nous montrons leur efficacité au regard des faits stylisés. Ce résultat illustre le fait que le complexe peut surgir du simple, et qu'il n'est pas nécessaire d'avoir de multiples comportements ou d'utiliser l'aléatoire pour assurer ces caractéristiques fondamentales d'une bonne approche de la simulation de marchés par agents artificiels.*

Mots-clés : *Finance, systèmes multi-agents, marchés artificiels, comportements déterministes, systèmes complexes*

Abstract

In recent years, many studies on financial markets have relied on artificial agents, whether for the evaluation of strategies, the study of price dynamics or the efficient execution of orders. The behaviours used in those studies, often Zero-Intelligence Traders, fundamentalists or chartists, are stochastic and therefore non-deterministic, mainly because such agents easily yield a market that continuously fixes prices. We argue here that a rational and fully deterministic behaviour is sufficient both to reproduce the classic stylized facts of the field, but also to ensure that agents with different initial parameters have different opportunities to enrich themselves. To illustrate this purpose, we introduce Deterministic Artificial Traders, or `DAT`*, and we show their performances in several situations. This result illustrates the fact that financial markets are a complex system, as some deterministic behaviours lead to some randomness in the market, both at the macroscopic and microscopic levels.*

Keywords: *agent-based computational economics, multi-agents systems, deterministic behaviours, complex systems*

1 Introduction

Depuis 1987 et le premier modèle multi-agent pour marchés financiers proposé par [6], beaucoup de travaux dans le domaine de l'économie et de la finance s'appuient sur des simulations microscopiques dans lesquelles des entités artificielles appelées "agents" ont leur propre comportement et prennent leurs décisions en toute autonomie. Ce courant de pensée est souvent nommé "Agent-based Computational Economics" (ACE). Les avantages de cette approche par la simulation sont indéniables puisque qu'il s'agit de la seule approche permettant d'individualiser les acteurs du système, d'offrir une approche comportementale adaptée à l'étude réalisée et surtout, de permettre un impact des différentes actions sur les agents eux mêmes. Elle offre ainsi l'opportunité d'évaluer les différents types de comportements aussi bien au niveau macroscopique qu'au niveau microscopique : Il est alors possible d'obtenir des modèles plus explicatifs que prédictifs. Même dans les cas les plus simples, l'agent a sa propre liquidité et ses propres placements et les différents ordres envoyés ont une incidence à la fois sur le marché et sur les agents eux-mêmes, formant ainsi la base de boucles de rétro-actions propres à un système complexe [13, 14]. Plusieurs études récentes présentent l'historique de ces travaux. On trouvera notamment dans [8, 7] ou même [1] de

nombreux détails sur cette évolution.

Nous rappelons que dans un système multi-agents, les agents sont autonomes. Ils prennent leurs décisions seuls, en percevant éventuellement leur environnement (marché, congénères). Par conséquent, chaque agent dispose d'une méthode de décision lui permettant d'envoyer un ordre, généralement un ordre à cours limité, défini par une direction un prix et une quantité. Dans cet article, un comportement désigne une fonction qui met en correspondance un ensemble de paramètres à une méthode de décision. Le système général appelle cette méthode équitablement pour chaque agent, soit en temps réel (processus, threads), soit en temps simulé (tour de parole). Bien sûr, l'agent peut, lorsque sa méthode de décision est appelée, décider de ne pas envoyer d'ordre. Il illustre ainsi son autonomie. Ce travail a été réalisé en utilisant la plateforme ATOM [9, 10], qui implémente un ensemble d'agents et de carnets d'ordres qui respectent ces différents principes.

Durant la dernière décennie, plusieurs comportements d'agents ont été proposés dans ce cadre comme ZIP, GD, GDX, les stratégies AA [12, 3] pour un tout autre objectif, mais la très grande majorité des travaux de la littérature s'appuie sur le fameux *Zero Intelligent Trader* (ou ZIT) proposé par [4]. Il possède un comportement simple et efficace et fournit une entropie constante au marché, au sens où des prix seront toujours fixés. Un autre point interessant est qu'il est également suffisant pour obtenir des faits stylisés reconnus dans ce domaine. Cependant, ce comportement est purement stochastique. Il n'existe pas à notre connaissance de comportement déterministe qui, lorsqu'il est exécuté sur un double carnet d'ordres amène à un marché qui à la fois fixe des prix continuellement, mais respecte aussi les faits stylisés. Et pourtant, le fait d'avoir des comportements déterministes est souhaitable, à la fois pour avoir plus de réalisme mais également d'un point de vue pratique pour pouvoir étudier les marchés comme un tout, incluant les agents qui en font partie. Cependant, la construction de comportements déterministes est loin d'être facile car sur un marché où il n'y a que des agents déterministes et rationnels, ces agents auront tendance à tous envoyer le même type d'ordre, ce qui ne fixera aucun prix puisque fixer un prix nécessite la rencontre d'un ordre `Ask` et d'un ordre `Bid` dans le double carnet d'ordres (voir figure 1).

1. Téléchargeable sur https://github.com/cristal-smac/atom.

Nous proposons par ailleurs dans cet article une méthode pour faire la "moyenne" de certains comportements déterministes, qui amène à d'autres comportements déterministes, ce qui nous permettra de répondre aux questions suivantes :

1. Un comportement déterministe peut-il animer un marché qui fixe des prix continuellement ?
2. Si oui, à quelle complexité de comportement faut-il s'attendre pour assurer la reproduction des faits stylisés ?
3. Est-ce que la chance existe sur de tels marchés, malgré le déterminisme des agents ?

En d'autres termes, le hasard peut-il émerger au niveau macroscopique à partir d'un niveau microscopique déterministe dans le cadre des carnets d'ordres à double enchère, et peut-il, à travers les boucles de rétroaction qui définissent les systèmes complexes, conduire à un certain caractère aléatoire au niveau microscopique ? Notons que pour répondre à ces questions, il faut réussir à construire un comportement autonome, qui ne nécessite pas l'utilisation d'agents tels que les ZIT pour fournir une certaine entropie au marché : notre agent doit fournir seul de l'entropie au marché, tout en étant déterministe.

Dans la section 2, nous résumons le fonctionnement des marchés financiers basés sur des carnets d'ordres à double enchère et présentons rapidement les faits stylisés observés sur ces dits marchés. Puis la section 3 décrit les comportements déterministes élémentaires que nous allons utiliser pour construire notre agent. D'abord nous construisons un simple agent fondamentaliste `DAT-F` [2]. Nous le raffinons ensuite en un agent fondamentaliste informé `DAT-IF` en introduisant un comportement multi-jours. Puis nous présentons un agent chartiste déterministe qui, contrairement au `DAT-F` et au `DAT-IF`, n'est pas un comportement autonome en ce sens que lorsque seuls des agents chartistes sont utilisés sur un marché, aucun prix ne sera fixé [3]. Enfin, la section 4 présente notre définition de la moyenne déterministe de comportements. Pour finir, la section 5 montre les résultats des comportements `DAT-IFC` obtenus en moyennant les `DAT-IF` avec des comportements chartistes.

2. Un fondamentaliste base son comportement sur une valeur fondamentale. Un chartiste regarde uniquement la forme des prix

3. Ceci provient du fait que nos agents chartistes utilisent l'historique des prix, qui est réduit à une unique valeur au début d'une simulation, pour prendre leur décision

2 Marché financier et faits stylisés

2.1 Fonctionnement d'un marché financier

Les marchés d'actions dirigés par les ordres permettent de négocier des titres financiers à l'achat ou à la vente sans intermédiaire, à l'instar d'Euronext et NYSE. Ces marchés gèrent l'offre et la demande grâce à un système à double carnet d'ordres pour chacun des titres gérés, l'un pour les achats (`Bid`), l'autre pour les ventes (`Ask`). Les Traders envoient différents types d'ordres dont le plus général est le limit order défini par un triplet (direction, prix, quantité). Le prix d'un ordre d'achat est considéré comme un prix maximum acceptable pour l'acheteur, tandis que le prix d'un ordre de vente est considéré comme un prix minimum acceptable pour le vendeur. Dans chacun des carnets, les ordres sont triés sur le prix puis la date d'arrivée en cas d'égalité. La meilleure offre d'achat `bestBid` est constituée par l'ordre d'achat ayant le prix le plus élevé, la meilleure offre de vente `bestAsk` est constituée par l'ordre de vente ayant le prix le moins élevé. L'écart entre ces deux offres est appelé le `bid-ask spread`. Dès que le `bestBid` devient supérieur ou égal au `bestAsk`, un ou plusieurs prix sont fixés, selon les ordres touchés et en fonction des quantités souhaitées. Les ordres impactés sont exécutés et retirés du carnet. Quand deux ordres fixent un prix, c'est le prix du premier arrivé qui est utilisé.

dir	prix	qute
`Ask`	1250	2
`Ask`	1230	1
`Ask`	1225	2
bid-ask spread		
`Bid`	1220	3
`Bid`	1210	1

dir	prix	qute
`Ask`	1250	2
bid-ask spread		
`Bid`	1230	1
`Bid`	1220	3
`Bid`	1210	1

Prices
1225
1250

FIGURE 1 – À gauche un état stable du carnet d'ordre. à droite sa transformation après exécution d'un limit order (`Bid`, 1230, 4) et en dessous, les prix fixés en conséquence.

Figure 1 on peut voir à gauche un double carnet d'ordre nécessaire à la gestion d'un titre. Dans la partie haute les ordres de vente, dans la partie basse les ordres d'achat. Cinq ordres sont déjà parvenus au marché. Le `bestBid` est à 1220 et

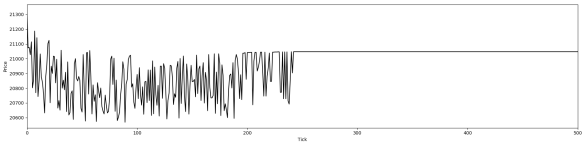


FIGURE 2 – Sequence de prix obtenue sur un marché utilisant 50 ZITs qui ne peuvent jamais avoir de liquidité négative ni d'actifs négatifs.

est donc inférieur au `bestAsk` qui est à 1225. Aucun prix n'est donc fixé. Si un ordre d'achat (`Bid`, 1230, 4) arrive, il sera placé en tête du carnet des achats et permettra l'exécution des deux meilleurs ordres de vente en fixant un prix à 1225 pour une quantité de 2 et un prix à 1230 pour une quantité de 1. Un résidu de cet ordre (`Bid`, 1230, 1) subsistera en tête du carnet des achats. Le cash et les titres de chaque agents seront mis à jour en conséquence.

2.2 Faits stylisés

Les faits stylisés sont des propriétés communes à tous les marchés financiers basés sur des carnets d'ordres à double enchère. Tout marché artificiel se doit de les reproduire afin d'être un modèle crédible.

Le plus évident des faits stylisés est que le marché ne se bloque jamais, où en d'autres termes qu'il fixe des prix continuellement. Si cette propriété est facilement vérifiée en utilisant de simples ZITs qui choisissent uniformément la direction, le prix et la quantité des ordres qu'ils envoient, il est néanmoins intéressant de constater que dès que des contraintes fortes sont appliquées à ces agents, cette propriété devient beaucoup plus rare. Par exemple, si l'on exécute un marché à double enchère peuplé d'agents ZIT à qui on interdit d'avoir des liquidités ou des actifs négatifs, alors le marché bloquera rapidement, comme on peut le voir sur la figure 2. On imagine donc aisément à quel point il est difficile de construire des agents qui génèrent un marché sur lequel les prix sont fixés en permanence lorsque la contrainte est d'avoir des agents purement déterministes.

Comme la majorité des articles de la littérature, nous nous concentrons ici sur les propriétés suivantes :

— *Queues épaisses*. La distribution des rentabilités possède des queues épaisses comparées à une distribution normale. Par ailleurs, la kurtosis est supérieure à 3.

— *Absence d'auto-corrélation*. l'auto-corrélation des rentabilités est négligeable pour de grandes valeurs de Δt
— *Lente décroissance de l'auto-corrélation des rentabilités absolues*. L'auto-corrélation des rentabilités absolues $((|\log(p_{t+1}) - \log(p_t)|)_t)$ décroit selon une loi de puissance dont l'exposant est compris entre -0,3 et -0,2.

3 Comportements déterministes élémentaires

Notations : Un ordre limite o est un triplet $(o_d,\ o_p,\ o_q)$ dont les éléments désignent la direction de l'ordre (`Ask` or `Bid`), son prix et sa quantité respectivement. Quand il n'est pas nécessaire de spécifier l'un de ces trois paramètres, il sera remplacé par le caractère _ . Si les agents ne souhaitent pas envoyer d'ordre, la convention $o = \emptyset$ est utilisée. De plus, nous noterons $[\![a,\ b]\!]$ l'ensemble des entiers compris entre $a \in \mathbb{Z}$ et $b \in \mathbb{Z}$.

Cette section se focalise sur la construction d'agents minimalistes, i.e. avec le comportement le plus simple possible, déterministe, et permettant de reproduire les faits stylisés classiques. Tout d'abord nous proposons l'agent `DAT-F`, de type fondamentaliste. Puis nous l'améliorons en introduisant un comportement multi-jours amenant à l'agent `DAT-IF`. Ces deux comportements peuvent, lorsqu'ils sont mis sur un marché qui ne contient que ces agents, donner lieu à un marché qui respecte certains faits stylisés. En particulier, le marché ne se bloque jamais. Enfin, un comportement chartiste est décrit.

3.1 Un agent déterministe fondamentaliste : `DAT-F`

Par souci de simplicité, le marché retenu ici est constitué d'un seul actif, et donc d'un seul carnet d'ordres. Cependant, le comportement de `DAT-F` peut être immédiatement généralisé à un marché avec plusieurs actifs/carnets d'ordres.

L'agent `DAT-F` possède une estimation de la valeur fondamentale de l'actif et se fie à cette estimation pour envoyer ses ordres. Son comportement est le suivant : si le `bestAsk` a un prix inférieur à l'estimation de l'agent, alors l'agent enverra une commande `Bid` au prix du `bestAsk`. De même, si le `bestBid` a un prix plus élevé que l'estimation de l'agent, alors l'agent enverra un ordre `Ask`, dont le prix sera égal au prix du `bestBid`. Autrement dit, l'agent n'accepte d'acheter qu'à un prix inférieur à son estimation, et de ne vendre qu'à un prix plus élevé. Sinon, l'agent va essayer de réduire le `bid-ask spread`, tout en le centrant autour de son estimation de la valeur fondamentale.

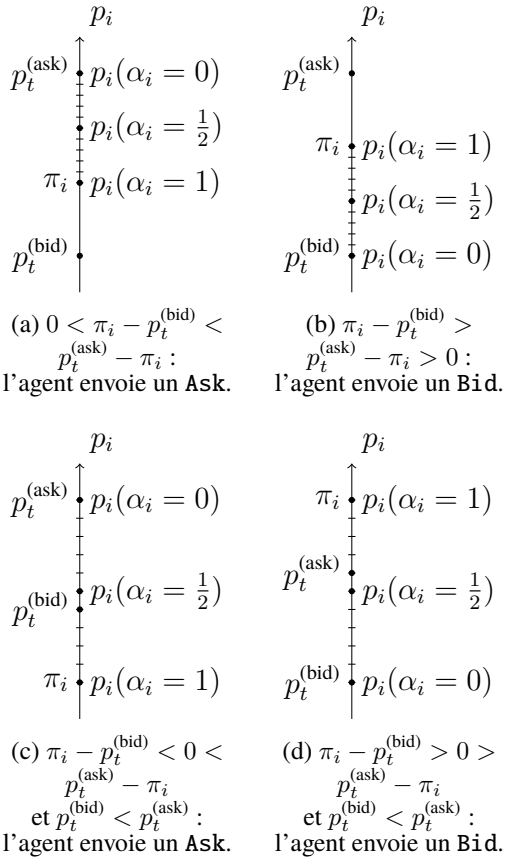
Un agent i possède plusieurs paramètres intrinsèques, qui détermineront quel ordre il enverra, lorsque le marché lui permettra de parler :
— $\alpha_i \in [0, 1]$: son agressivité
— $\kappa_i \in \mathbb{N}^*$: sa confiance
— $\pi_i \in \mathbb{N}^*$: son estimation de la valeur fondamentale

A l'instant t, l'ordre envoyé par cet agent, qui dépend des trois paramètres précédents, est noté $o_{i,t}^{(\mathrm{f})}$. De plus, $p_t^{(\mathrm{ask})}$ et $p_t^{(\mathrm{bid})}$ représentent le prix `bestAsk` et le prix `bestBid` dans le carnet d'ordres, si respectivement il y a au moins un `Ask` et un `Bid` dans ce carnet d'ordres. Dans le cas contraire, ce prix est égal au dernier prix fixé [4], p_t. Donc l'ordre envoyé sera :

$$o_{i,t}^{(\mathrm{f})} = \begin{cases} \text{s'il y a au moins un } \texttt{Ask} \\ \text{et si } p_t^{(\mathrm{ask})} \leq \pi_i \ : \\ \qquad (\texttt{Bid},\ p_t^{(\mathrm{ask})},\ \kappa_i) \\ \text{s'il y a au moins un } \texttt{Bid} \\ \text{et si } p_t^{(\mathrm{bid})} \geq \pi_i \ : \\ \qquad (\texttt{Ask},\ p_t^{(\mathrm{bid})},\ \kappa_i) \\ \text{sinon, si } \pi_i - p_t^{(\mathrm{bid})} > p_t^{(\mathrm{ask})} - \pi_i \ : \\ \qquad (\texttt{Bid},\ p_t^{(\mathrm{bid})} + \alpha_i(\pi_i - p_t^{(\mathrm{bid})}),\ \kappa_i) \\ \text{sinon, si } \pi_i - p_t^{(\mathrm{bid})} < p_t^{(\mathrm{ask})} - \pi_i \ : \\ \qquad (\texttt{Ask},\ p_t^{(\mathrm{ask})} + \alpha_i(\pi_i - p_t^{(\mathrm{ask})}),\ \kappa_i) \\ \text{sinon :} \\ \qquad \emptyset \end{cases}$$

Cet ordre peut s'expliquer de la manière suivante : l'agent n'achète jamais à un prix supérieur à π_i, ni ne vend à un prix inférieur. S'il y a déjà un ordre dans le carnet d'ordres avec lequel il peut faire une transaction en envoyant un ordre qui respecte cette contrainte, il le fait. Dans le cas contraire, il se contente de réduire le `bid-ask spread` en déplaçant soit le `bestAsk` soit le `bestBid` vers π_i tout en respectant les contraintes précédentes, comme cela est illustré dans la figure 3. Plus son agressivité α_i est élevée, plus l'agent aura tendance à acheter à un prix élevé et à vendre à un prix bas.
Il est à noter que lorsque la deuxième situation

4. Il y a toujours un prix d'ouverture dans toutes nos expériences, tout comme sur les marchés réels, donc ce dernier prix fixé existe toujours



(a) $0 < \pi_i - p_t^{(\text{bid})} < p_t^{(\text{ask})} - \pi_i$: l'agent envoie un `Ask`.

(b) $\pi_i - p_t^{(\text{bid})} > p_t^{(\text{ask})} - \pi_i > 0$: l'agent envoie un `Bid`.

(c) $\pi_i - p_t^{(\text{bid})} < 0 < p_t^{(\text{ask})} - \pi_i$ et $p_t^{(\text{bid})} < p_t^{(\text{ask})}$: l'agent envoie un `Ask`.

(d) $\pi_i - p_t^{(\text{bid})} > 0 > p_t^{(\text{ask})} - \pi_i$ et $p_t^{(\text{bid})} < p_t^{(\text{ask})}$: l'agent envoie un `Bid`.

FIGURE 3 – Prix p_i de l'ordre envoyé par le fondamentaliste fonction de $p_t^{(\text{bid})}$, π_i, $p_t^{(\text{ask})}$ et α_i. Les situations 3a et 3b représentent le cas où l'estimation de la valeur fondamentale est plus grande que le `bestBid` et plus petite que le `bestAsk`. Les figures 3c et 3d représentent ce qui se produit quand π_i est plus petit que p_t^{bid} ou plus grand que p_t^{ask}, respectivement. On note que quatre autres situations peuvent apparaitre, quand au moins une des parties `Bid` ou `Ask` du carnet d'ordres est vide, auquel cas $p_t^{\text{bid}} = p_t$ ou $p_t^{\text{ask}} = p_t$. Sous cette hypothèse, on peut donc avoir $\pi_i - p_t^{(\text{bid})} < p_t^{(\text{ask})} - \pi_i \leq 0, 0 \geq \pi_i - p_t^{(\text{bid})} > p_t^{(\text{ask})} - \pi_i$ ou dans l'un des cas 3c ou 3d avec $p_t^{(\text{bid})} \geq p_t^{(\text{ask})}$.

se présente, si son estimation de la valeur fondamentale π_i est à égale distance de $p_t^{(\text{ask})}$ et de $p_t^{(\text{bid})}$, alors l'agent n'envoie aucun ordre. Enfin, notons que, étant donné un agent i, la quantité o_q de ses ordres o est constante, égale à κ_i. Bien qu'étant irréaliste, cette façon de déterminer la quantité permet, tout en concervant un processus déterministe, d'avoir facilement des quantités variables, et donc d'être plus proche des faits stylisés souhaités.

3.2 Un agent informé, déterministe et fondamentaliste : `DAT-IF`

Le comportement fondamentaliste informé proposé dans cette sous-section étend le comportement précédent pour constituer ce que nous appelons un agent `DAT-IF`. Ces agents ont un paramètre supplémentaire, $\iota_i \in [\![0, 9]\!]$, qui représente leur niveau d'information, et leur estimation de la valeur fondamentale dépend maintenant du temps, et est donc désormais notée $\pi_{i,t}$.

Cette valeur fondamentale est calculée de la même manière que ce qui est proposé dans [11] : l'agent i dispose d'une estimation des dividendes versés par la société à la fin des ι_i prochaines périodes/jours. De cette estimation des dividendes futurs vient l'estimation de la valeur fondamentale par l'agent, qui ne change pas à l'intérieur d'une période. À l'intérieur d'un jour, le comportement d'un tel `DAT-IF` est donc parfaitement identique au comportement `DAT-F` décrit dans la sous-section précédente.

Notons n_d le nombre de jours de la simulation, $(d_j)_{1 \leq j \leq n_d}$ les dividendes versés à la fin des différents jours, $I = \max_i \iota_i + 1$ et T_j l'ensemble des moments composant le jour j. On notera qu'une distinction est faite entre un moment et un tick : à l'intérieur d'un même tick, chaque agent parle, donc plusieurs prix sont fixés à des moments différents.
Au début de la journée j, l'agent i reçoit des estimations. $\widehat{d_j^{(j,i)}}, \ldots, \widehat{d_{j+\iota_i-1}^{(j,i)}}$ des ι_i prochains dividendes. Chaque $\widehat{d_{j+k}^{(j,i)}}$ — l'estimation de l'agent i au jour j du dividende d_{j+k} payé à la fin du jour $j+k$ —, pour $k \in [\![0, \iota_i - 1]\!]$ est choisi uniformément entre $(1 - \varepsilon_{i,k}) d_{j+k}$ et $(1 + \varepsilon_{i,k}) d_{j+k}$, où $\varepsilon_{i,k} = \sqrt{k+1} \frac{I - \iota_i}{300}$: si un `DAT-IF` a un niveau d'information plus élevé, alors son estimation du dividende sera, en moyenne, plus proche de la valeur réelle. De même, plus le jour pour lequel un agent reçoit une estimation de la valeur de l'indice est éloigné d'aujourd'hui, plus l'estimation sera éloignée de la valeur réelle du dividende.

Maintenant que les estimations des dividendes futurs sont définies, nous pouvons définir l'estimation de la valeur fondamentale pour les agents `DAT-IF` qui ont un niveau d'information non nul, donc pour qui $\iota_i \geq 1$. Pour ce faire, nous utilisons une généralisation de la formule de Gordon [5] :

pour $j \in [\![1,\ n_d]\!]$ et $t \in T_j$,

$$\pi_{i,t} = \sum_{k=0}^{\iota_i - 2} \frac{\widehat{d^{(j,i)}_{j+k}}}{(1+r_e)^k} + \frac{\widehat{d^{(j,i)}_{j+\iota_i-1}}}{r_e(1+r_e)^{\iota_i-2}}$$

où $r_e = 0.05$ correspond au taux d'interêt ajusté en fonction du risque.

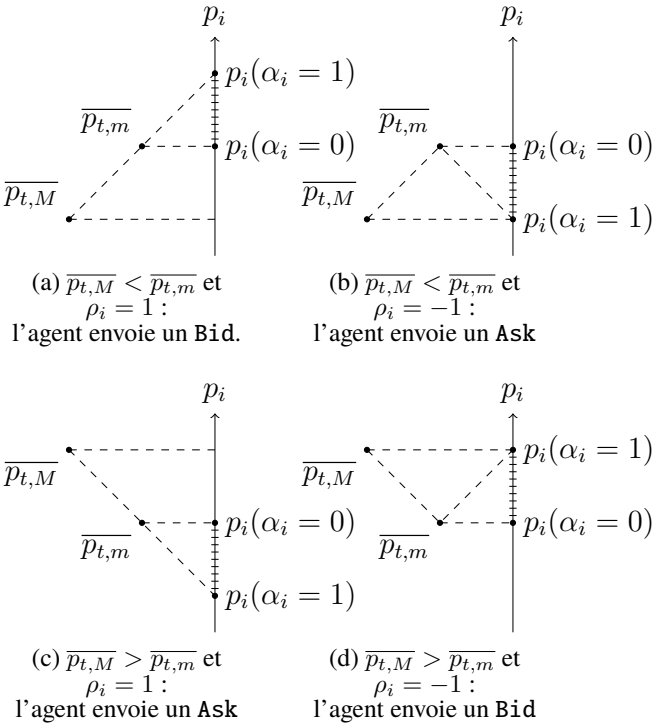
Pour un `DAT-IF` qui possède un niveau d'information ι_i égal à 0, nous choisissons de prendre $\forall j \in [\![1,\ n_d]\!], \forall t \in T_j,\ \pi_{i,t} = p_t$, où p_t désigne le dernier prix fixé au moment t. Contrairement au agents fondamentalistes informés de [11], les agents `DAT-IF` qui n'ont pas d'information ne sont pas des ZIT, mais des agents déterministes similaires à tous égards à ceux des autres agents informés. Ils utilisent simplement le dernier cours comme valeur fondamentale, et supposent donc que le marché est efficient. La seconde différence notable est que nos agents `DAT-IF` ne connaissent pas la valeur exacte des dividendes futurs, mais uniquement une approximation de celle-ci.

Enfin, notons que le comportement intra-journalier des agents de Toth *et al.* est entièrement stochastique, par opposition au comportement proposé ici. Dans les simulations suivantes, $d_1 = 1\ 000$ et d_{k+1} suivent une loi normale centrée sur d_k avec un écart-type de 10.

3.3 Un comportement déterministe chartiste

Un agent chartiste possède deux paramètres : une agressivité $\alpha_i \in [0,\ 1]$ et une valeur $\rho_i \in \{-1, 1\}$, qui représente ce qu'il pense de la tendance des prix : si $\rho_i = 1$, il pense que la tendance va surement persister (si le prix a déjà augmenté, alors il continuera probablement à augmenter), et si $\rho_i = -1$, il pense que la tendance va surement s'inverser.

Le comportement chartiste avec moyenne mobile utilisé par le `DAT-IFC` est définie comme suit : pour $T > 0$, si p_j désigne maintenant le dernier prix fixé au tick j, notons $\overline{p_{t,T}} = \frac{1}{T}\sum_{j=t-T+1}^{t} p_j$ la moyenne des prix des T derniers ticks, s'ils sont définis, et $\emptyset$ sinon, i.e. lorsque l'historique des prix ne comprend pas suffisamment de prix ($t < T - 1$).
Soit m et M deux entiers tels que $m < M$. Dans les simulations suivantes, nous avons choisi



(a) $\overline{p_{t,M}} < \overline{p_{t,m}}$ et $\rho_i = 1$: l'agent envoie un `Bid`.

(b) $\overline{p_{t,M}} < \overline{p_{t,m}}$ et $\rho_i = -1$: l'agent envoie un `Ask`

(c) $\overline{p_{t,M}} > \overline{p_{t,m}}$ et $\rho_i = 1$: l'agent envoie un `Ask`

(d) $\overline{p_{t,M}} > \overline{p_{t,m}}$ et $\rho_i = -1$: l'agent envoie un `Bid`

FIGURE 4 – Le prix p_i de l'ordre envoyé par l'agent chartiste en fonction de $\overline{p_{t,M}}$, $\overline{p_{t,m}}$, α_i et ρ_i. Seules les valeurs extrêmes de p_i — pour $\alpha_i = 0$ et $\alpha_i = 1$ — sont représentées.

$m = 20$ et $M = 50$. Alors :

$$o^{(c)}_{i,t} = \begin{cases} \text{si } \overline{p_{t,M}} \neq \emptyset \text{ et } [(\rho_i = 1 \text{ et } \overline{p_{t,m}} > \overline{p_{t,M}}) \\ \text{ou } (\rho_i = -1 \text{ et } \overline{p_{t,m}} < \overline{p_{t,M}})] \ : \\ \quad (\texttt{Ask},\ \overline{p_{t,m}} - \alpha_i|\overline{p_{t,m}} - \overline{p_{t,M}}|,\ \_) \\ \text{si } \overline{p_{t,M}} \neq \emptyset \text{ et } [(\rho_i = 1 \text{ et } \overline{p_{t,m}} < \overline{p_{t,M}}) \\ \text{ou } (\rho_i = -1 \text{ et } \overline{p_{t,m}} > \overline{p_{t,M}})] \ : \\ \quad (\texttt{Bid},\ \overline{p_{t,m}} + \alpha_i|\overline{p_{t,m}} - \overline{p_{t,M}}|,\ \_) \\ \text{sinon (i.e. si } \overline{p_{t,M}} = \emptyset \text{ ou } \overline{p_{t,m}} = \overline{p_{t,M}}) : \\ \quad \emptyset \end{cases}$$

Moins formellement, l'agent n'envoie pas d'ordre si l'historique des prix contient moins de M prix, ou si la moyenne des m derniers prix est égale à la moyenne des M derniers prix. Dans le cas contraire, si l'agent pense que les tendances sont persistantes et si les prix ont augmenté ($\overline{p_{t,m}} > \overline{p_{t,M}}$), alors, en notant δ la différence entre ces prix, l'agent enverra un ordre `Bid` à un prix compris entre $\overline{p_{t,m}}$ et $\overline{p_{t,m}} + \delta$. Dans la même situation, si l'agent pense que la tendance risque de s'inverser, alors il enverra un ordre entre $\overline{p_{t,m}}$ et $\overline{p_{t,m}} - \delta$. Le cas où les prix ont diminué est symétrique. Cela est illustré dans la figure 4.

4 Moyenne pondérée de comportements déterministes

Il serait assez facile de définir une moyenne pondérée de comportements si l'on accordait aucune importance au déterminisme : il suffirait de définir ce comportement moyen en envoyant l'ordre déterminé par le premier comportement avec une certaine probabilité, et d'envoyer l'ordre déterminé par le second comportement dans le cas contraire.

Nous considérons qu'un comportement est une fonction qui prend comme entrée l'état du marché et l'état de l'agent, et qui renvoie un ordre o qui est soit un ordre à cours limite, soit l'ordre nul $\emptyset$. Ainsi, pour construire un comportement qui met en œuvre une moyenne pondérée du `DAT-IF` avec un comportement chartiste, nous proposons de définir une moyenne pondérée des comportements comme la fonction qui prend les mêmes entrées avec ces comportements et qui renvoie la moyenne pondérée des ordres renvoyés par ces comportements.

Nous définissons donc la moyenne des ordres $(o_k)_{1\leq k\leq n}$ pondérés par $(\lambda_k)_{1\leq k\leq n}$, notée $\overline{(o_k,\lambda_k)_{1\leq k\leq n}}$, de la manière suivante : Si o_k est un ordre à cours limite, on considère d_k sa direction et p_k son prix ; si o_k est l'ordre nul $\emptyset$, on considère $d_k = p_k = \emptyset$. Une valeur, appelée direction numérique, est assignée à chaque direction : $g(\texttt{Ask}) = +1$, $g(\texttt{Bid}) = -1$ et $g(\emptyset) = 0$. Cette valeur est utilisée pour calculer la moyenne pondérée des directions numériques, $d = \sum_{k=1}^{n} \lambda_k g(d_k)$ et la direction moyenne, notée $\tilde{d}$ est définie par :

$$\tilde{d} = \begin{cases} \texttt{Ask} & \text{si } d > 0 \\ \emptyset & \text{si } d = 0 \\ \texttt{Bid} & \text{si } d < 0 \end{cases}$$

Cette direction sera la direction de l'ordre moyenné. Soit K l'ensemble des indices indexant un ordre dont la direction est égale à la direction moyenne, i.e. $K = \{k \in [\![1,\ n]\!] \mid d_k = \tilde{d}\}$, et soit $\Lambda = \sum_{k=K} \lambda_k$ le poids total des ordres indexés par K.

Enfin, l'ordre résultant peut maintenant être défini par $\overline{(o_k,\lambda_k)_{1\leq k\leq n}} = \emptyset$ lorsque $\tilde{d} = \emptyset$ et sinon,

$$\overline{(o_k,\lambda_k)_{1\leq k\leq n}} = \left(\tilde{d},\ \sum_{k\in K} \frac{\lambda_k}{\Lambda} p_k,\ \lceil |d| q_0 \rceil\right)$$

où $\lceil\cdot\rceil$ est la fonction plafond et q_0 est la quantité maximale qu'un agent peut envoyer dans un ordre — dans la simulation suivante, $q_0 = 10$.

La formalisation de cette définition ne révèle pas la simplicité élémentaire de l'idée qu'elle formalise, qui peut être résumée par :

- La direction moyenne $\tilde{d}$ est la direction prédominante (avec prise en compte des poids). S'il n'y a pas de direction prédominante, l'ordre retourné est l'ordre nul, $\emptyset$.
- Le prix de l'ordre moyenné est la moyenne pondérée des prix des différents ordres dont la direction est égale à la direction moyenne.
- La quantité moyenne est $\lceil |d| q_0 \rceil$, et est choisie de sorte qu'elle atteigne sa valeur maximale lorsque chaque ordre o_k, avec $1 \leq k \leq n$, a la même direction.

Notez que les quantités de commandes o_k n'ont pas d'influence sur l'ordre final. Nous avons fait ce choix parce qu'il nous permet de ne pas avoir à prendre en compte la quantité de l'ordre renvoyé par les comportements fondamentalistes qui était définie par un paramètre κ_i de confiance en soi, arbitraire et intrinsèque aux agents `DAT-F` et `DAT-IF`.

En faisant ainsi, il est donc possible de se passer de ce paramètre, ce qui minimise le nombre de paramètres caractérisant les agents. Notons également que de nombreux comportements proposés dans la littérature se contentent d'envoyer des ordres dont la quantité est égale à 1, car il est souvent difficile de définir une quantité qui a du sens. La méthode proposée ici permet de contourner cette difficulté : il suffit de définir des comportements dits élémentaires, sans définir la quantité des ordres envoyés par ces comportements. Ensuite, en appliquant notre méthode de moyenne pondérée, une quantité qui n'est ni arbitraire ni toujours égale à 1, est définie.

Enfin, notons que cette moyenne pondérée d'ordres a les propriétés que l'on peut attendre d'une moyenne :

- Si chaque o_k, pour $k \in [\![1,\ n]\!]$, possède la même direction d_0 et le même prix p_0, alors la direction moyenne et le prix moyen seront d_0 et p_0, respectivement.
- Le prix moyen est supérieur ou égal à $\min_k p_k$ et inférieur ou égal à $\max_k p_k$
- Pour tout k, si p_k est croissant, et si les autres p_l, pour $l \neq k$ restent constants, alors le prix moyen est aussi croissant.

5 Un agent déterministe, fondamentaliste et chartiste : DAT-IFC

5.1 Formalisation du comportement

La moyenne de plusieurs comportements étant définie, nous pouvons définir notre agent DAT-IFC qui est le résultat de la moyenne d'un DAT-IF et d'un chartiste. Cet agent possède 5 paramètres : (i) une agressivité $\alpha_i \in [0, 1]$, (ii) un niveau d'information $\iota_i \in \mathbb{N}$, (iii) une estimation de la valeur fondamentale $\pi_{i,t} \in \mathbb{N}^*$, (iv) un paramètre $\rho_i \in \{-1, 1\}$ représentant ce qu'il pense de la tendance des prix, et finalement (v) une valeur $\tau_i \in [0, 1]$ représentant l'importance donnée par les agents à la tendance des prix.

D'une part, avec les paramètres α_i et $\pi_{i,t}$, qui dépend de ι_i, l'agent détermine quel ordre $o^{(f)}_{i,t}$ il aurait envoyé s'il était purement fondamentaliste, donné par le comportement décrit dans la section 3.2. D'autre part, il détermine quel est l'ordre qu'il aurait envoyé s'il avait été purement chartiste, noté $o^{(c)}_{i,t}$ et défini en section 3.3. Cet ordre dépend de α_i, de ρ_i et de l'état du marché au temps t. L'ordre que l'agent va envoyer au final est l'ordre résultant $o_{i,t} = (o^{(f)}_{i,t}, 1 - \tau_i), (o^{(c)}_{i,t}, \tau_i)$: c'est une moyenne entre les deux ordres précédents, de sorte que le poids de l'ordre chartiste $o^{(c)}_{i,t}$ est τ_i — et donc le poids de l'ordre fondamentaliste $o^{(f)}_{i,t}$ est $1 - \tau_i$. Le comportement multi-jours est identique à ce qui est décrit dans la sous-section 3.2.

On notera qu'il s'agit d'un agent autonome : il n'a pas besoin d'autres agents pour négocier. Ceci est montré dans la sous-section suivante, car bien que nous n'utilisons que DAT-IFC sur notre marché, des prix sont continuellement fixés et nous observons les faits stylisés. Cette propriété distingue clairement le DAT-IFC des autres comportements proposés dans la littérature (ZIP, GD, GD, GDX ou AA), car notre agent illustre le fait qu'il existe des agents qui, tout en étant déterministes, peuvent fournir une certaine entropie au marché, sans avoir besoin d'agents tels que les ZIT pour leur permettre de négocier.

5.2 Faits stylisés

Les courbes suivantes sont obtenues avec un marché, simulé à l'aide d'un carnet à double enchères, avec 2 200 agents qui interagissent :

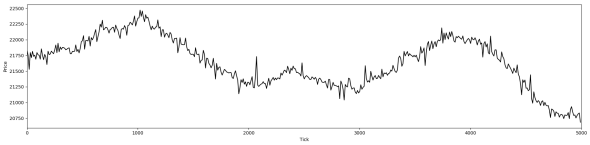


FIGURE 5 – Séquence de prix à la fin de chaque tick, en utilisant 2 200 DAT-IFC.

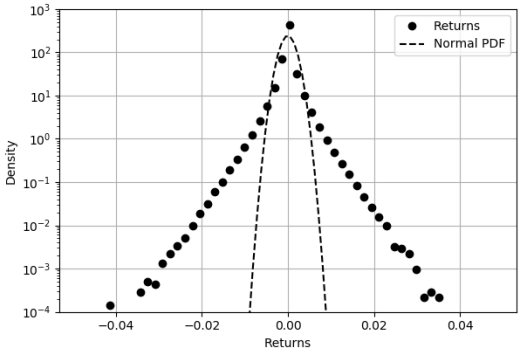


FIGURE 6 – Distribution logarithmique des rentabilités $(\log(p_{t+1}) - \log(p_t))_t$ (cercles) comparée à une distribution normale de même moyenne et écart-type (pointillés), sur une échelle semi-logarithmique.

Il y a un agent pour chaque valeur de $(\alpha, \iota, \tau, \rho) \in \{0, 0.1, \ldots, 1\} \times [\![0, 9]\!] \times \{0, \frac{1}{9}, \ldots, 1\} \times \{-1, 1\}$ — de cardinalité $11 \cdot 10 \cdot 10 \cdot 2 = 2\,200$. Le cours d'ouverture est fixé à 21 000. La simulation dure 100 jours de 50 ticks. A la fin de la simulation, 9 142 968 prix ont été fixés.

Les séries de prix, illustrées dans la figure 5, sont cohérentes avec les séries de prix observées sur les marchés financiers réels : en particulier, la signature plot est bornée par deux valeurs strictement positives et finies. La figure 6 montre la distribution des rentabilités, qui est leptokurtique, avec une kurtosis égale à 29,4. Pour finir, l'auto-corrélation des valeurs absolues des rentabilités logarithmiques sur une échelle log-log, représentée sur la figure 7, décroit selon une loi de puissance dont l'exposant est égal à -0.23 — On note qu'il se situe dans l'intervalle donné par [2].

5.3 Evolution de la richesse des agents

La rentabilité de l'agent i est définie comme le taux de croissance de la richesse d'un agent entre le début et la fin de la simulation, et est noté r_i. Notons $R_{\iota,\,\tau} = \{r_i \mid \iota_i = \iota \wedge \tau_i = \tau\}$ l'en-

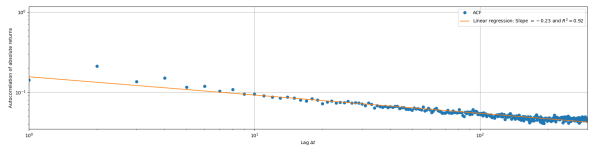


FIGURE 7 – Auto-corrélation des valeurs absolues des rentabilités logarithmiques sur une échelle log-log

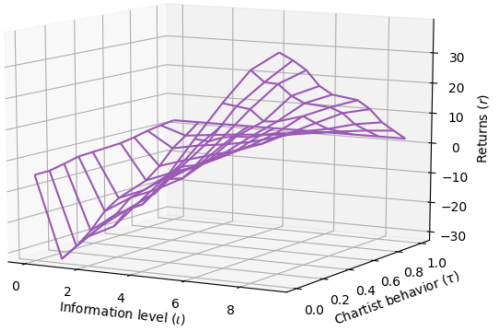


FIGURE 8 – Moyenne des rentabilités des agents pour un niveau d'information donné et une proportion de comportement chartiste. Chaque point correspond à une valeur de $(\iota,\ \tau,\ \mu(R_{\iota,\,\tau}))$.

semble des rentabilités des agents ayant un niveau d'information égal à ι avec une proportion de comportement chartiste égale à τ. Les figures 8 et 9 présentent respectivement les rentabilités moyennes des agents, et un encadrement de ces valeurs moyennes par l'écart-type, en fonction de leur niveau d'information et de leur proportion de chartistes.

Pour tous les points tels que $\tau = 0$, la courbe trouvée est similaire à la courbe trouvée par [11], ce qui n'est pas surprenant au premier abord : des agents tels que $\tau = 0.$ sont identiques aux agents de la section précédente. Toutefois, cela signifie que leur comportement est "stable" : malgré la présence d'agents avec un autre τ, des agents purement fondamentalistes très bien informés continuent d'avoir des rentabilités supérieures à celles des agents purement fondamentalistes mais mal informés.

De plus, le phénomène que nous souhaitions émerge de ce marché : d'une part, en moyenne, des agents purement fondamentalistes avec un haut niveau d'information sont parmi les agents qui ont les rentabilités les plus élevées ; d'autre part, il y a des agents qui ne sont pas bien in-

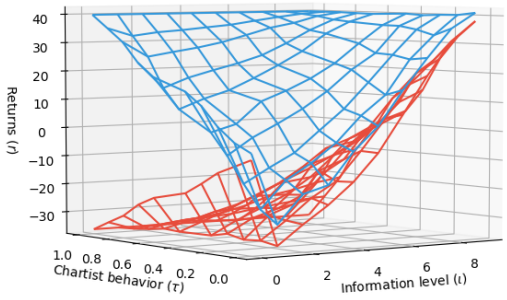


FIGURE 9 – Encadrement des rentabilités des agents. Chaque point correspond soit à une valeur de $(\iota,\ \tau,\ \mu(R_{\iota,\,\tau}) - \sigma(R_{\iota,\,\tau}))$ (points rouges) soit à $(\iota,\ \tau,\ \mu(R_{\iota,\,\tau}) + \sigma(R_{\iota,\,\tau}))$ (points bleus).

formés mais qui ont néanmoins de la chance et ont des rentabilités aussi élevées que les agents qui sont très bien informés. Cela se reflète dans le fait qu'il existe $\iota \leq 4$ et τ tel que $\mu(R_{\iota,\,\tau}) + \sigma(R_{\iota,\,\tau}) > \mu(R_{9,\,0}) - \sigma(R_{9,\,0})$. Par exemple, $\mu(R_{0,\,1}) + \sigma(R_{0,\,1}) = 39.82$ et $\mu(R_{9,\,0}) - \sigma(R_{9,\,0}) = 38.11$. Il y a donc une proportion non négligeable d'agents non informés et purement chartistes qui ont des rentabilités similaires à celles obtenues par des agents très bien informés et purement fondamentalistes.

Cette observation a été faite sur chaque simulation que nous avons réalisé : Nous avons exécuté 100 simulations de 50 jours de 50 ticks, et 66 d'entre elles étaient telles que $\mu(R_{0,\,1}) + \sigma(R_{0,\,1}) \geq \mu(R_{9,\,0}) - \sigma(R_{9,\,0})$, et parmi les 34 autres simulations, la différence relative entre ces deux grandeurs était inférieur à un dixième sur 27 d'entre-elles. Dans les 7 autres cas, cette différence était toujours inférieure à un quart.

Nous pouvons donc conclure que les chartistes chanceux et mal informés parmi les `DAT-IFC` ont des rentabilités du même ordre de grandeur que pour les agents très informés, permettant ainsi d'arriver à un marché qui respecte toutes les caractéristiques fondamentales d'un marché artificiel telles qu'elles sont définies dans notre introduction : les prix sont fixés en permanence, le marché obtenu respecte les faits stylisés les plus couramment étudiés — les queues épaisses, l'absence d'auto-corrélation des rentabilités, et la décroissance de l'acf des rentabilités absolues selon une loi de puissance — et il y a une différenciation comportementale des agents fonction

de leurs paramètres.

Cette différenciation conduit au fait que les agents les plus fondamentalistes et bien informés arrivent parmi les plus riches. Néanmoins, il y a des agents avec peu d'informations qui, étant chartistes et chanceux, font également partie de cette dernière catégorie. Cela signifie que le caractère aléatoire qui a émergé au niveau du marché a donné lieu à de l'aléatoire au niveau des agents.

6 Conclusion

Les simulations à base d'agents sont de plus en plus utilisées dans les recherches sur les marchés financiers car elles permettent un retour d'information sur les actions réalisées par les agents qui s'y trouvent et donc une étude détaillée des comportements possibles à la fois au niveau macroscopique et microscopique. Néanmoins, Les partisans de cette approche ont souvent recours à des comportements stochastiques, difficiles à reproduire et sans justification théorique comme le fameux Zero Intelligent Trader. Le succès de cet aspect stochastique est dû à la croyance populaire selon laquelle seul le hasard garantit que deux comportements identiques exécuteront des actions différentes, et par conséquent que le système général ne se bloquera pas automatiquement après un certain temps à cause d'une situation ne contenant que des acheteurs ou des vendeurs. De plus, peu d'auteurs proposent d'utiliser dans leurs modèles un système de fixation de prix proche de celui utilisé par les marchés réels, avec un carnet d'ordres à double enchères permettant de traiter des ordres limite avec direction, prix et quantités.

Dans cet article, nous avons proposé un comportement d'agent qui vise à envoyer des ordres limite réels traités par un carnet à double enchère, sur la base d'un comportement paramétrique assurant une différenciation comportementale fonction du niveau d'information des agents. Plus l'agent possède une information précise, plus il aura de chances de réussir. Nous montrons ici qu'il est possible de concevoir ce comportement dans une perspective complètement déterministe. D'une part nous montrons que l'aléatoire n'est pas nécessaire pour que nos agents évoluent différemment, d'autre part que ces agents permettent de constituer un système qui ne se bloque pas de lui-même, et qu'enfin, un ensemble d'agents de ce type garantit à la fois la reproduction des faits stylisés du domaine ainsi qu'un certain contrôle sur le succès des agents par le concepteur. Cet article, au carrefour de la finance et de l'informatique, constitue un pas en avant dans la compréhension des mécanismes profonds qui permettent l'émergence des faits stylisés ainsi que dans la modélisation du comportement de trading.

Références

[1] Hans M Amman, Leigh Tesfatsion, David A Kendrick, Kenneth L Judd, and John Rust. *Handbook of computational economics*, volume 2. Elsevier, 1996.

[2] Rama Cont. Empirical properties of asset returns : stylized facts and statistical issues. *Quantitative Finance*, 1(2) :223–236, 2001.

[3] Marco De Luca and Dave Cliff. Human-agent auction interactions : Adaptive-aggressive agents dominate. In *Twenty-second international joint conference on artificial intelligence, IJCAI*, 2011.

[4] Dhananjay K Gode and Shyam Sunder. Allocative efficiency of markets with zero-intelligence traders : Market as a partial substitute for individual rationality. *Journal of political economy*, 101(1) :119–137, 1993.

[5] Myron J Gordon and Eli Shapiro. Capital equipment analysis : the required rate of profit. *Management science*, 3(1) :102–110, 1956.

[6] Gew-rae Kim and Harry Markowitz. Investment rules, margin, and market volatility. *Journal of Portfolio Management*, 16(1) :45–52, 1989.

[7] Johann Lussange, Alexis Belianin, Sacha Bourgeois-Gironde, and Boris Gutkin. A bright future for financial agent-based models. *arXiv preprint arXiv :1801.08222*, 2018.

[8] Thomas Lux and Michele Marchesi. Scaling and criticality in a stochastic multi-agent model of a financial market. *Nature*, 397(6719) :498, 1999.

[9] Philippe Mathieu and Olivier Brandouy. A generic architecture for realistic simulations of complex financial dynamics. In *Advances in Practical Applications of Agents and Multiagent Systems*, pages 185–197. Springer, 2010.

[10] Philippe Mathieu and Rémi Morvan. A deterministic behaviour for realistic price dynamics. *Physica A : Statistical Mechanics and its Applications*, 525 :33 – 49, 2019.

[11] Bence Toth, Enrico Scalas, Jürgen Huber, and Michael Kirchler. The value of information in a multi-agent market model. *The European physical journal B*, 55(1) :115–120, 2007.

[12] Perukrishnen Vytelingum, Dave Cliff, and Nicholas R Jennings. Strategic bidding in continuous double auctions. *Artificial Intelligence*, 172(14) :1700–1729, 2008.

[13] Gerhard Weiss. *Multiagent systems : a modern approach to distributed artificial intelligence*. MIT press, 1999.

[14] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.

Modèle multi-agents pour la prédiction des risques en chirurgie

Bruno Perez[a] bruno.perez@univ-fcomte.fr
Christophe Lang[a] christophe.lang@univ-fcomte.fr
Julien Henriet[a] julien.henriet@univ-fcomte.fr
Laurent Philippe[a] laurent.philippe@univ-fcomte.fr

[a]Institut FEMTO-ST/CNRS
Université de Bourgogne-Franche-Comté, France

Résumé

La gestion des risques résultant des actions et des états des différents éléments composant une salle d'opération lors d'une intervention chirurgicale est une préoccupation majeure. La simulation basée sur les agents montre un intérêt à travers ses concepts d'interaction, d'interactivité et d'autonomie des différentes entités du simulateur. Il s'agit dans le cadre de notre étude d'implémenter un générateur d'alertes à l'écoute de l'évolution de différents paramètres appliqués aux agents du simulateur (fatigue humaine, efficience des matériels, taux d'infection ...). Cet article présente notre modèle, son implémentation et les premiers résultats obtenus. Notons que cette étude a permis aussi d'identifier plusieurs verrous scientifiques, tels que l'intégration des différents niveaux d'abstraction, le couplage d'espèces, la coexistence de plusieurs échelles dans un même environnement, l'agrégation d'indicateurs et la déduction d'alertes non prévisibles. Le raisonnement à partir de cas (RàPC) est un début de réponse relatif au dernier verrou évoqué, et sera abordé dans cet article.

Mots-clés : *système multi-agents, raisonnement à partir de cas, prédiction, modélisation, multi-échelles, chirurgie*

Abstract

Risk management resulting from the actions and states of the different elements making up a operating room is a major concern during a surgical procedure. Agent-based simulation shows an interest through its interaction concepts, interactivity and autonomy of different simulator entities. We want in our study to implement a generator of alerts to listen the evolution of different settings applied to the simulator of agents (human fatigue, material efficiency, infection rate ...). This article presents our model, its implementation and the first results obtained. It should be noted that this study also made it possible to identify several scientific obstacles, such as the integration of different levels of abstraction, the coupling of species, the coexistence of several scales in the same environment and the deduction of unpredictable alerts. Case-based reasoning (CBR) is a beginning of response relative to the last lock mentioned and will be discussed in this paper.

Keywords: *multi-agent system, artificial intelligence, predictive, modeling, multiscale, surgery*

1 Introduction

Dans les hôpitaux, la chirurgie est pratiquée dans des contextes de plus en plus innovants et performants. Les instruments sont connectés et entretenus avec toute la rigueur imposée par les normes d'hygiène et de sécurité. Tous les paramètres sont vérifiés avant et après la chirurgie (identité et état du patient, matériels de chirurgie, instruments de chirurgie, etc.) de manière à laisser le moins de place possible à des événements imprévus. Mais même dans ce contexte sécurisé, minimiser le risque d'un événement indésirable grave reste l'une des principales préoccupations des praticiens. En effet, malgré tous les protocoles mis en œuvre, le risque infectieux pour le bloc, même en cas de chirurgie propre, est de 1% selon le collège des universitaires des maladies infectieuses et tropicales (CMIT). Parmi les autres défaillances, figurent également les erreurs de dosage, la maladresse gestuelle et la fatigue humaine.

Dans ce contexte, nous visons à modéliser l'environnement et les ressources du bloc opératoire afin de prédire et de quantifier l'exposition du patient aux risques. Notre ambition est de créer un simulateur capable de générer un grand nombre de situations variées afin d'interpoler et de quantifier les évolutions possibles d'une situation donnée.

Notre étude orientée vers les modèles prédictifs composés d'entités non déterministes, pose entre autres la problématique de la détermination de seuils d'alerte en lien étroit avec l'acquisition de connaissances. En effet, est-il possible d'optimiser la définition de seuil et dans un même

temps d'agréger une base de connaissances dynamique à notre système ? Une réponse possible est l'agentification des seuils, enrichie par le couplage entre SMA et RàPC (raisonnement à partir de cas) qui résout un problème cible par analogie avec les connaissances antérieures (base de cas) évolutives. Nous proposons donc d'explorer, dans cet article, les pistes conceptuelles en relation avec notre positionnement. La littérature propre au couplage SMA/RàPC et en lien avec le domaine prédictif, sera quant à elle l'objet d'étude de la section 2 consacrée à l'état de l'art. Quelques définitions et rappels seront donnés en préambule dans cette section. Nous axerons dans la section 3 notre réflexion sur la caractérisation de la criticité, puis sur l'intérêt du couplage SMA/RàPC en réponse à notre problématique : "prédictivité des risques dans un contexte non déterministe". Nous préciserons notre positionnement et nos contributions appliqués à ces problématiques dans un contexte théorique. Nous exposerons nos premiers résultats dans la section 4 et nous les commenterons dans la suivante. La conclusion caractérisera la pertinence de nos choix et les perspectives à envisager.

2 Définitions et travaux connexes

Les systèmes prédictifs visent à délivrer de nouvelles informations extraites de simulations ou d'apprentissages passés. Ils dépendent principalement des données disponibles et de l'environnement. L'ère du big data est une piste actuelle majeure dans la production de ces dernières, que l'on souhaite à la fois exhaustives et fiables. La question se pose lorsque l'accès à ces données de masse est restreint. Notre problématique qui entre dans ce cas de figure nous oblige donc à considérer d'autres systèmes tels que les SMA couplés au RàPC capables de prédire avec peu de connaissances initiales. Nous allons dans la suite de cette section rappeler les principes et définitions de ces deux paradigmes.

Les définitions extraites de la littérature caractérisent les SMA comme des systèmes composés d'entités autonomes qui interagissent les unes avec les autres en fonction de certaines règles dans un environnement spécifique. Dans [1], Ferber définit les agents en tant qu'entités logicielles autonomes capables de percevoir (par messages, par capture de données ...) dans un environnement fermé au monde extérieur. Dans ce contexte, les agents cognitifs agissent avec réflexion et prise de conscience de leur environnement composé d'autres entités, contrairement aux agents réactifs qui réagissent seulement à des stimulations. Ainsi, dans le cas de problèmes complexes comme l'analyse épidémiologique et écologique des maladies infectieuses, des modèles standard basés sur les équations différentielles deviennent très rapidement inadaptées en raison d'un trop grand nombre de paramètres et sont supplantés par les SMA [2]. On observe aussi l'intégration de plus en plus fréquente du paradigme agent dans le renforcement de l'apprentissage automatique [3] que l'on ne peut dissocier de l'acquisition de connaissances. Elle caractérise l'autonomie des agents et peut être mise en œuvre au préalable (connaissances innées) ou être acquise par l'expérience. Ce deuxième cas concerne la capacité de l'entité à explorer une base de connaissances [4]. Ses performances dépendent principalement de la collecte d'informations qui enrichissent son apprentissage. L'architecture RàPC conçue selon ce principe, peut être une solution à cette exigence.

La notion d'acquisition des connaissances est au coeur des modèles dynamiques enrichis par l'expérience. Notre modélisation des entités non déterministes s'inscrit dans ce cadre et les valeurs des solutions ne sont pas nécessairement connues. Il est donc essentiel d'avoir une mémoire des cas précédents et une ontologie pour les structurer. Le raisonnement fondé sur des cas (RàPC) est un paradigme de l'intelligence artificiel basé sur l'élaboration de solutions par analogie avec les expériences passées et des connaissances générales dans le domaine d'application. Il est largement utilisé en médecine [5], dans la maintenance industrielle [6] ou dans l'analyse boursière [7]. Pour trouver une solution, le RàPC a besoin d'une base de cas résolus, de la caractérisation des similitudes et enfin de la connaissance des processus d'adaptation. Plusieurs ouvrages traitent de l'acquisition de connaissances, de la similarité ou de la prise de décisions avec SMA [8]. Les différentes approches prédictives structurent leur modélisation à partir de l'analyse d'expériences passées. La nature des données et les outils d'analyse distinguent ces concepts prédictifs. Deux approches principales sont envisagées dans la littérature pour l'utilisation du RàPC dans un SMA. Dans la première approche, le couplage RàPC avec SMA intègre dans l'agent la capacité à résoudre des problèmes à partir d'expériences extraites d'une base de cas. Dans [9] les problèmes sont résolus localement avec parfois une approche collaborative [10]. Dans [11], le raisonnement à partir de cas donne au système multi-agents la possibilité d'accéder à une base de données structurée qui accélère l'exploration des données. Ce mode de recherche

de cas construit pour un problème spécifique montre cependant ses limites sur des cas non similaires. De plus, le couplage RàPC/SMA apporte un caractère dynamique au RàPC (statique) alors que nous aimerions une modélisation dynamique enrichie par des expériences. En effet, la simulation multi-agents reste au cœur de notre travail dont l'enjeu consiste à "prédire dans un contexte non déterministe". Le RàPC s'intègre dès lors comme une source d'acquisition active des connaissances. La seconde approche propose plutôt un système d'aide à la décision dans un environnement multi-agents. L'un des principaux objectifs est de suggérer des réponses possibles à différents contextes contraints par de nombreux paramètres. Encore une fois, l'intérêt d'un raisonnement cas par cas réside dans sa capacité à trouver des solutions par analogie. Ceux-ci peuvent être ordonnés selon une hiérarchie (agent négociateur (AN), agent expert (AE)) [12] ou organisés en comités de collaboration. Le système peut explorer sa propre base de cas [13] , plusieurs bases fusionnées [14], ou des bases de données indépendantes. La recherche de cas similaires intègre différents principes d'initiation et d'apprentissage tels que les réseaux de neurones artificiels [15]. Ces outils pour l'aide à la décision qui intègrent une plus grande adaptabilité dans l'acquisition de connaissances sont l'une des possibilités que nous avons explorées. Ce type de couplage est toutefois limité aux bases de cas, tandis que d'autres types de données telles que les traces sont des sources d'information intéressantes. Indépendamment de l'interaction dynamique avec toutes les entités de son environnement (humain et matériel), notre modèle doit aussi intégrer l'anticipation de situations non déterministes à travers une approche multidimensionnelle des phénomènes. Nous proposons donc une nouvelle architecture qui aborde cette question dans la section suivante.

3 Un modèle SMA pour le bloc opératoire

Dans ce contexte non déterministe, et afin de répondre à notre objectif (pronostiquer les défaillances humaines et matérielles), nous avons décidé de modéliser le bloc opératoire en tant que système multi-agents. Ce type de simulation ouvre un grand nombre de perspectives et permet de mettre en évidence des risques minimisés ou ignorés jusqu'à présent, et enfin d'apporter une contribution en termes de sécurité en salle d'opération. UML (Unified Modeling Language) [16] et AML (Agent Modeling Language) ont été choisis comme langage de modélisation pour élaborer l'architecture de notre SMA.

L'architecture de notre modèle, qui intègre le paradigme BDI (Belief (croyance), Desire (désir), Intention) possède cinq espèces d'entités : Personnal, Material, Infection, Patient (singleton), Alert (singleton). La Figure 1 donne un descriptif des variables d'état de ces agents :

A titre expérimental, nous avons simulé l'évolution d'un site infectieux en parallèle avec la fatigue humaine en calibrant des cycles de simulation à 30 secondes.

Modélisation de la fatigue. La fatigue peut être modélisée en utilisant plusieurs méthodes d'acquisition de données. En mode non connecté, les données sont extraites à partir des fichiers statistiques ou issues d'une fonction. S'il s'agit d'une capture dynamique, les capteurs connectées (bracelet pour la fatigue, capteurs électrochimiques pour le taux d'infection) ainsi que les monitorings fourniront l'information. Dans notre cas, nous avons choisi de définir la croissance de la fatigue à partir d'une fonction exponentielle (paramétrable par l'utilisateur) car elle s'applique à des phénomènes continus et met en évidence la nature non linéaire de la fatigue. Elle est donnée par la relation : $f(t) = ae^{(k \times t)}$ où a est la valeur initiale, k la constante de croissance, et t la variable de temps. Dans la seconde phase du projet, une capture de données dynamique sera préférée. Notons que dans notre cas nous avons traité le niveau de fatigue d'un seul agent sachant que chaque intervenant peut posséder son propre seuil de risque.

Modélisation de l'infection. Les deux principaux types de contamination majeure observables sont, d'une part, les agents exogènes de la salle d'opération et, d'autre part, les agents endogènes. Notre système est capable de simuler l'évolution du taux d'infection des hôtes exogènes en fonction des agents décontaminants et du temps. Le modèle est initialisé en fonction des valeurs paramétrables suivantes :

- Number of susceptible host (nombre de particules saines) : 495 (en vert) ;
- Number of infected host (nombre de particules infectées) : 5 cfu (unité de formation de colonies en rouge) / m ^{3} ;
- Number of resistant host (nombre de particules résistantes) : en bleu.

Agent	Variables	Commentaire
Personal	*intention*	opérer un patient dans des conditions optimales de sécurité
	desire	utiliser les ressources humaines et matérielles (personal, material)
	belief	mesures utiles à la prise de décision (monitoring, alert)
	tiredness	taux de fatigue (échelle de 1 légèrement fatigué jusqu'à 5 épuisé)
	experience	junior, sénior
Material	*function*	fonctionnalité matérielle
	mat_tiredness	efficacité des matériaux (échelle de 1 efficace à 3 inefficace)
	infected	boolean
Infection	*type*	type d'agent infectieux (contaminant, résistant)
	local	a un impact sur une zone, sur le bloc opératoire ou sur tous les deux
	desire	définie en fonction de son type (contaminant, résistant)
	belief	apprécier le recouvrement avec le futur hôte
Patient	*state*	état de santé
	surgery_type	urgent, non-urgent, complexe,non-complexe
Alert	*intention*	prévenir une défaillance
	desire	alerte préventive (avant que la panne ne se produise)
	belief	écouter et suivre l'évolution des données influençant l'intervention chirurgicale
	level	seuils d'alerte

FIGURE 1 – Variables d'état des agents simulés

La proportion de particules infectées par rapport aux particules saines ($\frac{5}{495}$) indique un taux d'infection du site opératoire (ISO) qui avoisine 1 %. Ce pourcentage proposé par l'équipe médicale, apparait dans plusieurs études telles que [17][18]. La progression ou la régression de l'infection est liée au chevauchement entre les particules saines ou contaminées en fonction du temps.

Simulation de la fatigue couplée avec l'infection. Le couplage des deux défaillances permet d'avoir une vision globale des risques en salle d'opération. Il est alors possible de définir un niveau d'alerte corrélé à plusieurs risques. Le résultat de l'interaction entre les entités (infection, fatigue) est présenté dans la Partie 4.

Agent Alert. L'agent cognitif Alert est central dans notre architecture et joue les rôles de :

- centralisateur (centraliser les différents seuils d'alerte) ;
- adaptateur (gère un niveau d'alerte collective) ;
- régulateur (propose des solutions possibles).

Notre originalité par rapport aux approches décrites dans cette partie, réside tout d'abord dans le fait que nous avons conçu un système multi-échelles d'agents dans lequel les défaillances humaines (fatigue) et matérielles (efficience) ainsi que les maladies nosocomiales sont modélisées. Ainsi nous avons fait converger (centraliser) des types de données possédant des échelles et des métriques différentes. D'autre part, l'une des clefs de voute de notre modélisation prédictive est formalisée par l'agrégation des différents niveaux propres à chaque indicateur (variable d'état) écouté par le système. L'une des problématiques qui en découle porte sur :

- la définition ;
- la représentation ;
- la mesure ;

de ces seuils collectifs.

Nous proposons en réponse à ce questionnement, une approche multidimensionnelle. Il ne s'agit pas de comparer des courbes de tendance ou d'analyser des dispersions mais plutôt de repérer un point du plan affine (2D) ou de l'espace affine (3D). Dans le premier cas, 2 tendances évoluent dans un espace affine tel qu'à chaque coordonnée corresponde une cellule (agent) d'une grille capable d'interaction

avec son environnement (reconnaissance de recouvrement). L'exemple de la Figure 2 est une représentation graphique de la criticité entre deux variables (infectious_rate, human_tiredness) appartenant respectivement aux espèces *personnal* et *infection*. L'évolution de ces variables est dépendante des comportements et des capacités des agents (BDI) mais aussi de l'interaction entre ces deux espèces. Chaque point de coordonnées (infectious_rate, human_tiredness) se superpose à une cellule de la grille typée selon un niveau de criticité. Dans notre exemple, la position du point de coordonnées (human_tiredness, infectious_rate) indique une criticité de niveau acceptable (vert).

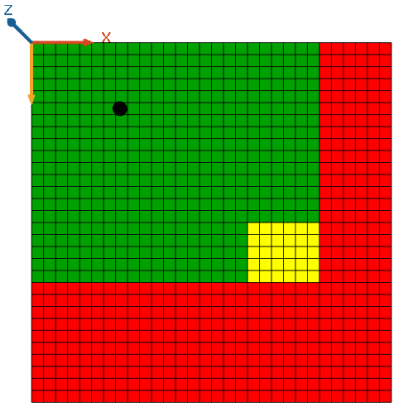


FIGURE 2 – Repérage du niveau de risque entre deux variables

L'intérêt et l'originalité se caractérisent par la possibilité d'agentifier des descripteurs et ainsi d'étendre la notion d'intervalle à celle d'espace. Notre positionnement est aussi envisageable dans un espace 3D comme le montre la Figure 3. L'exemple révèle le positionnement de trois coordonnées propres à trois variables d'état à l'instant t. On note que le point appartient à un volume typé comme "critique".

Dans un environnement non évolutif composé d'un nombre limité :
- de cellules (grid 2D) ;
- de volumes (espace affine 3D) ;
- d'espèces (personnal, infection) ;

une bibliothèque de seuils est une solution envisageable mais restrictive.

Nous suggérons pour palier cette limite de coupler notre SMA avec le raisonnement à partir de cas. Nous levons ainsi le verrou scientifique posé par la rigidité d'un système complexe sans acquisition de connaissances évolutives et adaptées. En effet, selon Jean Lieber [19],

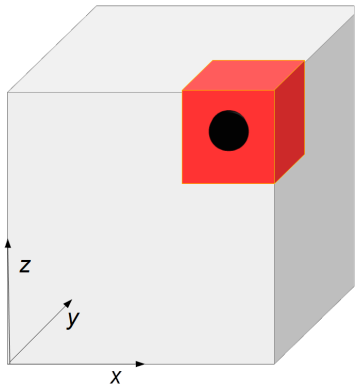


FIGURE 3 – Seuil criticité en 3D

"raisonner à partir de cas, c'est résoudre le problème cible en faisant appel à une base de cas, et par voie de conséquence en s'adaptant : RàPC : (cible,BaseDeCas) $\longmapsto$ Sol(cible) $\in$ Solutions "

Le RàPC se décompose selon les 5 étapes principales suivantes :
- L'**élaboration** qui consiste à formaliser un problème afin de le rendre exploitable par une machine.
- La **remémoration** qui consiste à résoudre le problème à partir de la base de cas :
  Remémoration : (cible,BaseDeCas) $\longmapsto$ (srce,Sol(srce)) $\in$ BaseDeCas.
- La **réutilisation** qui consiste à appliquer la solution du problème appartenant à la base de cas et éventuellement à l'adapter. L'adaptation consiste à résoudre cible à partir du cas remémoré (srce,Sol(srce)).
  Adaptation : (srce,Sol(srce),cible). $\longmapsto$ Sol(cible)
- La **révision** permettant à l'expert d'apporter des corrections à la solution cible proposée par le système.
- La **mémorisation** qui consiste à enrichir la base de cas avec de nouvelles solutions.

Élaboration. Appliqué à notre situation, les cas sont définis par l'application : $U \rightarrow R$ avec U caractérisant une séquence de quadruplets (E, A, V, t) et R le couple (état du système, préconisation). A chaque entité E, est affectée l'attribut A de valeur V à l'instant t (cycle). Dans l'exemple (nurse,fatigue,1.5,1200) $\rightarrow$ (N,Normal), la valeur 1.5 est affectée à l'attribut "fatigue" de l'agent "nurse" au cycle 1200. L'application retourne un état "N" ($\neg$ alert) et

une préconisation “Normal".

Remémoration. L'étape de remémoration qui consiste à trouver les cas les plus similaires, est basée dans un premier temps sur un filtrage des cas sources qui possèdent le même nombre de quadruplets que le cas cible. Dans un second temps, un calcul de similarité permettra de calculer avec une granularité plus fine le cas le plus similaire parmi ceux déjà extraits. Les quadruplets n'étant pas forcément homologues (attributs différents), nous avons choisi de comparer chaque quadruplet du cas cible à l'ensemble des quadruplets du cas source. Le calcul de distance vectorielle choisi est défini suivant la relation :

$$sim(\overrightarrow{C}, \overrightarrow{S}) = \frac{1}{n^2} \sum_{i=1}^{n} \sum_{j=1}^{n} \sqrt{\sum_{k=1}^{4} (1 - I)^2}$$

$I = \left(\frac{x_{ki}}{y_{kj}}\right)$ si $x_{ki} \leq y_{kj}$. Dans le cas contraire ($y_{kj} \leq x_{ki}$) $I = \left(\frac{y_{kj}}{x_{ki}}\right)$. x_{ki} et y_{kj} sont respectivement les valeurs associées à chaque élément du quadruplet cible ($\overrightarrow{C}$) et du quadruplet source ($\overrightarrow{S}$) parmi n quadruplets. Le fait de comparer les éléments en les quotientant nous exonère des problématiques d'échelles propres aux différents descripteurs. La colonne “simil" de la figue 6 donne un exemple de résultat issu de ce calcul.

Réutilisation. Cette étape permet de définir le cas cible. Dans le cas d'un traitement automatique, l'héritage de la solution du cas source viendra modifier les attributs des agents appartenant aux espaces affines de criticité (grid 2D, ou volume 3D). Cette mise à jour de la base SMA est opérée parallèlement à celle de la base du RàPC qui s'enrichit d'un nouveau cas. Lorsqu'une adaptation est nécessaire, (similarité suffisante mais imparfaite), nous appliquons la décomposition des tâches suivante proposée par Jean Lieber [19] :

chemin(srce,cible) =
$(p_0 \overset{r1}{\rightarrow} p_1 \overset{r2}{\rightarrow} p_2 \ldots p_{n-1} \overset{rn}{\rightarrow} p_n)$

A titre d'exemple, considérons le cas suivant :

cible p_0 =
(nurse_fatigue_limit∧nurse_dexterity_threshold),
srce p_n =
(surgeon_fatigue_limit∧surgeon_dexterity_limit).

La Figure 4 qui résume cette requête nous montre que l'agent interrogé n'appartient pas à la base de cas (surgeon $\neq$ nurse, mais qu'il appartient à une classe parent (personnal) avec laquelle il est possible de comparer le degré de similarité d'un agent de même nature (appartenant au même groupe). On peut donc associer à cette nouvelle classe les seuils collectifs hérités de la classe surgeon.

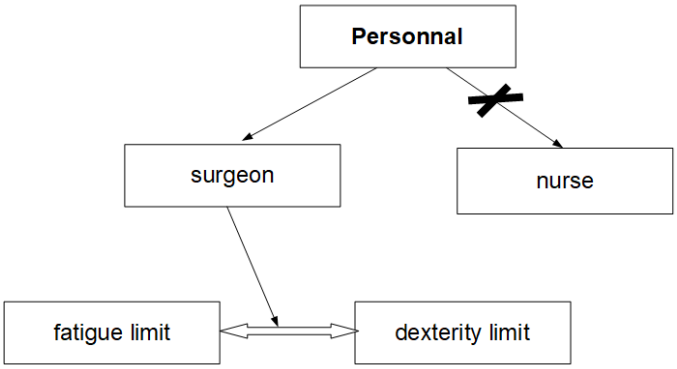


FIGURE 4 – Adaptation

Révsion et mémorisation. Chaque nouveau cas cible, après avoir éventuellement été corrigé par les experts du domaine, vient alimenter la base de cas et dans un même temps celle du SMA avec pour finalité l'optimisation de la recherche d'évènement similaires.

Implémentation Le choix de la plateforme destinée au développement de notre simulateur, s'est porté sur GAMA [20] qui permet de construire les modèles dans un environnement de développement intégré (IDE) incorporant le langage GAML (GAMA Modeling Language). Cette spécificité garantit l'enrichissement du modèle lors de son implémentation. La plateforme, riche de plusieurs composants, permet aussi de placer plusieurs modèles de visualisation dans une fenêtre d'affichage. Notons enfin, qu'il est possible de construire des modèles très complexes grâce aux outils de gestion de l'espace très performants dans différents environnements synchronisés à l'intérieur d'un espace continu de référence.

La section résultats qui suit présente les premiers résultats obtenus dans un contexte de simulation basé sur le risque de la fatigue infectieuse et humaine.

4 Résultats

Les premiers résultats obtenus concernent les alertes de la fatigue humaine et de la propagation infectieuse. Les niveaux d'alerte sont définis pour chaque agent en fonction de leurs objectifs, rôles et attributs. Les paragraphes suivants décrivent les résultats obtenus pour chacun de ces agents.

Modélisation de la fatigue

La courbe de fatigue humaine observée dans le graphique "Personnal Tiredness" de la Figure 5 est une fonction équivalente à une hyperbole dont l'évolution dépend du type défini en paramètre (sommeil, anxiété ...). Dans notre exemple nous avons considéré seulement deux individus sachant qu'il est possible d'assigner un type de fatigue à chaque personne qui travaille dans la salle d'opération. Le seuil d'alerte est défini par l'utilisateur.

Modélisation de l'infection.
Les résultats observables pour le risque infectieux montrent une corrélation entre l'évolution des agents contaminants et la diminution de la population saine. A cela s'ajoute l'interaction des agents immunitaires qui peuvent inverser cette tendance. Le déplacement autonome et aléatoire dans l'espace de ces entités permet de considérer tous les scénarios selon les paramètres définis initialement et modifiables lors de la simulation. Les données d'entrée sont des données statistiques et il est tout à fait possible de les obtenir avec une capture en temps réel en intégrant le Big Data. Comme la fatigue, le seuil d'alerte est défini par l'utilisateur.

Simulation du risque fatigue couplé au risque d'infection : une approche SMA
L'un des avantages de notre approche multi-agents est la possibilité de coupler les espèces comme indiqué dans le "Global Risk" de la Figure 5 où les risques "fatigue" et "infection" sont agrégés. Un seuil est défini par l'utilisateur sachant que la valeur du risque est un calcul combiné à tous les autres risques. Dans notre exemple, il est atteint alors que le seuil d'infection ne l'est pas. Une restrictivité apparait cependant lorsque les attributs à comparer nécessitent un nombre important de combinaisons. Ainsi, l'agrégation de quatre agents exigerait 2^4 combinaisons (ensemble des parties d'un ensemble). D'autre part, l'apprentissage du SMA, est dépendant d'expériences capitalisées.

Simulation du risque fatigue couplé au risque d'infection : une approche SMA/RàPC
Dans cette approche, les seuils limites ne résultent pas d'un paramétrage à priori, mais sont issus d'une base de cas enrichie par l'expérience. Les simulations alimentent cette base "au fil de l'eau" en fonction d'une amplitude prédéfinie (tous les 100 cycles dans notre exemple). Lors de cette phase d'apprentissage, la recherche de similitudes (étape remémoration du cycle RàPC) génère un nombre de cas éligibles qui sont ensuite comparés au cas cible et éventuellement adaptés. Un ajustement continu par les expert garantie l'efficience des résultats.

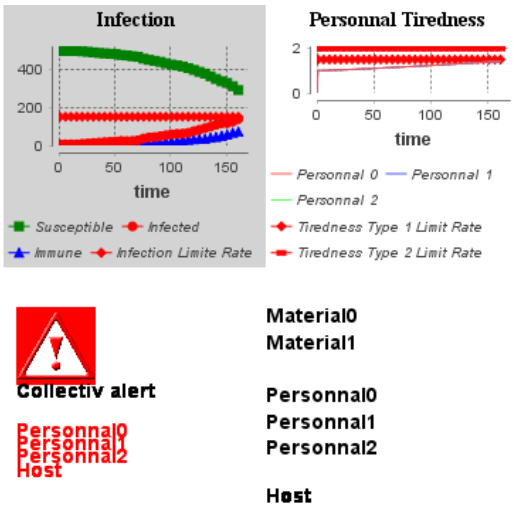


FIGURE 5 – Couplage infection et fatigue

La Figure 6 récapitule partiellement (tous les cas ne sont pas représentés) une recherche de cas. Parmi les cas de la base possédant 2 quadruplets le cas 35 est le meilleur candidat. La proximité du cas ($1.018 \ll 1.2$ (seuil d'acceptation paramétrable) génère l'adaptation : IBODE $\rightarrow$ CHIR et retourne à l'utilisateur l'état du système (alerte) et une préconisation (pause agent chir). Le seuil de 1.2 a été choisi car une similitude est vérifiée entre le cas cible et le cas source dans 95 % des cas lorsque $sim(\overrightarrow{C}, \overrightarrow{S}) \leq 1.2$. La mémorisation qui suit permet d'une part l'enrichissement de la base de cas d'autre part la mise à jour de la base de connaissances des agents du SMA.

Agent Alerte
L'agent centralisateur "alerte" intègre tous les risques, individuels ou collectifs. La sécurité dans la salle d'opération n'est plus seulement un ensemble d'alertes individuelles mais aussi collectives. Cette approche micro et macro permet d'envisager des niveaux critiques invisibles

Case Num.	E	A	V	t	E	A	V	t	Simil	Alert/ Preco.
0 (cible)	surgeon	fatigue	2.7	400	staphy	infection	270	400		
1	surgeon	fatique	3.2	700						
8	bistoury	fatigue	0.9	1200	nurse	fatigue	2.1	1200	1.482	N/Normal
35	nurse	fatique	2.5	300	staphy	infection	280	300	1.018	O/Pause Pers.

FIGURE 6 – Recherche de cas similaires

à ce jour. En effet l'alerte observée dans notre exemple est atteinte alors qu'individuellement elle ne l'était pas pour les agents.
Dans le diagramme de la Figure 7 nous avons représenté le résultat de 25 simulations du risque infectieux couplé à la fatigue humaine. Les valeurs expriment le déclenchement de l'alerte issue de l'agrégation du niveau de fatigue humaine couplé à celui du risque infectieux en fonction du temps (nombre de cycles). Ces alertes collectives se déclenchent alors qu'individuellement les seuils ne sont pas atteints. Par souci de clarté, 25 valeurs de simulation ont été choisies au lieu de 100, cependant, les tendances de la courbe sont similaires et nous montrent que les résultats varient peu lorsque les paramètres sont identiques.

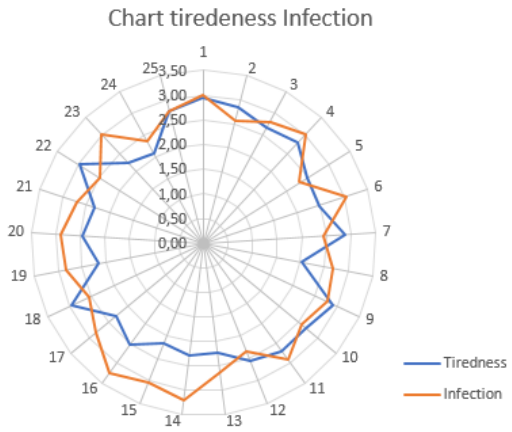


FIGURE 7 – Graphique 25 simulations couplant infection et fatigue

5 Discussion

Le paradigme multi-agents autorise une modélisation multi-niveaux et nous permet de considérer un même caractère selon plusieurs échelles. Ainsi par exemple, on peut distinguer au sein du bloc opératoire l'infection d'un organe (macro) mais aussi la quantité de particules infectieuses présentes dans l'air (micro). Concernant la prédictivité, l'agentification des zones de criticité a permis d'affiner le niveau des seuils. Les premières simulations révèlent une vraisemblance des résultats avec des tendances de courbes similaires. L'écart-type inférieur à 0,5 sur l'ensemble des valeurs confirme cette évaluation, permettant ainsi d'envisager la mise en place de réponses en termes de sécurité. Toutefois, les données simulées devront être comparées aux données réelles dans une prochaine étude.

Cependant des limites sont observables lorsque le nombre de quadruplets à comparer est supérieur à trois. En effet nous perdons la finesse du repérage offert par des espaces affines en 2D ou 3D. Le couplage SMA/RàPC est une réponse prometteuse. Actuellement, nous considérons la base de cas comme une ressource endogène au RàPC qui mériterait d'être étendue à d'autre systèmes d'acquisition de connaissances telles que les bases de traces ou certaines bases externes.

Dans la prochaine étape, nous allons, avec l'équipe médicale, définir les données du modèle et commenter les résultats. C'est une phase importante aussi bien en termes de validation des résultats qu'en termes de développement. C'est au cours de cette phase que nous déterminerons par exemple d'autres risques à modéliser, sachant que les seules limites sont d'ordre matériel. Nous définirons également le mode d'acquisition de données. Cela permettra au système de les exploiter en ligne de façon dynamique ou hors ligne. Dans le premier cas, nous considérerons le simulateur comme un générateur d'alertes, et dans le second cas comme un générateur de scénarios à des fins prédictives.

6 Conclusion

Dans cet article, nous avons présenté le simulateur que nous avons conçu et mis en œuvre afin

de prédire les risques liés à la fatigue humaine et matérielle ainsi qu'aux infections nosocomiales en chirurgie. Les résultats ont prouvé que notre modèle peut gérer de telles données dans un environnement multi-échelles. De même, notre approche multidimensionnelle des niveaux de criticité, a ouvert de nombreuses perspectives en termes d'optimisation des résultats (granularité). Les limites observées au delà d'un espace 3D ont ouvert notre champ d'investigation vers le raisonnement à partir de cas qui optimise l'acquisition de connaissances de notre simulateur. Notre système, a également prouvé son efficacité puisque la plupart des situations simulées étaient correctes. Néanmoins, nous envisageons dans une seconde phase, d'une part d'intégrer notre système dans en mode dynamique (connecté aux capteurs), d'autre part de concevoir un module de propositions correctives.

Remerciements

Les auteurs remercient le Pr. Auber et le Dr Boulard du service de chirurgie pédiatrique du Centre Hospitalier Régional Universitaire de Besançon pour leur expertise apportée dans le domaine d'application de l'étude.

Références

[1] Jacques Ferber. Multi-agent systems : towards a collective intelligence. *Reading : Addison-Wesley*, 1998.

[2] Benjamin Roche, Jean-François Guégan, and François Bousquet. Multi-agent systems in epidemiology : a first step for computational biology in the study of vector-borne disease transmission. *BMC bioinformatics*, 9(1) :435, 2008.

[3] Jelle R Kok and Nikos Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7(Sep) :1789–1828, 2006.

[4] Ying Shen. *Élaboration d'ontologies médicales pour une approche multi-agents d'aide à la décision clinique*. PhD thesis, Paris 10, 2015.

[5] Nabanita Choudhury and Shahin Ara Begum. A survey on case-based reasoning in medicine. *Int. Journal of Advanced Computer Science and Applications*, 7(8) :136–144, 2016.

[6] A Camarillo, J Ríos, and KD Althoff. Cbr and plm applied to diagnosis and technical support during problem solving in the continuous improvement process of manufacturing plants. *Procedia Manufacturing*, 13 :987–994, 2017.

[7] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, Marco De Gemmis, and Georgios Lekkas. Personalized finance advisory through case-based recommender systems and diversification strategies. *Decision Support Systems*, 77 :100–111, 2015.

[8] Mihalis Giannakis and Michalis Louis. A multi-agent based system with big data processing for enhanced supply chain agility. *Journal of Enterprise Information Management*, 29(5) :706–727, 2016.

[9] Mohammed Ahmed Jubair, Salama A Mostafa, Aida Mustapha, and Hanayanti Hafit. A survey of multi-agent systems and case-based reasoning integration. In *2018 International Symposium on Agent, Multi-Agent Systems and Robotics (ISAMSR)*, pages 1–6. IEEE, 2018.

[10] Santi Ontañón and Enric Plaza. Learning and joint deliberation through argumentation in multiagent systems. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*. ACM, 2007.

[11] Moamin A Mahmoud, Mohd Sharifuddin Ahmad, Mohd Zaliman M Yusoff, and Aida Mustapha. Context identification of scientific papers via agent-based model for text mining (abm-tm). In *New Trends in Computational Collective Intelligence*, pages 51–61. Springer, 2015.

[12] Pancho Tolchinsky, Sanjay Modgil, Ulises Cortés, and Miquel Sànchez-Marrè. Cbr and argument schemes for collaborative decision making. *Frontiers in Artificial Intelligence and Applications*, 144 :71–82, 2006.

[13] Desirée Vögeli, Sebastian Grabmaier, Matthias Jüttner, Michael Weyrich, Peter Göhner, and Wolfgang M Rucker. Intelligent and distributed solving of multiphysics problems coordinated by software agents. In *International Conference on Agents and Artificial Intelligence, Madeira, Portugal*, 2018.

[14] David B Leake and Raja Sooriamurthi. Managing multiple case bases : Dimensions and issues. In *FLAIRS Conference*, pages 106–110, 2002.

[15] Cristian I Pinzón, Javier Bajo, Juan F De Paz, and Juan M Corchado. S-mas :

An adaptive hierarchical distributed multi-agent architecture for blocking malicious soap messages within web services environments. *Expert Systems with Applications*, 38(5) :5486–5499, 2011.

[16] OMG Adopted Specification. Uml 2.0 superstructure specification.

[17] Jose L Alfonso-Sanchez, Isabel M Martinez, Jose M Martín-Moreno, Ricardo S González, and Francisco Botía. Analyzing the risk factors influencing surgical site infections : the site of environmental factors. *Canadian Journal of Surgery*, 60(3) :155, 2017.

[18] Arnaud Florentin. *Construction d'outils nécessaires au suivi et à la maîtrise de la qualité de l'air dans un établissement de Santé : exemple de la démarche qualité du Service d'Hygiène du CHU de Nancy*. PhD thesis, UHP-Université Henri Poincaré, 2011.

[19] Jean Lieber. *Contributions à la conception de systèmes de raisonnement à partir de cas*. Habilitation à diriger des recherches, Université Henri Poincaré-Nancy I, 2008.

[20] Edouard Amouroux, Thanh-Quang Chu, Alain Boucher, and Alexis Drogoul. Gama : an environment for implementing and running spatially explicit multi-agent simulations. In *Pacific Rim International Conference on Multi-Agents*, pages 359–371. Springer, 2007.

CogLogo: une implémentation de MetaCiv pour NetLogo

François Suro suro@lirmm.fr
Jacques Ferber ferber@lirmm.fr
Tiberiu Stratulat stratulat@lirmm.fr

LIRMM, Université de Montpellier, CNRS, Montpellier, France

Abstract

CogLogo est une extension pour NetLogo qui implémente les principes de MetaCiv. Le but est de proposer un framework pour la modélisation du système de décision d'un agent social intégrant un mécanisme de renforcement.

Mots-clés : *ABS, Agent Based Simulation, multi-agent systems, MASQ, cognitons, MetaCiv, CogLogo, NetLogo*

1 Introduction

MetaCiv est un framework générique de SMA reposant sur le méta-modèle MASQ, pour l'implémentation de systèmes sociaux complexes.

Cette architecture est constituée de deux éléments fondamentaux : les cognitons et les plans.

1. **Les cognitons** sont des "unités cognitives élémentaires" [1] dynamiques, pouvant être ajoutées ou retirées de l'esprit de l'agent. Ils peuvent représenter des croyances, des savoir-faire ou des percepts. L'ensemble des cognitons présents dans l'esprit d'un agent à un instant donné constitue son état mental.
2. **Les plans** représentent les différentes activités que peut réaliser un agent. Chaque plan est lui-même constitué d'une succession d'actions, plus ou moins complexes. Afin de guider la prise de décision de l'agent, à chaque plan est associé un poids qui sera déterminé par les cognitons présents dans l'esprit de l'agent. De manière générale, un agent choisira d'appliquer un plan ayant le poids le plus élevé possible. Néanmoins, l'architecture de MetaCiv ne fixe pas le mécanisme précis du processus conatif, et il est possible d'en envisager plusieurs en fonction des objectifs du SMA. Par exemple, on peut choisir systématiquement le plan de poids le plus fort, effectuer un tirage aléatoire qui favorise les poids forts, etc.

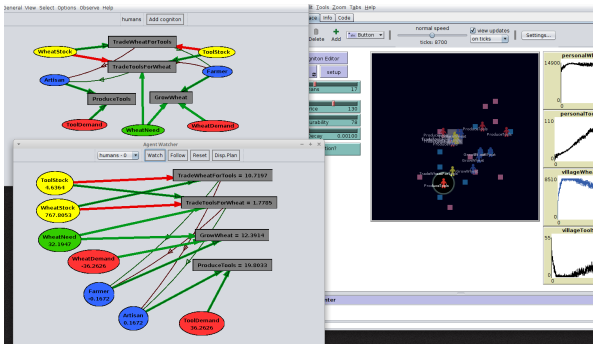


Fig. 1 – en haut à gauche, l'éditeur de schéma cognitif, à bas à gauche la fenêtre d'observation d'un agent montrant le poids des cognitons et la valeur calculée des plans.

2 L'extension CogLogo

CogLogo [1] est une extension NetLogo [2] écrite en Java qui implémentes l'architecture à base de cogniton de MetaCiv [3]. CogLogo fournit une **interface graphique pour définir un schéma cognitif**, le rapport cogniton-plan à travers les liens d'influence. Le schéma cognitif prends en charge le processus de décision qui aboutit au choix du plan à exécuter. L'implémentation du Plan se fait sous forme d'une procédure NetLogo. CogLogo fournit aussi un **mécanisme de renforcement automatique**, sous forme de liens de feedback, un **choix de règles pour le processus de décision**, poids maximum ou stochastique, ainsi que la possibilité d'**observer l'évolution de l'esprit de chaque agent en temps réel**. Coglogo est un projet open source disponible à cette adresse :

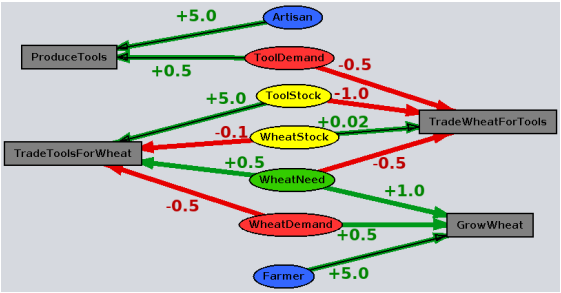
`https://github.com/suroFr/CogLogo`

3 Mise en place d'une simulation avec CogLogo

Un des exemples de simulation fournit avec CogLogo est un modèle Artisan-Agriculteur. Dans ce modèle les agents doivent produire de la nourriture pour survivre, ils peuvent aussi produire des outils à durabilité limitée qui multiplient le

rendement de leur activité de production de nourriture. Les agents ont la possibilité d'échanger de la nourriture contre des outils ou des outils contre de la nourriture. On souhaite observer l'émergence d'une spécialisation des agents en artisans et agriculteurs.

Le schéma cognitif des agents est le suivant :



De plus nous utilisons le système de renforcement (*feedback links*) qui augmente le poids des cognitons *Artisan* et *Farmer* si l'agent réussit une transaction d'outils contre blé et blé contre outils respectivement (ce qui caractérise un professionnel est la capacité à commercialiser son produit, et non à produire).

Ci-dessous quelques fonctions qui permettent à CogLogo de s'interfacer avec NetLogo :

```
;; choose-next-plan renvoie une chaîne de
    caractères, le nom d'une fonction, qui
    peut être appelée par la primitive
    "run"
run coglogo:choose-next-plan

;; set-cogniton-value change la valeur de
    cogniton fournit en premier paramètre
    à la valeur fournie en second
    paramètre.
coglogo:set-cogniton-value "ToolStock"
    toolStock
```

4 Résultats de la simulation

Voici le résultat d'une exécution de la simulation avec 25 agents sur une durée de 500 000 ticks

Les agents en rouge produisent des outils, en bleu de la nourriture. Les agents en jaune sont en train de réaliser un échange au marché (village).

La couleur de l'habitation d'un agent représente la moyenne de l'activité conduite par l'agent pendant les 25 000 derniers ticks : En rouge, production d'outils, en bleu production de nourriture, la teinte intermédiaire de violet indique la prédominance d'une activité ou l'autre.

Dans cette exécution de la simulation le manque

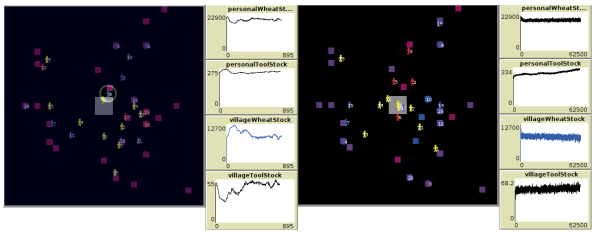


Fig. 2 – à gauche : état initial, à droite : tick 27 000

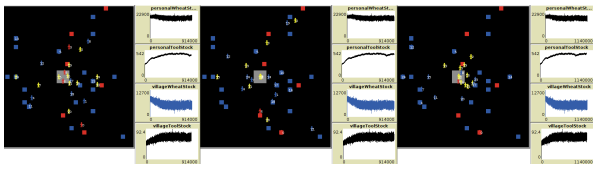


Fig. 3 – de gauche à droite : tick 400 000, 450 000 et 500 000

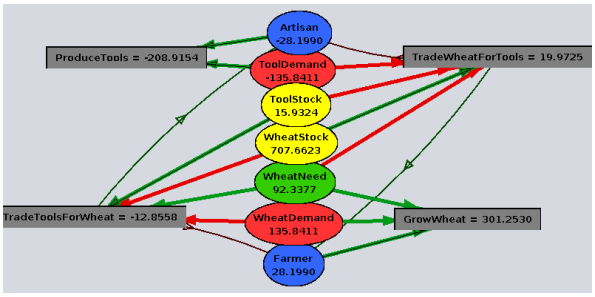


Fig. 4 – état interne d'un agent spécialisé en agriculteur

d'outils a été comblé par une spécialisation progressive de certains agents en artisans, mais dans d'autres exécutions, il est arrivé que trop d'agents se spécialisent en artisans, causant une surproduction d'outils. Dans ce cas, à cause du manque de succès dans l'échange d'outils contre de la nourriture, les agents les moins spécialisés parmi les artisans ont progressivement changé leur activité vers l'agriculture, causant un retour à l'équilibre du marché.

Références

[1] Suro, F., *CogLogo*, SMILE : Système Multi-agent, Interaction, Langage, Évolution., Laboratoire d'informatique, de robotique et de microélectronique de Montpellier., France, 2017, https ://github.com/suroFr/CogLogo

[2] Wilensky, U., *NetLogo*, Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL., 1999, http ://ccl.northwestern.edu/netlogo/

[3] Ferber, Jacques and Nigon, Julien and Maillé, Gautier and Stratulat, Tiberiu., *MetaCiv : un framework multi-agent basé sur MASQ pour modéliser des sociétés humaines*, JFSMA : Journées Francophones sur les Systèmes Multi-Agents, 2014, https ://hal-lirmm.ccsd.cnrs.fr/lirmm-01463384

Répartition des tâches pour la collecte de colis : démonstration

Maxime Morge,
maxime.morge@univ-lille.fr

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

Résumé
Comme les problèmes de la patrouille ou de la recherche de chemin multi-agents, le problème de la collecte de colis par des agents coopératifs [6] constitue un banc d'essai pour étudier, raffiner, expérimenter et évaluer les algorithmes multi-agents. Nous proposons ici un comportement d'agent pour négocier la répartition des tâches au cours de collecte.

Mots-clés : *Résolution collective de problème, Négociation, Modèles de comportement agent*

1 Collecte de colis

Problème. L'environnement que nous considérons est une grille de cellules (cf. Fig. 1). Chaque cellule contient au plus une entité, qu'elle soit passive (un objet) ou active. Quelque soit leur poids, les colis dispersés comme la destination où ils doivent être déposés sont des objets. Une entité active est soit le corps d'un agent soit une équipe, c.-à-d. un ensemble de corps d'agents. La taille d'une entité active, c.-à-d. le nombre de corps qui la constitue, détermine le poids maximal des colis qu'elle peut transporter.

Action. À chaque pas de simulation, les actions des entités actives, éventuellement conflictuelles, sont simultanées. Chaque entité active soumet une influence, c.-à-d. une action qu'elle souhaite réaliser [3]. La réaction du simulateur, qui résout les conflits et (in)valide les influences, génère une nouvelle instance de problème ou clôt la simulation. Les actions réalisables par une entité active sont : (a) le déplacement vers l'une des cases adjacentes, (b) le chargement d'un colis si l'entité en a la capacité, (c) le dépôt d'un colis si l'entité chargée est adjacente à la destination, (d) l'agrégation avec une autre entité active consentante, ou (e) la dissolution [2].

Effort. Arbitrairement, chaque action a un coût unitaire qui représente l'effort nécessaire pour la réaliser. Le coût de la collecte d'un colis par une entité active correspond (a) à la distance la plus courte pour atteindre une case adjacente à ce colis, (b) au chargement, (c) à la distance la plus courte pour atteindre une case adjacente à la destination, et (d) au dépôt du colis. Le nombre d'actions accomplies par les corps d'agent correspond à l'effort que ces derniers ont réalisé et le nombre de pas de simulation aux efforts consentis par le corps le plus sollicité (en anglais, *makespan*).

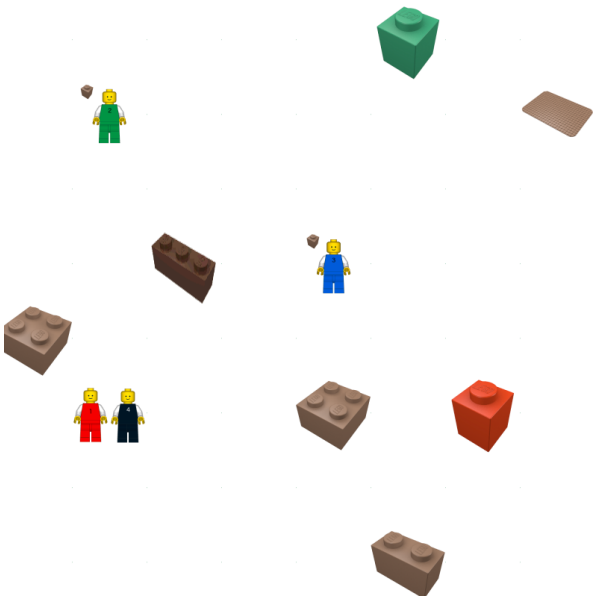


FIGURE 1 – Problème de la collecte multi-agents de colis [5]. Un colis ciblé par une entité est coloré.

Tâches. Pour être tous collectés, chaque colis doit être ciblé par une entité active. Il est important de noter que le coût d'une tâche (i.e. un colis à collecter) ne correspondent pas nécessairement aux efforts effectivement observés. Certaines actions peuvent échouer (e.g. un déplacement vers une case occupée) ou être non sollicitée (e.g. une entité refoulée par une autre). Afin de réduire le *makespan*, les agents peuvent être amenés à échanger des tâches au cours de la collecte à cause de ces aléas d'exécution.

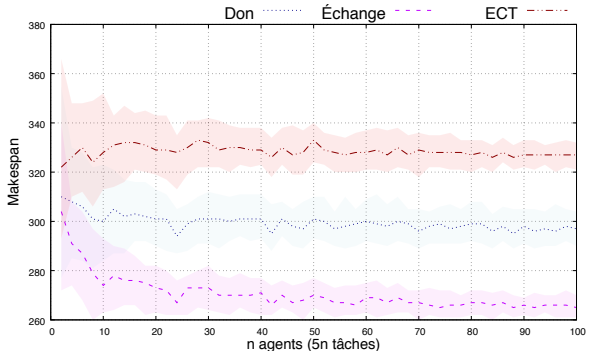


FIGURE 2 – *Makespan* obtenu par les stratégies de don/échange et par l'heuristique ECT

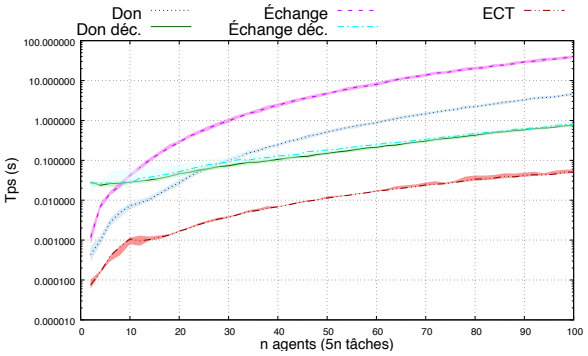


FIGURE 3 – Temps d'exécution d'ECT et de nos stratégies avec une échelle logarithmique

2 Réallocation de tâches

Nous proposons un modèle abstrait de réallocation des tâches pour l'équilibrage de charge [4]. L'approche centrée individu nous permet d'introduire une variété de stratégies et de règles de décision sociale pour l'élicitation des préférences individuelles et collectives. Dans le contexte de la collecte de tâches, nous proposons un nouvel algorithme multi-agents pour la négociation biunivoque de dons et d'échanges de tâches afin de minimiser collectivement le *makespan*. Un agent négocie en utilisant ses connaissances locales sur les coûts des tâches et ses croyances sur la charge de travail de ses pairs. Ces croyances sont mises à jour au cours de la négociation. Notre modélisation nous permet d'exprimer l'algorithme sous la forme d'un ensemble de comportements d'agents afin de distribuer leur exécution dans un environnement de calcul haute performance. Ces comportements déterminent les tâches à réaliser et celles à négocier. Le processus de négociation est itératif et concurrent à la réalisation des tâches. Les négociations sont bilatérales, concurrentes et répétées.

Nous avons réalisé une évaluation empirique rigoureuse qui compare notre algorithme aux algorithmes centralisés existants, par exemple l'heuristique ECT [1] - en anglais, *earliest completion time*. Cette évaluation montre que la réaffectation des tâches réduit significativement le *makespan* (cf. Fig. 2). Au delà des simples dons, les échanges de tâches améliorent le *makespan*. De plus, le temps d'exécution de nos stratégies est réduit quand elles sont distribuées (cf. Fig. 3). On observe que la durée d'exécution de la résolution décentralisée n'est pas pénalisée par la stratégie d'échange.

Perspectives. Nous souhaitons proposer une stratégie pour que les agents négocient la formation de coalitions afin de transporter des colis qu'ils ne peuvent pas collecter seul :
Virtus unita fortior.

Remerciements. Je remercie Antoine Nongaillard et les relecteurs qui, par leur remarques, ont permis d'améliorer cet article.

Références

[1] Ibarra, O. H. and Kim, C. E. (1977). Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors. *JACM*, 24(2) :280–289.

[2] Mathieu, P., Picault, S., and Secq, Y. (2016). Design Patterns pour les environnements dans les simulations multi-agents. *RIA*, 30(1-2) :133–158.

[3] Michel, F. (2007). Le modèle IRM4S. de l'utilisation des notions d'influence et de réaction pour la simulation de systèmes multi-agents. *RIA*, 21(5-6) :757–779.

[4] Morge, M. (20 février 2019a). *Scalable Multi-Agent Task Allocation*. `https://github.com/cristal-smac/ScaMATA`.

[5] Morge, M. (20 février 2019b). *Scalable Situated Multi-Agent Task Allocation*. `https://github.com/maximemorge/ScaSMATA`.

[6] Weyns, D., Helleboogh, A., and Holvoet, T. (2005). The Packet-World : A Test Bed for Investigating Situated Multi-Agent Systems. In *Software Agent-Based Applications, Platforms and Development Kits*, pages 383–408. Springer.

Gestion dynamique supervision et optimisation de microréseaux urbains pour l'autonomie (GYSOMATE)

Y. Gangat [a,b] yassine.gangat@univ-reunion.fr
T. Issoufaly [a] taher.issoufaly@univ-reunion.fr
D. Grondin [a] dominique.grondin@univ-reunion.fr
N. Coquillas [a] nicolas.coquillas@univ-reunion.fr
M. Benne [a] michel.benne@univ-reunion.fr
D. Payet [b] denis.payet@univ-reunion.fr
J-P. Chabriat [a] jean-pierre.chabriat@univ-reunion.fr

[a] Laboratoire d'Energétique, d'Electronique et Procédés (LE²P), [b] Laboratoire d'Informatique et de Mathématiques (LIM), Université de La Réunion, Saint Denis, Réunion, France

1 Introduction

Le concept de système de gestion de l'énergie (SGE) répond aux défis énergétiques relatifs aux sources d'énergie renouvelables (ENR) et à la protection de l'environnement. L'expansion des ressources énergétiques distribuées représente un changement de paradigme dans les réseaux électriques, rendant les micro-unités de production et de stockage pertinentes, à la fois en termes de développement durable et d'énergie à la demande. La fiabilité des réseaux interconnectés de petite taille et isolés à grande échelle dépend d'un SGE efficace. [1]

La Réunion, située dans une zone tropicale, constitue un terrain fertile pour la recherche dans ce domaine, notamment avec l'objectif 2030 d'atteindre 100 % d'ENR. Cependant, le développement des réseaux intelligents à la Réunion nécessitera des travaux de R&D adaptés aux caractéristiques de l'ile. Cela ne peut pas être réalisé uniquement avec des théories et des pratiques de simulation, excluant ainsi certaines situations dépendant de la réalité.

Nous proposons une plateforme de test pour simuler et émuler un réseau intelligent entièrement personnalisable.

2 La plateforme

Notre objectif n'est pas de fournir la meilleure méthode de prévision ou de SGE, mais plutôt de permettre la comparaison entre différentes configurations. Cette plateforme a été construite à partir des 5 modules suivants : un simulateur temps réel (RTS), un nanogrid, une interface homme-machine (IHM), une base de données (BDD), et un SMA.

2.1 RTS, Nanogrid, IHM & BDD

RTS : De nombreux travaux dans le domaine du *model-based design* font appel à des simulations Hardware-In-the-Loop, il s'agit d'un RTS mettant en œuvre la modélisation de l'environnement étudié et des équipements physiques en boucle fermée : les entrées et les sorties du système testé sont connectées à un ordinateur qui reproduit le fonctionnement de l'environnement. Le RTS est réalisée avec la plateforme RT-LAB qui convertit des modèles Matlab-Simulink pour les implémenter dans un environnement en temps réel sur le simulateur OP4510, puis les exécute.

Nanogrid : Comme il est complexe de connecter des unités de production, de stockage et de consommation (dû à des aspects juridiques et techniques) nous avons mis en place un nanogrid (12V), construit à partir d'émulateurs pour les sources ENR et pour les appareils consommant de l'énergie, de dispositifs de stockage d'énergie, des capteurs et actionneurs, et de quelques appareils électriques contrôlables au sein du laboratoire (climatisation, chauffe-eau).

L'architecture déployée est similaire à celle des cas réels étudiés. Les émulateurs, capteurs et actionneurs seront déployés à l'aide de microcontrôleurs Arduino et Raspberry Pi. Peu coûteux et ouverts, ils offrent une grande liberté de développement. La liaison avec d'autres équipements électriques se fait via le SGE et plus particulièrement le SMA. Ce nanogrid peut être connecté à un autre nanogrid virtuel simulé sur le RTS. L'introduction de ce nanogrid nous apporte plusieurs avantages. Cela nous permet de visualiser les effets en direct du SGE et d'interagir de manière réelle avec le système.

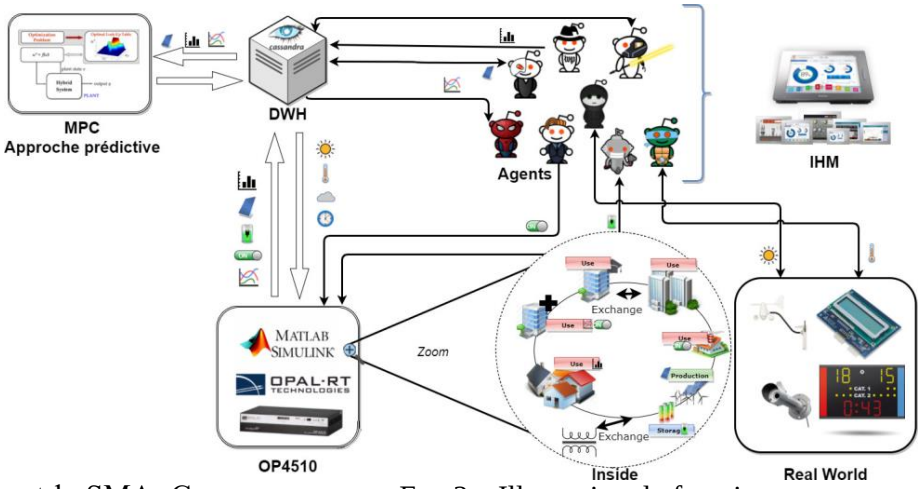


FIG. 2 – Illustration du fonctionnement

IHM : est initialisée par un fichier XML contenant la configuration du scénario (ID, nature, informations de chaque agent et de chaque élément) et est alimentée par les données et les journaux des agents. Elle peut également envoyer des commandes ou des demandes à certains agents.

BDD : Nous avons utilisé InfluxDB pour stocker nos données provenant de nos partenaires, de nos simulations et émulations et de notre laboratoire (capteurs, prévisions, etc.).

2.2 Le système multiagent

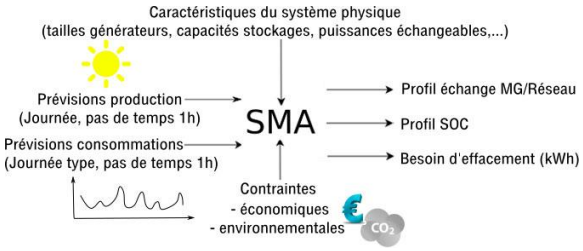


FIG. 1 – Outils de prédiction

La puissance de notre architecture SMA réside dans le fait qu'il n'est pas uniquement responsable du SGE, comme dans la plupart des SMA tels que Powermatcher et Volttron[2]. Nous avons choisi une architecture qui permet au SMA d'être au centre de notre plateforme. En effet, il sert d'outils de prédictions (Fig1), mais il est aussi le moyen de communication entre chaque module. Les agents interagissent avec les capteurs du monde réel, les objets simulés/émulés, l'utilisateur et les données (Fig2). Nous avons utilisé deux *frameworks* JADE[3] et SKUAD *(Software Kit for Ubiquitous Agent Development)*, ce dernier est développé au LIM.

3 Perspectives

Cette plateforme permet le développement de SGE pour des systèmes distribués. Des algorithmes de prise de décisions dans des environnements incertains et multiacteurs peuvent être développés notamment pour l'étude de l'autoconsommation collective d'énergie photovoltaïque[4].

Remerciements

Ces travaux ont été réalisés dans le cadre du projet GYSOMATE, financé par le FEDER, l'État et la Région Réunion.

Références

[1]Carrasco, J. M., et al. (2006). *Powerelectronic systems for the grid integration of ENR:A survey*. IEEE Transactions on industrial electronics.

[2]Zandi, H., et al. (2018). *Home Energy Management Systems: An Overview*. Oak Ridge National Lab.(ORNL), United States.

[3]Bellifemine, F., et al. (2002). *Jade programmer's guide*. Jade version, 3, 13-39.

[4]Y. Gangat,et al.(2018)*Simulation&Emulation plat form for smartgrid technologies*.IEEE,ICCEREC