Formation de coalitions stables dans un contexte non-déterministe et instable

Pascal François Faye, Samir Aknine, Onn Shehory, Mbaye Sène

Formation de coalitions stables dans un contexte non-déterministe et instable

Pascal Francois FAYE[1,2] Samir AKNINE[1] Onn Shehory[3] Mbaye SENE[2]
[1] LIRIS UCBL, FRANCE [2] LID UCAD, SENEGAL [3] IBM Lab. Haifa, Israel
pascal-francois.faye@etu.univ-lyon1.fr, samir.aknine@univ-lyon1.fr, ONN@il.ibm.com,
ngalagne@yahoo.com

Résumé

Nous présentons une nouvelle méthode pour la formation de coalitions stables dans un contexte de tâches à évolution non déterministe et d'agents autonomes égoïstes avec des disponibilités imprévisibles. Notre méthode se fonde sur un mécanisme d'alliances et de recommandations pour faciliter la formation de coalitions stables en dépit de la dynamique des agents et des tâches. Une évaluation de performances expérimentale de notre méthode est également proposée pour dégager les propriétés intrinsèques au modèle.

Mots Clef

Alliance, recommandation, stabilité, coalition.

Abstract

We present a new method for stable coalition formation in a context of dynamic tasks and selfish autonomous agents with unpredictable availability. Our method is based on a mechanism of alliances and recommendations to ease stable coalition formation despite the dynamism of the agents and of the tasks. An experimental performance evaluation of our method is also proposed to highlight the properties of the model.

Keywords

Alliance, recommendation, stability, coalition.

1 Introduction

Pour faire coopérer des agents hétérogènes et compétitifs dans un système multi-agents (SMA), on a le plus souvent recourt aux méthodes de formation de coalitions. Une coalition est formée d'un ensemble d'agents qui se mettent temporairement en groupe pour atteindre les mêmes objectifs [7], [8]. Les méthodes actuelles considèrent que : (1) l'ensemble des coalitions stables qui maximisent l'utilité des agents peut être calculé a priori de manière déterministe, (2) les coalitions sont stables durant l'exécution des tâches, (3) les tâches n'évoluent plus une fois que les coalitions sont formées. Cependant, ces considérations sont des hypothèses fortes pour un certain nombre d'applications réelles, par exemple dans le cas d'un sinistre où des secouristes qui disposent de téléphones ou de PDA hébergeant des agents intelligents doivent s'organiser en groupe de manière décentralisée afin de réaliser des tâches complexes (extinction d'incendies, évacuation de survivants, ...). Il est clair que dans ce contexte un agent ne peut pas connaître a priori la distribution des compétences, les préférences, la disponibilité des autres agents et ne peut communiquer qu'avec son voisinage direct. À ces contraintes s'ajoutent la dynamique des tâches qui peut altérer l'efficacité des actions relatives aux interventions urgentes et l'intérêt des secouristes (éviter les risques et maximiser leurs gains).
Notre contribution est une nouvelle méthode de formation de coalitions stables qui tient compte de l'absence d'informations sur les préférences, sur la disponibilité des agents et de la dynamique des tâches. Autrement dit, nous proposons un mécanisme de négociation fondé sur la construction progressive et dynamique de connaissances et d'affinités entre des agents autonomes et égoïstes afin de faciliter la formation de coalitions stables.
Ce travail est organisé comme suit : la section 2 propose un état de l'art des méthodes de formation de coalitions stables. La section 3 décrit les concepts de notre mécanisme. La section 4 détaille notre principe de formation de coalitions stables. La section 5 propose une évaluation de performances et met en évidence les propriétés essentielles de notre méthode. La section 6 conclut ce travail.

2 Travaux antérieurs

Dans [2], les auteurs proposent un mécanisme incitatif pour la formation de coalitions dans les réseaux MANETs dans un contexte coopératif. Les auteurs modélisent le réseau comme un graphe dont les noeuds sont les agents et les arcs les liens de communication entre les agents. Ils définissent une coalition stable comme étant formée de tous les agents dont les liens peuvent être activés. Contrairement à notre approche, leur mécanisme nécessite la connaissance de tous les liens possibles et les positions des participants potentiels. En outre, il est indispensable de réévaluer le graphe à chaque fois qu'un agent change de voisinage ou qu'il devient indisponible. [9] traite l'inaccessibilité dans les réseaux MANETs. Ils proposent de maintenir et de construire une distribution optimale d'agents spécialisés et dont le rôle est de connecter les agents. Toutefois, cette solution ne peut être efficace que si ces agents spécialisés sont disponibles, car aucune solution n'est proposée pour gérer leur indisponibilité. [4] propose une méthode de formation de coalitions dynamiques (DCF-A) pour permettre aux agents cognitifs de réagir face aux évènements imprévus. Dans le système DCF-A, chaque coalition construite

est représentée par un agent appelé *leader* qui agit comme représentant de la coalition. Le *leader* simule des évènements hypothétiques et ajuste la configuration de la coalition au fur et à mesure des simulations. Lorsque le *leader* de la coalition parvient à une amélioration significative de la valeur de la coalition par ses simulations, il en informe tous les membres de cette coalition sur les alternatives possibles. Cette approche est inadaptée au contexte dynamique en raison de la centralisation des décisions autour du *leader*. De plus, aucun mécanisme pour gérer l'indisponibilité des *leaders* n'est proposé. Pour former des coalitions stables dans [5], les auteurs recherchent le bien-être collectif qui maximise le graphe des structures de coalitions (le *core*). Seules les structures de coalitions qui maximisent le bien-être social sont considérées comme stables. L'objectif de [1] est d'étudier la stabilité des structures de coalitions (la stabilité du *core*) pour les jeux coopératifs dans des environnements structurés. Les auteurs représentent les fonctions caractéristiques des jeux modélisés par un graphe dans le but de déterminer les coalitions possibles. Dans cette méthode, une coalition est un ensemble connexe dans le graphe représentant l'environnement des agents. Pour déterminer les structures de coalitions possibles, ils supposent connaître les besoins, ou au moins les positions des agents. Toutefois, il est prouvé dans [6] que la recherche d'une structure de coalitions stables est un problème difficile. Une autre restriction de ces travaux est qu'ils ne proposent pas de méthodes de formation de coalitions dynamiques car ils obligent la restructuration des coalitions après l'indisponibilité d'un ou de plusieurs agents pour garantir la continuité de l'exécution des tâches. Cependant, la restructuration des coalitions n'est pas toujours optimale ou possible dans un contexte d'environnement dynamique. Dans [3] les auteurs considèrent la dynamique des tâches mais avec des agents homogènes et coopératifs. Ils proposent une méthode de formation de coalitions basée sur les MDP mais sans tenir compte de la disponibilité aléatoire des agents lors de l'exécution des tâches, sans chercher à garantir la stabilité des coalitions formées et ils utilisent des connaissances a priori sur les préférences des agents et sur la dynamique des tâches pour guider le choix des coalitions à former.

3 Concepts du modèle

Nous définissons une tâche comme un ensemble d'actions à exécuter (par exemple un feu à éteindre) tandis qu'un but est un objectif qu'un agent veut atteindre (par exemple une récompense). Pour l'exécution d'une tâche, les agents doivent former une coalition. Une coalition C est définie par C=$\{A_c, R_{min}, T_c\}$, où A_c=$\{a_1, a_2, ..., a_k\}$: $A_c \subseteq A$ est l'ensemble des agents de C, R_{min}=$\{R_1, R_2, ..., R_m\}$ est un ensemble de ressources nécessaires à C, T_c est la tâche qui a induit la formation de C. Notre approche considère que toute tâche est dynamique. Une tâche dynamique est une tâche à évolution non prévisible qui peut s'amplifier, régresser ou être stationnaire.

La tâche d'une coalition C se formalise comme un tuple T_c=$\{\Theta_{T_c}, S_{T_c}, \varphi_{T_c}, \Delta_{T_c}\}$, où Θ_{T_c}=$\{t_1, t_2, ..., t_n\}$ comprend une ou plusieurs sous-tâches t_i et où S_{T_c}=$\{S_{t1}, S_{t2}, ...S_{tn}\}$ identifie l'ensemble des états des sous-tâches. φ_{T_c}=(α, β) est la localisation de T_c dans l'environnement c'est-à-dire latitude α et longitude β. Les contraintes associées à l'exécution de T_c sont exprimées par Δ_{T_c}=$(D_c, Cons_c, val_c)$ où D_c est le temps d'exécution stochastique de T_c par C, $Cons_c$ les contraintes concernant les ressources acceptables requises pour T_c (e.g. autonomie énergétique minimale, fiabilité minimale, etc.) et val_c la récompense associée à la réalisation de T_c.
Le temps d'exécution stochastique est une durée non prévisible a priori pour la réalisation d'une tâche et qui dépend de l'évolution aléatoire des tâches. Pour estimer ce temps D_c (cf. équation 1), un agent qui identifie une tâche à exécuter commence par estimer le nombre moyen $\bar{N}$ d'agents requis pour la tâche en supposant que chaque agent qui accepte de participer à la coalition y contribue avec au moins une ressource. D_c est donné par :

$$D_c = \frac{\bar{N}}{\sum_{i=1}^{\bar{N}} \bar{X_{a_i}}} \quad (1)$$

où $\bar{X_{a_i}}$ est la contribution marginale de a_i qui mesure son niveau de contribution dans la réalisation de la tâche. On considère que tout agent qui initie une demande de formation de coalition ne possède aucune connaissance sur la contribution marginale des agents avec lesquels il doit coopérer sauf s'ils ont précédemment eu à coopérer. Dans l'ignorance, il estime que les agents ont au moins le même niveau de contribution que lui. La récompense de a_i qui participe à la coalition C est calculée comme suit :

$$reward_{a_i} = val_c * \frac{\delta t_{a_i}}{D_c} \quad (2)$$

où val_c est la récompense associée à la réalisation de T_c, δt_{a_i} la période durant laquelle a_i est resté dans C. Ainsi, $reward_{a_i}$ est maximale si δt_{a_i}=D_c. Cette formulation de la récompense a pour objectif de motiver les agents à rester dans leur coalition et à minimiser les désengagements volontaires car un agent peut quitter sa coalition à tout instant, même s'il est initiateur de celle-ci. Pour tenir compte de la disponibilité aléatoire des agents, nous définissons deux classes d'agents pour une coalition : les agents appelés *primaires* et les agents appelés *subordonnés*.
$a_i \in A_c$ est dit *primaire* s'il est engagé dans une coalition C et a choisi un agent suppléant $a_{alter} \in A \setminus A_c$ appelé *subordonné* qui s'engage à l'assister au cas où il serait en manque de ressources pour se maintenir dans C ou s'il doit rechercher de nouvelles ressources pour stabiliser C. L'ensemble des *primaires* d'une coalition C est appelé $backbone$. Notons qu'un agent ne peut appartenir qu'au *backbone* d'une seule coalition. Cette contrainte permet d'éviter que l'indisponibilité d'un agent n'entraîne l'instabilité de plusieurs coalitions. Ainsi, un agent bien qu'il ait

été à l'origine de la coalition, une fois que celle-ci est établie, il a la même responsabilité dans la coalition que les autres agents du backbone car le contrôle de la coalition devient décentralisé. Donc, même l'agent initiateur doit choisir un agent subordonné qui peut le remplacer en fonction des contraintes de la coalition. Si un agent du backbone n'a pas de subordonné, les autres agents primaires (backbone) de sa coalition vont prendre en charge son indisponibilité. Nous reviendrons plus en détails sur la relation entre *primaire* et *subordonné* dans la section 4.

Les contraintes de $a_i \in A$, où A=$\{a_1, a_2, ..., a_n\}$ est l'ensemble des agents, sont données par les paramètres $\{R_{a_i}, Aut_{a_i}, Hs_{a_i}, \vartheta(a_i), B_{a_i}\}$ où R_{a_i} est l'ensemble de sa(ses) ressource(s), Aut_{a_i} son autonomie énergétique, Hs_{a_i} son *historique* des interactions qui contient l'ensemble des $alliances$, $\vartheta(a_i)$ sa vue et B_{a_i} son but. Pour atteindre ses buts, a_i doit déterminer dynamiquement les coalitions à former avec les agents de son voisinage que nous appelons ici *vue*. Le voisinage d'un agent dépend de la force du signal du dispositif qui héberge l'agent. Ainsi, la vue d'un agent peut avoir 0, 1, ... n agents voisins.

La *vue* $\vartheta(a_i)$ de a_i qui a comme voisins a_j et a_k, est exprimée par $\vartheta(a_i)$=$(\{a_j, x_{i,j}\}, \{a_k, x_{i,k}\})$. Les paramètres $x_{i,j}$ et $x_{i,k}$ sont des paramètres booléens qui spécifient respectivement s'il existe ou non une *alliance* entre a_i et ses voisins a_j et a_k. Si $x_{i,j}$=$vrai$ cela signifie qu'il existe une *alliance* entre a_i et a_j sinon, si $x_{i,j}$=$faux$ cela signifie qu'il n'existe pas d'*alliance* entre a_i et a_j. Une *alliance* entre a_i et a_j se traduit par la relation Al_{a_i,a_j}=$(\{R_{a_i}, R_{a_j}\}, \{T^{a_i}_{help}, T^{a_j}_{help}\})$. Cette relation signifie que a_i met à la disposition de a_j sa(ses) ressource(s) R_{a_i} durant un temps $T^{a_i}_{help}$ lorsqu'il est sollicité par a_j et que a_j applique le principe de réciprocité à l'égard de a_i en faisant de même avec sa(ses) ressource(s) R_{a_j} durant un temps $T^{a_j}_{help}$. Maintenir des *alliances* permet aux agents de construire dynamiquement leurs connaissances sur les autres agents et facilite l'acquisition d'informations et de ressources. L'ensemble des *alliances* valides de a_i définit son *historique* Hs_{a_i}=$\cup_{j=1}^{|A'|} Al_{a_i,a_j} : A' \subseteq A$. Nous entendons par *alliance* valide une *alliance* maintenue dans le temps du fait que l'agent allié est considéré comme fiable. La fiabilité d'un allié a_j est calculée par a_i en utilisant la loi de *Poisson* [11] qui est la loi de probabilité a priori si nous suivons le nombre d'événements aléatoires (nombre de fois que l'*alliance* n'a pas été respectée) dans un intervalle de temps. Ainsi, la fiabilité de a_j est exprimée par : ρ_{a_j}=$e^{(-\lambda_{a_j})} * (\frac{\lambda^k_{a_j}}{k!})$: λ_{a_j} les événements aléatoires. Comme les agents sont autonomes et égoïstes, alors a_j n'informe pas nécessairement a_i de ses propres intentions. De ce fait, a_i suppose en calculant la valeur de la fiabilité, que a_j respectera toujours cette *alliance* (probabilité d'avoir zéro non respect de l'*alliance*). Alors, dans l'expression de la fiabilité, a_i met la valeur de k à 0. Il obtient ainsi : ρ_{a_j}=$e^{(-\lambda_{a_j})} * (\frac{\lambda^0_{a_j}}{0!})$, c'est-à-dire :

$$\rho_{a_j} = e^{(-\lambda_{a_j})} \quad (3)$$

où λ_{a_j} est le taux de désengagement (nombre de fois que l'*alliance* n'a pas été respectée) de a_j de *alliance* avec a_i. Nous supposons que si a_i relève dix désengagements consécutifs de a_j, alors a_i supprime les *alliances* qu'il a avec a_j donc $e^{-1} < \rho_{a_j} < 1$. Cependant, a_i n'a pas besoin de mémoriser la valeur λ_{a_j} de chaque allié a_j. Par exemple, si pour a_i, ρ_{a_j}=0.74, alors $e^{(-\lambda_{a_j})}$=0.74 $\Rightarrow ln(e^{(-\lambda_{a_j})})$=$ln(0.74) \Rightarrow -\lambda_{a_j}$=$ln(0.74)$ $\Rightarrow \lambda_{a_j}$=$-ln(0.74) \Rightarrow \lambda_{a_j}$=$\frac{3}{10}$. Cela signifie que a_i avait relevé 3 désengagements de a_j. Si suite à cela a_j ne respecte pas leur *alliance*, alors a_i met à jour ρ_{a_j} : ρ_{a_j}=$e^{-(\frac{3+1}{10})}$=0.67 sinon, si a_j accepte et coopère avec a_i alors la mise à jour de ρ_{a_j} donne : ρ_{a_j}=$e^{-(\frac{3-1}{10})}$=0.81. Autrement dit, le calcul directe de la mise à jour de ρ_{a_j} en ρ'_{a_j} est ρ'_{a_j}=$e^{\frac{-[10*(-ln(\rho_{a_j}))+1]}{10}}$ si a_j ne respecte pas leur *alliance* et ρ'_{a_j}=$e^{\frac{-[10*(-ln(\rho_{a_j}))-1]}{10}}$ dans le cas contraire. Ainsi, un a_j doit respecter ses *alliances* s'il veut être accepté dans une coalition et atteindre ses buts.

La proposition 1 justifie l'utilisation de la loi de Poisson.

Proposition 1. Le calcul de la fiabilité est cohérent même si de nouveaux événements aléatoires (pannes, hors de portée, ...) impactent la fiabilité des agents puisque la somme de tous ces évènements aléatoires engendre un processus de Poisson de paramètre λ=$\Sigma_{i=1}^{\infty}\lambda_i$ où $\lambda_i : i \in [1, \infty[$ est le paramètre d'un évènement aléatoire.

Preuve. Soit $Proc_1$ un processus de Poisson dont la probabilité que sa variable X prenne la valeur k est : $P[X$=$k]$=$e^{-\lambda_1} * \frac{\lambda_1^k}{k!}$. Soit $Proc_2$ un processus de Poisson dont la probabilité que sa variable y prenne la valeur k s'écrit : $P[Y$=$k]$=$e^{-\lambda_2} * \frac{\lambda_2^k}{k!}$. Si le processus $Proc$ résulte de la somme des deux processus $Proc_1$ et $Proc_2$, alors la probabilité que sa variable Z prenne la valeur k peut être exprimée par $P[Z$=$k]$=$P[X$+Y=$k]$.

$P[Z$=$k]$=$\Sigma_{i=0}^{k} P[X$=$i] * P[Y$=$k-i]$.

$P[Z$=$k]$=$\Sigma_{i=0}^{k}(e^{-\lambda_1} * \frac{\lambda_1^i}{i!}) * (e^{-\lambda_2} * \frac{\lambda_2^{(k-i)}}{(k-i)!})$.

$P[Z$=$k]$=$e^{-(\lambda_1+\lambda_2)}\Sigma_{i=0}^{k}\frac{\lambda_1^i}{i!} * \frac{\lambda_2^{(k-i)}}{(k-i)!}$.

$P[Z$=$k]$=$e^{-(\lambda_1+\lambda_2)}\Sigma_{i=0}^{k}\frac{\lambda_1^i}{i!} * \frac{\lambda_2^{(k-i)}}{(k-i)!} * \frac{k!}{k!}$.

$P[Z$=$k]$=$\frac{e^{-(\lambda_1+\lambda_2)}}{k!}\Sigma_{i=0}^{k}(\lambda_1^i * \lambda_2^{(k-i)})\frac{k!}{(i!)*(k-i)!}$.

Nous savons que $\frac{k!}{(i!)*(k-i)!}$=C_k^i, alors :

$P[Z$=$k]$=$\frac{e^{-(\lambda_1+\lambda_2)}}{k!}\Sigma_{i=0}^{k} C_k^i \lambda_1^i * \lambda_2^{(k-i)}$. D'après le théorème du Binôme de Newton : $(a$+$b)^n$=$\Sigma_{j=0}^{n} C_n^j * a^j * b^{(n-j)}$. Alors, $P[Z$=$k]$=$e^{-(\lambda_1+\lambda_2)} * \frac{(\lambda_1+\lambda_2)^k}{k!}$. Ainsi, $Proc$ est aussi un processus de Poisson de paramètre λ_1+λ_2. Ce qui prouve notre proposition. ■

Notons qu'une *alliance* peut être *équitable*, *préférée* et *non-dominée*.

Une *alliance* Al_{a_i,a_j} est dite *équitable* si elle est réalisable, si elle n'est pas contraignante pour les deux alliés a_i et a_j, et si elle améliore la capacité des deux agents.

Si l'*alliance* Al_{a_i,a_j} est réalisable alors :

- a_i et $a_j \in A$, R_{a_i} et $R_{a_j} \in R_{min}$.

- $R_{a_i} \neq NULL$ et $R_{a_j} \neq NULL$.
Si l'*alliance* Al_{a_i,a_j} n'est pas une contrainte alors :
- $T^{a_i}_{help} \leq D_c \leq Aut_{a_i}$ $T^{a_j}_{help} \leq D_c \leq Aut_{a_j}$
-$R_{a_i} \cap R_{a_j}$=$\emptyset$ et $T^{a_i}_{help} \sim T^{a_j}_{help}$.
Si l'*alliance* Al_{a_i,a_j} améliore la capacité des agents alors :
-$\forall a_i \in A, \forall R_{a_j} \in Al_{a_i,a_j} \subseteq \cup_{k=1}^{|A'|} Al_{a_i,a_k} : A' \subseteq A$ alors $R_{a_i} \neq R_{a_j}$.
-$\forall a_i, a_j$ et $a_k \in A$, $Al_{a_i,a_j} \cap Al_{a_i,a_k}$=$\emptyset$.
Ainsi, comme une *alliance* de a_i dépendent : de $T^{a_j}_{help}$, de R_{a_j} et de ρ_{a_j} de l'allié a_j, alors a_i peut avoir une préférence entre ses alliés. Pour a_i, Al_{a_i,a_j} est *préférée* à Al_{a_i,a_k} si, a_i a plus confiance en la fiabilité de a_j qu'en celle de a_k. Nous notons cette relation de préférence par : $a_j \succ_{Al} a_k$.
Cependant une *alliance* peut être *équitable* et *préférée* ou *équitable* mais non *préférée*. Pour faire la distinction entre ces deux cas, nous définissons le concept d'*alliance non-dominée* comme une *alliance équitable* et *préférée*.
Il est clair qu'un agent va progressivement connaître les agents fiables avec lesquels coopérer. Il va donc chercher à établir des *alliances non-dominées* afin de maximiser sa capacité à former des coalitions stables pour atteindre ses buts malgré le contexte stochastique du problème.
Pour gérer les coûts de communication réseau nous définissons les concepts d'envoi de messages en mode *non-return broadcast* et de durée de validité TTL (Time To Live) des messages. Le mode *non-return broadcast* signifie que si un message (demande de coalitions, information sur une tâche, etc.) est émis par un agent a_i, ses agents voisins ne doivent pas lui retourner la même information. Tandis que la durée *TTL* signifie qu'un message sera relayé par les agents jusqu'à ce que TTL=0 car à chaque fois qu'un agent doit transmettre un message il décrémente de 1 la valeur de TTL du message reçu.

$$TTL = |\frac{Sz}{2 * \Upsilon}| \quad (4)$$

où Sz est la taille de la zone que couvre la tâche à exécuter, Υ est la portée du signal de l'équipement où est déployé l'agent qui a envoyé le message de départ (e.g. Bluetooth 100 mWatt, $\Upsilon \leq$100 mètres). TTL=1 pour tout message de partage d'informations entre les agents *primaires* d'une même coalition C. De ce fait, un message qui ne concerne que C ne sera reçu que par les agents de C.
Dans la section suivante, nous décrivons notre méthode de formation de coalitions stables qui tient compte de la disponibilité aléatoire des agents et de la dynamique des tâches. Par coalition stable, nous entendons une stabilité du point de vue de l'équilibre de Nash [10].

4 Mécanisme de formation de coalitions

Le mécanisme avec lequel un agent a_i gère sa demande de formation d'une coalition C pour une tâche T_c est décrit par l'algorithme 1. a_i commence par calculer le temps de réponse εt nécessaire pour que l'agent voisin le plus éloigné ait le temps de répondre à la demande. εt=(2*dt_1)+dt_2 où dt_1 regroupe le temps de modulation du message à transmettre et de démodulation du message reçu tandis que dt_2 est le temps nécessaire pour envoyer et recevoir un message sur le réseau de communication. Après cela, a_i diffuse un message $Declare(T_c, R_{min}, Al^P_{a_i})$ qui contient la tâche T_c, le minimum de ressources R_{min} requis et la proposition d'une *alliance* $Al^P_{a_i}$. La proposition $Al^P_{a_i}$ contient la(les) ressource(s) R_{a_i} et le temps $T^{a_i}_{help}$ durant lequel R_{a_i} va(vont) être mise(s) à la disposition d'un agent qui accepte l'*alliance*.
Si a_j est prêt à rejoindre C et est d'accord avec $Al^P_{a_i}$, il envoie à a_i un message $Help(P^{a_j}_s, t_i, R_{a_j})$ qui contient la sous-tâche $t_i \in T_c$ qu'il souhaite gérer, sa *probabilité de stabilité* $P^{a_j}_s$ (cf. équation 5) et sa(ses) ressource(s) R_{a_j}. Si a_i valide la participation de a_j avec un message $Def\_Ack(a_j)$ alors Al_{a_i,a_j} est établie et chaque agent met à jour son *historique*. L'*historique* de a_i répertorie l'ensemble des alliés de a_i où chaque allié a_j est identifié et relié avec sa fiabilité ρ_{a_j}, sa(ses) ressource(s) R_{a_j} et sa probabilité de stabilité $P^{a_j}_s$ qui s'exprime par :

$$P^{a_j}_s = 1 - Q^{a_j}_s \quad (5)$$

où $Q^{a_j}_s$ est la probabilité d'indisponibilité de a_j. Pour déterminer $Q^{a_j}_s$, a_j utilise la *loi géométrique modifiée* [11] qui est la loi de probabilité a priori quand on s'intéresse à l'arrivée de sa première indisponibilité.
$Q^{a_j}_s$=$(q^{a_j}_s)^k$*$(1 - q^{a_j}_s)$, où $q^{a_j}_s$=$e^{-\lambda_{a_j}} * \frac{(\lambda_{a_j})^k}{k!}$.
λ_{a_j} est le taux d'indisponibilité de a_j, k est le nombre de fois où l'agent à été indisponible. Ainsi pour a_i, si $P^{a_k}_s > P^{a_j}_s$ alors a_i va préférer a_k au lieu de a_j.
Si a_j demande à a_i de modifier la(les) ressource(s) R_{a_i} et/ou le temps $T^{a_i}_{help}$ proposé(s) alors a_i vérifie si cette modification engendrerait une *alliance équitable*. Si l'*alliance* est *équitable* et que a_i l'accepte alors a_i valide la participation de a_j et les deux mettent à jour leur *historique*. Dans le cas où a_i valide la participation de a_j, il supprime de la liste R_{min} préétablie pour la coalition, la(les) ressource(s) R_{a_j}. A l'issue de cette étape, a_i et a_j synchronisent leurs connaissances à propos de la tâche et se coordonnent avec les autres membres de la coalition s'il en existe. Tout a_j accepté pour satisfaire R_{min} devient agent *primaire*, responsable de sa sous-tache t_j et doit chercher un *subordonné* ($Find\_substitute(\Delta_{t_i})$). Cette distribution de la gestion des sous-tâches t_i de T_c permet d'éviter une centralisation du contrôle de la coalition.
Si les ressources minimales R_{min} ne sont pas trouvées, alors chaque a_i du *backbone* de C utilise son Hs_{a_i} pour trouver des alliés a_j de fiabilité ρ_{a_j} maximale qui ont les ressources recherchées. Pour chaque ressource, s'il y a plusieurs alliés a_j qui peuvent la fournir, a_i les trient suivant $P^{a_j}_s$ avant de les contacter ($Recom(T_c, R_{min}, Al^P_{a_i})$) un à un par ordre décroissant de $P^{a_j}_s$. Ce message $Recom(T_c, R_{min}, Al^P_{a_i})$ est une demande de *recommandation* qui signifie qu'un allié a_j de a_i doit fournir $R_{a_j} \in R_{min}$ conformément à l'*alliance* sous peine de voir a_i réduire ρ_{a_j} (cf. équation 3).

On a deux cas pour chaque demande de *recommandation*.
(1) a_j a une *alliance* avec a_i : Alors, si a_j peut satisfaire a_i, il répond avec $SendRecom(a_j, Al_{a_i,a_j})$ et attend la réponse $AckSendRecom(a_j, Al_{a_i,a_j})$ de a_i avant de rejoindre la coalition de a_i ($AckRecom(a_j)$). Si a_j ne peut pas satisfaire a_i, il utilise son Hs_{a_j} pour trouver un allié qui peut servir a_i. Alors il répond à a_i en le mettant en rapport avec son allié. Sinon, si a_j ne trouve pas d'allié il ignore la demande de *recommandation* tout en sachant que sa fiabilité sera réduite par a_i.
(2) a_j n'a pas d'*alliance* avec a_i : Alors, a_j envoie à a_i une demande d'établissement d'*alliance* $SendNewAl(Al_{a_j,a_i})$ avant d'attendre sa réponse d'acceptation $AckSendNewAl(Al_{a_j,a_i})$. Si a_i accepte, a_j mémorise a_i dans son Hs_{a_j} et attend la validation de sa participation dans C par un $Def\_Ack(a_j)$. Dans le cas contraire, a_j ignore la demande de *recommandation* de a_i.

Algorithme 1 : Negotiation

Require: T_c
1: dt_1=($2*\frac{Taille\_trame}{Debit\_reseau}$), dt_2=($2*\frac{(2*\Upsilon)*Taille\_trame}{Debit\_reseau}$).
2: Calculer εt=($2*dt_1$)+dt_2 où Υ est le rayon du voisinage.
3: Déterminer S_z pour calculer le TTL et déterminer R_{min}.
4: Calculer D_c, initialiser $T^{a_i}_{help}$=D_c et $Al^P_{a_i}$=$(R_{a_i}, T^{a_i}_{help})$
5: Déterminer les contraintes Δ_{T_c}=$(D_c, Cons_c, val_c)$
6: $Declare(T_c, R_{min}, Al^P_{a_i})$ où T_c=$\{\Theta_{T_c}, S_{T_c}, \varphi_{T_c}, \Delta_{T_c}\}$
7: **repeat**
8: **if** $(Help(P^{a_j}_s, t_i, R_{a_j}))$ et $t_i \subseteq T_c$ et $R_{a_j} \subseteq R_{min}$ **then**
9: $supprimer$ R_{a_j} de R_{min}
10: $Def\_Ack(a_j)$ et enregistrer a_j dans son Hs_{a_i}
11: **else**
12: $T^{a_i}_{help}$=$(T^{a_i}_{help} - \varepsilon t)$ et $Al^P_{a_i}$=$(R_{a_i}, T^{a_i}_{help})$
13: $Declare(T_c, R_{min}, Al^P_{a_i})$
14: **end if**
15: k++ et attendre les réponses pendant εt
16: **until** (k==3)||(R_{min}==$NULL$)
17: **if** (R_{min}==$NULL$) **then**
18: $Find\_substitute(\Delta_{t_i})$ où $\Delta_{t_i} \subseteq \Delta_{T_c}$
19: **else**
20: $Recom(T_c, R_{min}, Al^P_{a_i})$
21: **if** $(SendRecom(a_j, Al_{a_i,a_j}))$ **then**
22: $AckSendRecom(a_j, Al_{a_i,a_j})$
23: **if** $AckRecom(a_j)$ et $R_{a_j} \in R_{min}$ **then**
24: $supprimer$ R_{a_j} de R_{min}
25: $Def\_Ack(a_j)$ et enregistrer a_j dans son Hs_{a_i}
26: **end if**
27: **end if**
28: **if** $SendNewAl(Al_{a_j,a_i})$ et Al_{a_j,a_i} est *équitable* **then**
29: $AckSendNewAl(Al_{a_j,a_i})$
30: $Def\_Ack(a_j)$ et enregistrer a_j dans son Hs_{a_i}
31: **end if**
32: **end if**

Proposition 2. La transitivité des *alliances* qui améliore les moyens de formation et de stabilisation des coalitions, est vérifiée grâce au principe de *recommandation*.
preuve. Si $a_i \succ_{Al} a_j$ et $a_j \succ_{Al} a_k$ existent alors si a_i a besoin des ressources de a_k alors a_j va recommander a_k avec une fiabilité $\rho_{a_k} \geq 0.5$ à cause de Al_{a_i,a_j} et du fait que a_j, étant égoïste, cherchera à éviter que sa fiabilité soit décrémentée par a_i. Ainsi, $a_i \succ_{Al} a_k$ est vérifiée. Ce qui prouve notre proposition.■
La recherche de *subordonné* par $a_i \in C$ commence par l'envoi ($Find\_substitute(\Delta_{t_i})$) des contraintes Δ_{t_i} de la sous-tâche $t_i \in T_c$ qu'il gère. Chaque $a_j \in A \setminus A_c$ intéressé, envoie sa probabilité de stabilité $P^{a_j}_s$ à a_i. Après une attente de durée égale à εt, a_i sélectionne la meilleure $P^{a_j}_s$ reçue avant de le notifier à l'intéressé. Si a_j choisi accepte, alors a_i l'enregistre comme son *subordonné* sinon a_i notifie l'agent qui a la meilleure probabilité de stabilité qui suit. Si a_i trouve un *subordonné* a_j, il partage cette information avec les autres *primaires* de sa coalition C qui pourront contacter a_j si a_i quitte C sans avoir eu le temps d'en notifier a_j. Si a_i n'a pas trouvé de *subordonné*, il fait une demande de *recommandation* aux agents de son Hs_{a_i} avant d'attendre les propositions.
Proposition 3. Le principe de *recommandation* permet à un agent de corriger un jugement erroné sur sa fiabilité.
preuve. Considérons trois agents a_i, a_j, a_k et deux *alliances* Al_{a_i,a_j} et Al_{a_j,a_k} où a_i et a_j participent dans la coalition C. Supposons que, a_k considère que a_i n'est pas fiable. Si a_j utilise son *alliance* avec a_k pour former ou stabiliser C alors il existera une *alliance* Al_{a_i,a_k}. Si a_i et a_k restent dans C jusqu'à la fin de l'exécution de la tâche, l'*alliance*, Al_{a_i,a_k} sera validée par a_i et a_k. Ainsi, a_k recalculera la fiabilité de a_i et enregistrera a_i dans son Hs_{a_k}. Ceci prouve notre proposition. ■
Proposition 4. Une coalition C est dans un équilibre de Nash, si chaque *primaire* de C a une *alliance non-dominée* avec au moins un agent de la coalition C.
preuve. Si a_i de C a une *alliance non-dominée* avec au moins un a_j de C alors $a_i \succ_{Al} a_j$ existe pour a_i. Dans ce cas, a_i n'aura pas l'intention de quitter C tant que a_j est dans C de peur que ce dernier ne réduise sa fiabilité. De plus, a_i ne voudrait pas perdre la possibilité d'avoir davantage d'*alliances* avec les autres agents de la coalition C en utilisant la transitivité des *alliances* (cf. proposition 2) et la récompense associée à sa participation à la coalition. Nous déduisons que si un *primaire* a_i a une *alliance non-dominée* avec au moins un agent de sa coalition, il n'aura pas l'intention de la quitter. Au contraire, il cherchera à préserver et à améliorer ses relations avec les autres agents de sa coalition. Ainsi, la coalition sera dans un équilibre de Nash. Ce qui prouve notre proposition. ■
Corollaire. La *recommandation* permet à une coalition de converger et de rester dans un équilibre de Nash.■

5 Évaluation de performances

Nous avons développer notre propre simulateur sous Java vu qu'à ce jour, aucun travail du domaine ne tient compte du contexte dynamique des agents et des tâches que nous considérons. Dans nos simulations les agents sont autonomes et égoïstes. Aléatoirement, un agent peut initier ou

non une demande de coalitions, il peut aussi participer ou non à une tâche. La localisation et les ressources requises pour une tâche sont aussi générées de la même manière. Le nombre d'agents dans l'environnement est choisi entre 10 et 100 et le nombre de tâches simultanées dans l'environnement est compris entre 3 et 8 tâches où les propriétés de chaque tâche sont générées aléatoirement. Les simulations sont effectuées dans une machine Intel i7 (4 GHZ) avec 4GB de RAM. La figure 1 montre le pourcentage de coali-

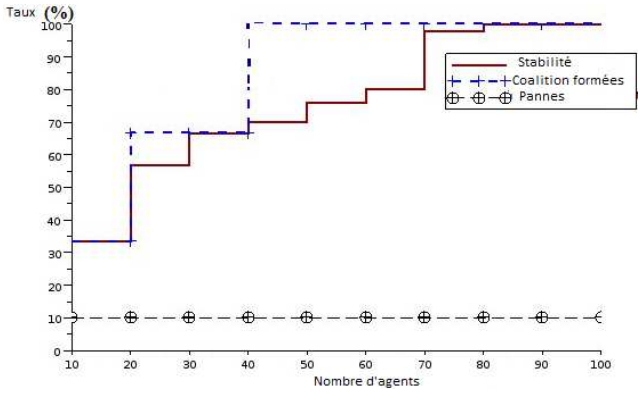

FIGURE 1 – Pourcentage de stabilisation des coalitions.

tions formées et de coalitions stables malgré la dynamique des tâches et la disponibilité aléatoire des agents (pannes) qui suit la distribution de Poisson ($panne \sim P(0.5)$). Nous voyons que tant qu'il y a suffisamment de ressources (40 agents), le taux de stabilisation des coalitions sera toujours supérieur à 70%. Cependant, s'il n'y a pas suffisamment de ressources pour l'ensemble des coalitions (< 20 agents) 50% des coalitions seront néanmoins stabilisées.
Dans la suite, nous présentons le taux (figure 2) et le temps (figure 3) de stabilisation d'une coalition(C1), de deux coalitions(C2), de trois coalitions(C3), etc. en fonction du *Ratio* lorsque la disponibilité aléatoire (panne) des agents suit la distribution Normal $panne \sim N(0.5, (0.1)^2)$. Le *Ratio*

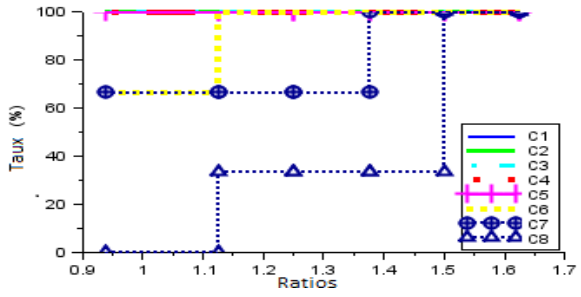

FIGURE 2 – Niveau de stabilité garantie suivant les ratios.

est égal au nombre de ressources dont les alliés de l'initiateur de la coalition détiennent sur le nombre total de ressources nécessaires pour former la coalition. Exemple si a_i forme une coalition nécessitant 15 agents dont 10 sont ses alliés, alors le *ratio* est $\frac{10}{15}$=0.66.

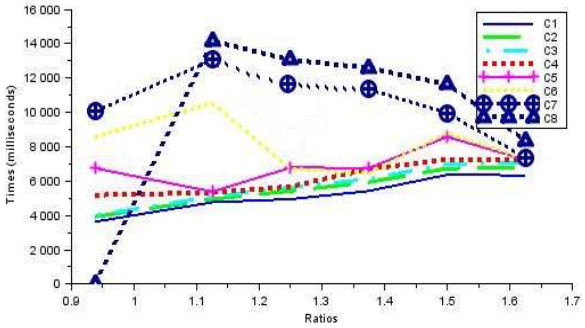

FIGURE 3 – Temps moyen par niveau de stabilité garantie.

Dans la simulation suivante (figure 4) où nous avons 20 agents de disponibilité aléatoire $panne \sim N(0.5, (0.1)^2)$, nous lançons 6 tâches de manière séquentielle. Nous comparons le temps nécessaire pour former une coalition avec ceux qui commencent par calculer l'ensemble des coalitions stables (noté CS dans la figure 4) comme dans [1] avant de commencer leurs négociations et leurs formations. La figure 4 montre qu'après la deuxième tâche notre méthode donne des temps de formation de coalitions plus petit ; ceci grâce au principe de *recommandation* basé sur les *alliances* qui raccourcissent les temps de recherche de ressources.

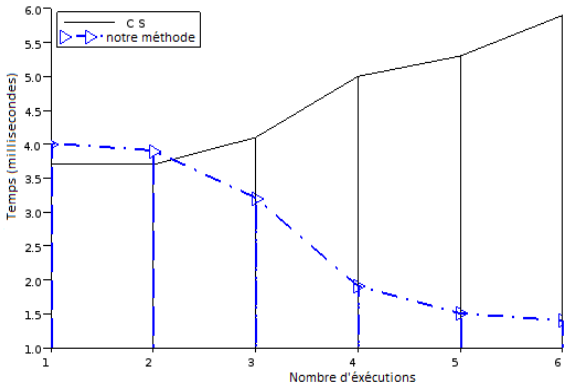

FIGURE 4 – Temps nécessaire pour former les coalitions.

6 Conclusion

Dans ce travail, nous avons proposé une nouvelle approche de formation de coalitions stables avec des agents autonomes et égoïstes de disponibilité imprévisible dans un contexte de tâches dynamiques. Les résultats expérimentaux montrent l'efficacité de notre méthode qui se fonde sur un mécanisme de négociation soutenu par les concepts d'*alliances* et de *recommandation*. Dans nos travaux futurs, nous avons l'intention d'étudier les problèmes de la redistribution des ressources et de la coordination inter-coalitions pour augmenter l'efficacité des agents et le nombre de tâches réalisées.

Références

[1] G. Chalkiadakis, E. Markakis, and N. R. Jennings. Coalitional stability in structured environments. *international joint conference on Autonomous Agents and Multi-Agent Systems*, pages 779–786, 2012.

[2] T. Jiang, G. Theodorakopoulos, and J. S. Baras. Coalition formation in manets. *Army Science Conference, Orlando, FL, USA*, November 2006.

[3] M. A. Khan, D. Turgut, and L. Bölöni. Optimizing coalition formation for tasks with dynamically evolving rewards and nondeterministic action effects. *AAMAS*, 2011.

[4] M. Klusch and A. Gerber. Issues of dynamic coalition formation among rational agents. *Intelligent Systems, IEEE*, June 2002.

[5] T. Sandholm. Distributed rational decision making. *In the textbook Multiagent Systems : A Modern Introduction to Distributed Artificial Intelligence, Weiss, G., ed., MIT Press*, pages 201–258, 1999.

[6] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Anytime coalition structure generation with worst case guarantees. *arXiv preprint cs/9810005*, 10 1998.

[7] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé. Coalitions structure generation with worst case guarantees. *AI Journal*, 1999.

[8] O. Shehory and S. Kraus. Methods for task allocation via agent coalitions formation. *AI Journal*, 1998.

[9] D. Šišlák, M. Pěchouček, M. Rehák, J. Tožička, and P. Benda. Solving inaccessibility in multiagent systems by mobile middle agent. *Multiagent and Grid Systems*, pages 73–87, 2005.

[10] H. Xu, D. Marc Kilgour, K. W. Hipel, and E. A. McBean. Theory and implementation of coalitional analysis in cooperative decision making. *Theory and Decision*, 76 :147–171, February 2014.

[11] R. D. Yates and D. J. Goodman. Probability and stochastic processes : A friendly introduction for electrical and computer engineers. 2005.