Formation de Coalition pour la Maximisation de l'Utilité de la Réponse : Application à la Recherche d'Information

Habiba Belleili
Laboratoire LRI
Université Badji Mokhtar
BP 12, Annaba Algérie
belleili_h@yahoo.fr

Maroua Bouzid
GREYC-CNRS
BD Maréchal Juin,
BP 5186,
14032 Caen Cedex
bouzid@info.unicaen.fr

Mokhtar Sellami
Laboratoire LRI
Université Badji Mokhtar
BP 12, Annaba Algérie
sellami@lri-annaba.net

RÉSUMÉ. Ce papier aborde le problème de la négociation entre des agents munis de ressources limitées et sous contraintes temporelles dans un environnement dynamique. La société d'agents consiste en des agents ayant le même but qui est de répondre dans les meilleurs délais aux requêtes de clients. Les agents sont dotés chacun d'une seule technique locale pour améliorer « progressivement » la qualité de la requête. Les agents doivent entrer en négociation pour former une coalition qui maximise l'utilité de la réponse de la nouvelle requête. Le but est de minimiser le nombre de messages échangés entre les agents pour former une coalition afin de réduire au minimun le temps de négociation.

ABSTRACT. This paper is concerned with the negotiation problem between agents with limited resources and under time constraints in dynamic environment. The society of agents has the same goal which is to respond with best delays at client requests. Each agent has a local technique for improving "progressively" the quality of the request. Agents must begin a negotiation cycle for coalition formation which maximizes the utility of the response to the new request. The goal is to minimize the number of exchanged messages between agents for a coalition formation in order to minimize the negotiation time.

MOTS-CLES : Formation de Coalitions, Négociation coopérative, Raisonnement progressif, système multi-agents, utilité d'une réponse.

KEYWORDS: Coalition Formation, Cooperative Negotiation, Progressive Reasoning, Multi agent systems, Utility of a Response.

1. Introduction

Un moteur de recherche typique est composé de plusieurs modules de recherche d'informations qui exécutent des tâches telles que la formation de la requête, l'optimisation de la requête, l'évaluation de la requête, l'amélioration de la précision, le clustering et la visualisation de la requête. Pour chacune de ces phases, il existe un grand nombre de techniques qui ont été développées [5]. Actuellement, les moteurs de recherche sont construits en intégrant un ensemble fixe de modules et techniques. Le choix est fait en off-line par les concepteurs du système. Cette approche statique exclut les techniques qui donnent un bon résultat dans des situations particulières. De plus, les systèmes actuels de recherche d'informations sont optimisés pour une charge particulière ; ils ne peuvent pas répondre dynamiquement à des charges variables, disponibilité de ressources de calcul, et aux caractéristiques spécifiques d'une requête donnée.
La possibilité d'adapter dynamiquement la profondeur du traitement au temps disponible a été largement étudiée par la communauté de l'intelligence artificielle. Ces efforts ont conduit au développement d'une variété de techniques tels que : les *algorithmes anytimes* [16], *design to time* [3], *le calcul flexible* [4], *le calcul imprécis* [7], et *le raisonnement progressif* [8].
Notre proposition est inspirée des travaux sur le raisonnement progressif qui a pour motivation d'adapter la qualité de la solution au temps disponible. Ce raisonnement est fondé sur une technique multi-niveaux qui transforme progressivement une solution approximative en une solution précise. Si l'exécution d'un niveau compromet les délais impartis d'un problème il est dit que ce niveau est « sauté » selon la terminologie employée dans [8]. Une propriété importante que vérifie le raisonnement progressif est l'amélioration décroissante de la qualité. En effet l'amélioration effectuée au niveau i est plus grande que l'amélioration effectuée au niveau i-1.
Notre approche peut être vue comme une « *agentification* » des niveaux des unités de raisonnement progressif, où chaque agent est muni d'une méthode d'amélioration de la solution. Les agents vont entrer en négociation afin de former une coalition qui maximise l'utilité de la réponse à la nouvelle requête. Plusieurs travaux utilisant une fonction d'utilité pour la prise de décision ont été proposés. En particulier, nous citons [1], [13] pour l'allocation des données dans un environnement multi-agents, [10] pour la coordination de plans et [9] pour la coordination de plans par la collecte de relations temporelles entre eux.
La stratégie globale de l'ensemble d'agents est de fournir les réponses aux requêtes des clients en réduisant au minimum leur temps d'attente (le délai est borné et fixé à l'avance).
La formation de coalition est une importante méthode pour la coopération dans des environnements multi-agents. Plusieurs solutions au problème de formation de coalition ont été présentées [12], [14], chacune de ces solutions est appropriée pour une variété d'environnements. Parmi ces solutions, il y a celles qui sont appropriées pour des agents individuellement rationnels d'autres sont conçues pour des cas de rationalité limitée. Certains models de formation de coalitions sont basés sur les concepts de la théorie des jeux [6] d'autres se sont basés sur la recherche opérationnelle, la théorie des graphes et les aspects algorithmiques des problèmes combinatoires. Dans ce papier, nous présentons un nouvel algorithme distribué de formation de coalition adapté à une architecture d'agents hiérarchiques pour la résolution distribuée de problème à contraintes temporelles. Cet algorithme cherche à minimiser le coût de la coalition tout en maximisant l'utilité de la réponse. Notre approche est inspirée du problème du Sac à Dos qui est un problème d'optimisation NP-complet et pour lequel existent plusieurs algorithmes d'approximation centralisés. Cet article est donc organisé comme suit : dans la section 2, nous présentons un exemple d'application de notre approche. La section 3, aborde l'architecture de notre système. Dans la section 4, nous proposons un scénario pour la formation d'une coalition. Nous présentons dans la section 5 une analyse de l'approche. Enfin, nous terminons dans la section 6 par une conclusion et les perspectives de notre travail.

2. Domaine d'Application

Pour illustrer notre approche nous avons choisi l'application de la fouille de données tirée de [15]. Le système a en entrée une requête composée d'une liste de mots clés. Pour l'amélioration de la requête, les techniques suivantes peuvent être utilisées :
1) Une technique pour parcourir la requête en utilisant des concepts permettant de reconnaître des noms d'entreprise, des noms de personnes, des dates, des lieux etc.
2) Une technique qui examine la requête et permet l'ajout d'autres mots pour son amélioration ;
3) Une technique qui utilise des méthodes d'analyse pour reconnaître quelques phrases dans la requête.
4) Une technique qui formule une nouvelle requête à partir des documents trouvés en réponse à la requête courante. Cette technique, dite analyse contextuelle locale (LCA), est une méthode statistique qui permet d'étendre la requête aux mots qui dépendent du contexte.
5) Une technique qui utilise une bibliothèque (thésaurus) permettant de déterminer les mots les plus proches de la requête afin d'améliorer de manière rapide les performances.

Dans notre approche, nous envisageons un agent par technique. Les agents sont hiérarchiques. La figure 1 illustre l'exemple et la structure hiérarchique des agents.

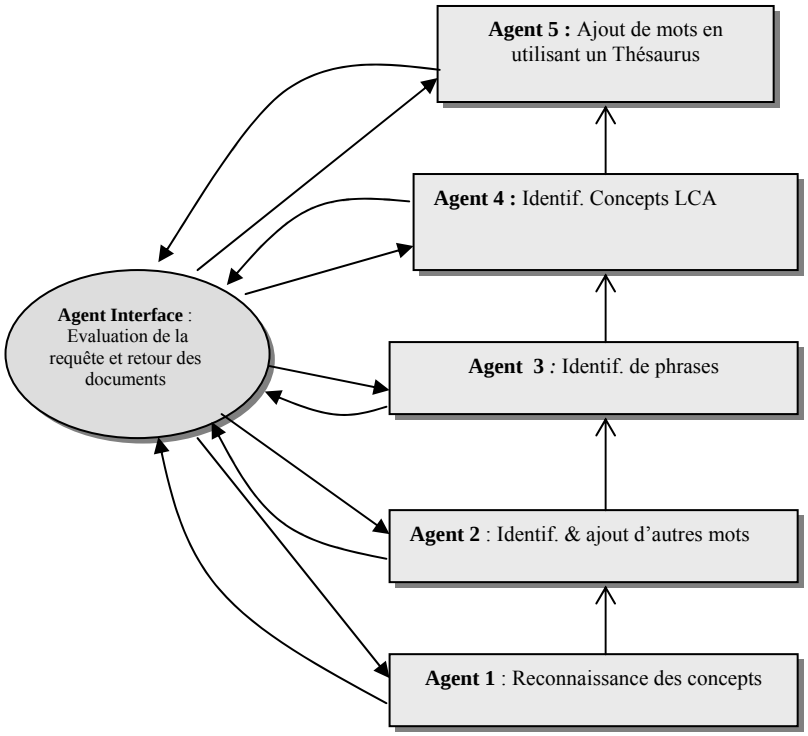


Figure 1 : Agents Hiérarchiques pour l'amélioration de la qualité de la requête

3. Architecture du système

Le système est composé d'un agent *Interface* noté *A* chargé de recevoir les requêtes sous leurs formes originales (de chez le client) et d'exécuter effectivement la requête en fournissant une réponse définitive au client. Une nouvelle requête, avant d'être exécutée par l'agent *A*, va être traitée progressivement par un ensemble d'agents, en vue d'une amélioration progressive de la qualité de la requête, qui permettra une utilité maximale de la réponse. Soit Gamma= $\{\alpha_1,\alpha_2,...,\alpha_n\}$ cet ensemble. Les agents de l'ensemble Gamma sont munis chacun d'une technique d'amélioration de la qualité de la requête. Ces agents sont numérotés de 1 à n. L'ordre de numérotation n'est pas aléatoire mais reflète l'ordre dans lequel sera améliorée progressivement la requête.

3.1 Voisinage d'un agent

Chaque agent α_i ($i\neq1$) de l'ensemble Gamma possède deux voisins directs par défaut qui sont le voisin du niveau inférieur α_{i-1} et le voisin du niveau supérieur α_{i+1}.
Les voisins indirects du niveau inférieur de l'agent α_i sont les α_k ($1\leq k<i$). Les voisins indirects du niveau supérieur de l'agent α_i sont les α_k ($i<k\leq n$).
Ainsi un agent α_i, pour une requête donnée, ne peut avoir en entrée que les résultats fournis par un agent du niveau inférieur et ne peut fournir les résultats de sa technique locale qu'à un agent du niveau supérieur ou à l'agent *A*.

3.2 Accointances d'un agent

Les accointances d'un agent α_i sont formées par la distribution de probabilités de sa technique locale. Celle ci est en fonction de l'agent du niveau inférieur qui aurait fourni une qualité donnée en entrée. En d'autres termes, chaque agent α_i possède le profil des agents voisins du niveau inférieur (Tableau 1). Ces accointances sont construites d'une manière empirique.

α_{i-1}		⋯	α_1	
Prob	**durée**		**prob**	**durée**
0.6	30		0,7	75
0.2	25		0,3	100
0.2	20		-	-

Tableau 1. **Profil des agents voisins à α_i i>1 (niveaux inférieurs)**:

la durée d'exécution de la technique locale de l'agent α_i est de 30 avec une probabilité de 0,6, 25 avec une probabilité de 0.2 et 20s avec une probabilité de 0.2, si les entrées viennent de l'agent α_{i-1}. Elle dure 75 avec la probabilité de 0,7 et 100 avec la probabilité de 0.3 si les entrées viennent de l'agent α_1

3.3 Propriétés

- **Propriété 1 : Incertitude sur la durée d'une méthode**

Une propriété importante de cette architecture est qu'une entrée (résultat de traitements antérieurs des agents des niveaux inférieurs) fournie à un agent de niveau i peut être de qualité plus ou moins bonne. C'est la raison pour laquelle la durée de sa méthode locale n'est pas une constante.

- **Propriété 2 : Traitement séquentiel**

L'agent du niveau i (i≠1) ne procède à sa part de traitement au profit d'une requête donnée que s'il a reçu une entrée correspondant au résultat des améliorations antérieures effectuées par un ou plusieurs agents du niveau inférieur. Cette dernière propriété confère aux agents hiérarchiques une propriété d'interdépendance dans le traitement des requêtes.

- **Propriété 3 : Voisinage d'un agent pour une requête donnée**

Pour chaque agent, le voisin du niveau inférieur et le voisin du niveau supérieur pour le traitement d'une requête donnée sont arrêtés grâce à des cycles de négociation.

4. Formation de coalition

Nous nous situons dans un problème à contraintes temporelles, et pour des raisons de surcharges qui peuvent compromettre l'échéance de la requête certains agents ne peuvent être prévus dans le traitement d'une requête. Notre proposition a pour but de répondre à la question suivante : Quels sont les agents qui doivent se retirer du traitement de la dite requête? Ou en d'autres termes, quels sont les agents qui compromettent l'échéance d'une requête ? Si tous les agents ne peuvent pas être inclus dans le traitement d'une requête, quels sont ceux d'utilité minimale ?

Une recherche exhaustive ne peut être appliquée car il faut examiner les 2^n coalitions possibles ce qui rend la complexité exponentielle, de plus, vue la nature dynamique de l'environnement dans lequel évoluent les agents nous ne pouvons pas construire des heuristiques.

Dans cette section nous présentons la prise de décision orientée utilité (utility driven decision), ensuite nous présentons le principe de notre proposition et nous terminons par l'algorithme de formation de coalition.

4.1 Prise de décision orientée utilité

L'utilité d'un agent pour une requête R_j correspond au temps de réponse d'un agent pour la requête R_j. Plus le temps de réponse d'un agent pour une requête R_j est petit, plus son utilité pour le traitement de la requête R_j est grande. L'utilité d'un agent à une requête est donc une fonction inverse de son temps de réponse à cette même requête.

Le temps de réponse d'un agent à une requête est calculé en fonction du temps d'attente local de la requête R_j et de la durée maximale de sa technique locale .

Le temps d'attente locale d'une requête R_j au niveau d'un agent α_i dépend des temps d'attentes antérieurs de R_j au niveau des agents des niveaux inférieurs. Ces derniers correspondent au temps d'attente cumulé de la requête jusqu'à l'agent α_i .

Exemple : Soit une architecture à trois agents, à l'arrivée d'une requête R_j les agents ont respectivement les charges suivantes : 4 ut, 6 ut, 12 ut

Le temps d'attente locale de R_j au niveau de l'agent 1 est 4 ut, au niveau de l'agent 2 est 6-4=2 ut et au niveau de l'agent 3 est 12-(4+2)=6 ut.

De plus, comme cela a été déjà souligné, la durée de la technique locale d'un agent dépend de l'agent du niveau inférieur qui fournit l'entrée (correspondant au résultat d'une amélioration) concernant la requête R_j .

La fonction d'utilité d'un agent pour une requête R_j n'est pas une fonction indépendante mais elle dépend étroitement des temps de réponse des agents des niveaux inférieurs maintenus dans le traitement de la dite requête et de l'agent du niveau inférieur qui lui fournit l'entrée (pour la durée max de sa technique locale).

L'utilité d'un agent α_i ne pourra donc être calculée sans les informations concernant le temps d'attente cumulé de la requête jusqu'à α_i et l'identité de l'agent du niveau inférieur qui fournirai l'entrée à α_i . Ces informations seront transmises par envoie de messages correspondants.

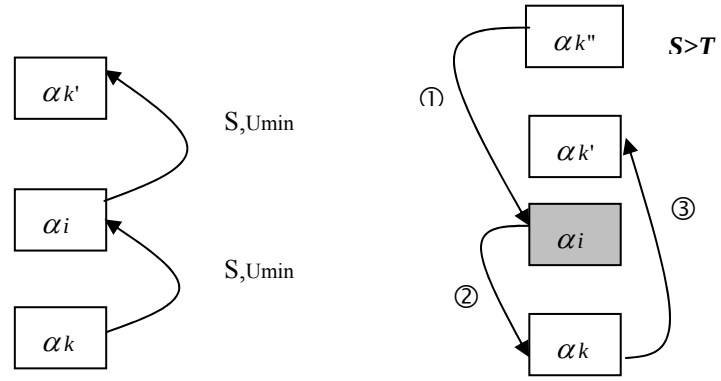
4.2 Principe de l'Approche

A l'arrivée d'une requête, les agents vont entrer en communication pour former une coalition qui maximise l'utilité de la réponse à la nouvelle requête. Cela revient à identifier parmi tout l'ensemble Gamma ceux qui participerons au traitement et à l'amélioration progressive de la requête tout en respectant le temps de réponse maximum à la requête (T).

C'est un problème d'optimisation similaire au problème du sac à dos [11]. Le problème du sac à dos appartient à la classe des problèmes NP-complets et pour lequel il existe plusieurs algorithmes approximatifs. Le problème du sac à dos correspond à une tâche de sélection d'objets qui vont être emportés dans le sac. Le sac à dos a une capacité limitée. Les objets sont choisis parmi un grand nombre. Chaque objet a un volume et une utilité qui lui est propre. Le problème consiste à choisir les objets qui vont être emportés dans le sac de manière à maximiser l'utilité, sans dépasser la capacité du sac.

Nous proposons une approche distribuée à ce problème en l'adaptant à la structure hiérarchique de notre système. La différence avec le problème original est que les objets sont dynamiques (agents) de plus, les poids (durées des méthodes) et les utilités de chacun des agents pour le traitement de la nouvelle requête varient au cours du temps. La durée d'une méthode dépend de l'agent du niveau inférieur qui fournit l'entrée. Le remplissage du sac à dos se fait progressivement à partir de l'agent du niveau 1. Si au niveau d'un agent la capacité du sac S est dépassée, alors l'agent d'utilité minimale va se retirer. Dans ce cas, le remplissage du sac va reprendre à partir du premier voisin du

niveau inférieur de l'agent d'utilité minimale (qui vient de se retirer)(voir Figure 1). Ce processus va être réitéré jusqu'à l'agent du dernier niveau. La figure 2 illustre le principe de l'approche.



Avant dépassement de S :
Remplissage de S et mise à jour de *Umin*

Après dépassement de S au niveau de αk" avec arg(Umin)= αi :

① αi : *Receive*(αk" , αi , *Invitation-de-se-retirer*,*Rj*)

② αi : *Send*(αi , αk , *Nouveau-Voisin*, αk' , *Rj*); // αi se retire

③ αk : *Send*(αk , αk' ,*Invitation-de-participer*, S,..,U,...,*Rj)*; (reprise)

Figure 2 : Remplissage du sac (S) et dépassement de capacité

4.3 L'Algorithme de formation de coalition

Dans cette section nous présentons successivement les notations utilisées dans l'algorithme, la fonction d'utilité qui sera le critère de décision des agents, les primitives de communication et enfin les primitives internes de chaque agent.

4.3.1 Notations

- $T_{\alpha i,Rj}$: le temps nécessaire pour un agent αi afin d'exécuter ses engagements antérieures à l'arrivée de la requête Rj .
- $t_{\alpha i,Rj}$: le temps d'attente locale de la requête Rj au niveau de l'agent αi . $t_{\alpha i,Rj}=\max\left(T_{\alpha i,Rj}-t_{cum,\alpha k}^{Rj},0\right)$ où αk est le voisin du niveau inférieur de l'agent αi pour le traitement de la requête Rj .

  Exemple : si $T_{\alpha 2,Rj}=30$ et $t_{cum,\alpha i}^{Rj}=40$, $t_{\alpha 2,Rj}=\max(30-40,0)=\max(-10,0)=0$.

- $t_{cum,\alpha i}^{R_j}$: le temps d'attente cumulé de la requête R_j jusqu'à l'agent α_i, $t_{cum,\alpha i}^{R_j} = t_{cum,\alpha k}^{R_j} + t_{\alpha i,R_j}$, où α_k est le voisin du niveau inférieur de l'agent α_i pour le traitement de la requête R_j.

4.3.2 Coût d'une coalition

Il est représenté par les ressources de calculs estimées pour répondre à une tâche. Il dépend des coûts cumulés des coalitions antérieures. Il correspond au contenu du sac S.

Le coût d'une coalition est réparti entre ces membres (agents) en temps de réponses de chacun.

4.3.3 Utilité d'un Agent pour une Requête

Elle est calculée en fonction du temps nécessaire au bout duquel un agent fournit une réponse à la requête R_j. Ce temps correspond *au temps d'attente locale* de R_j ajouté à *la durée maximale de la méthode locale* de l'agent. Celle ci varie en fonction de la qualité en entrée fournie par l'agent du niveau inférieur. Elle est notée $g_{\alpha i}(R_j,t)$ où t est le temps de réponse estimé de l'agent α_i à la requête R_j.

4.3.4 Interaction entre les agents

L'interaction entre les agents se fait via un protocole de communication. Celui-ci est basé sur les primitives de passage de message suivantes :

Send/Receive(Sender,Receiver,<Message>)

Il y a cinq types de messages:

- Message « *invitation-de-participer* » : ce type de message est une invitation de l'agent émetteur à l'agent récepteur d'examiner la possibilité de participer au traitement de la nouvelles requête. Les paramètres de ce message sont :
  - *Le contenu du sac S jusqu'à l'agent émetteur,*
  - *Temps d'attente cumulé de la requête R_j jusqu'à l'agent émetteur,*
  - *Valeur de l'utilité minimale obtenue jusqu'à l'agent émetteur,*
  - *Identité de l'agent d'utilité minimale jusqu'à l'agent émetteur*
  - *Identificateur de la requête,*
- Message « *invitation-de-se-retirer* » : ce message est une invitation de se retirer de la négociation, à cause de l'utilité minimale de l'agent receveur. Ce message admet un seul paramètre qui est *l'identificateur de la requête*.
- Message « *Nouveau-Voisin* » : ce type de message est un message de mise à jour qui permettra à l'agent qui le reçoit de connaître et de mettre à jour son nouveau voisin du niveau supérieur après que l'agent émetteur s'est retiré. Les paramètres

de ce message sont : *l'identificateur de la requête* et *l'identificateur du nouveau voisin.*

- Message « *PasDeVoisinSup* » ce type de message est envoyé seulement par l'agent du dernier niveau quand il doit se retirer à cause de son utilité minimale. Le paramètre de ce message est l'identité de la requête R_j . Le récepteur de ce message déduit qu'il n'a plus d'agent du niveau supérieur et qu'il est le dernier agent dans la coalition formée pour la requête R_j (C_{Rj}).
- Message « *fin-de-negociation* » : ce type de message est initié par le dernier agent de la hiérarchie qui se trouve dans la coalition formée pour une requête R_j . Cet agent peut être l'agent du niveau « *n* » s'il ne s'est pas retiré. Le paramètre de ce message est *l'identificateur de la requête*.

4.3.5 Primitives de base

Chaque agent α_i peut être dans l'un des états suivants pour le traitement de la requête Rj :

- Etat 0 : pas de négociation en cours ;
- Etat 1 : négociation en cours;
- Etat 2 : se retire de la négociation ;
- Etat 3 : fin de négociation et l'agent est concerné par le traitement de la requête R_j .

  Il est, en d'autres termes, dans la coalition d'agents qui traitera la requête R_j . Dans cet état l'agent doit connaître son voisin du niveau inférieur et son voisin du niveau supérieur.

  Les primitives suivantes sont utilisées par chacun des agents :

- *State (x)* fonction qui délivre l'état courant de l'agent *x,*
- *NeighbourLow*(*x,r*):fonction qui retourne l'agent voisin à *x* du niveau inférieur pour la requête *r.*
- *NeighbourHigh*(*x,r*):fonction qui retourne l'agent voisin à *x* du niveau supérieur pour la requête *r* s'il existe, sinon elle retourne *x* (l'agent lui même).
- *arg* (*U*) : fonction qui retourne l'identité de l'agent possédant l'utilité *U.*
- *UpdateNeighbourLow*(*x,y*) : *y* devient le nouveau voisin du niveau inférieur de l'agent *x*.
- *UpdateNeighbourHigh(x,y*) : *y* devient le nouveau voisin du niveau supérieur de *x*.
- *Minimum(*U,U') : Cette fonction retourne l'utilité minimale entre les deux utilités U et U'.

4.3.6 Comportement des agents

Nous avons trois comportements d'agents différents, le premier comportement correspond à l'agent du niveau 1, le deuxième correspond aux agents des niveaux intermédiaires et le dernier correspond à l'agent du dernier niveau.

4.3.6.1 Comportement de l'agent du niveau 1 (α_1)

L'agent du niveau 1 (α_1) est l'agent obligatoire, il représente le traitement minimal qui doit être appliqué à la requête. De ce fait il ne peut pas se retirer, une utilité maximale (∞) lui est constamment attribuée. C'est le premier agent qui génère et estime le coût occasionné par la coalition $\{\alpha_1\}$ pour la requête R_j en tenant compte des coûts des coalitions formées pour les requêtes $R_{j'}$ qui sont arrivées avant R_j et qui sont en attente de traitement. Les messages qu'il peut recevoir lors de la négociation pour une requête R_j sont : l'identité de son nouveau voisin du niveau supérieur (message *Nouveau-Voisin*) lors du retrait de son voisin courant du niveau supérieur. L'autre type de message est qu'il n'y a plus de voisin de niveau supérieur (message *PasDeVoisinSup*), à ce moment là la coalition pour la requête R_j se résume en un seul membre $\{\alpha_1\}$, les agents des niveaux supérieurs se sont tous retirés (cas de surcharge du système) ; le dernier type de message est le message de *fin-de-negociation* envoyé par le dernier agent dans la coalition C_{Rj} ; à la réception de ce message, l'agent α_1 envoie un message similaire à son voisin du niveau supérieur et passe à l'état 3. la figure 2 détaille l'algorithme correspondant.

Etat 0:
receive (**'**nouvelle requête'**, A) :**
calcule $T_{\alpha 1,Rj}$ **;** $t_{\alpha 1,Rj} \leftarrow T_{\alpha 1,Rj}$; $t^{Rj}_{cum,\alpha 1} \leftarrow T_{\alpha 1,Rj}$ **;**
$S \leftarrow S + t_{\alpha 1,Rj} + \max\delta_{\alpha 1}$ *; //calcule du coût de la coalition occasionné par* $\alpha 1$
Send **(** $\alpha 1,\alpha 2,$ *invitation-de-participer*, *S*, $t^{Rj}_{cum,\alpha 1}$, ∞ , $\alpha 1, Rj$);
State($\alpha 1$)$\leftarrow$ 1**;**
Etat 1: case event of
receive (αk, $\alpha 1$, *Nouveau-Voisin,* $\alpha k'$ *,Rj):*
UpdateNeighbourHigh($\alpha 1$, $\alpha k'$) *;*
Send ($\alpha 1$, $\alpha_{k'}$ *,invitation–de-participer,* , $t^{R_j}_{cum,\alpha_1}$, ∞ , α_1, R_j); *//reprise*
receive (αk, $\alpha 1$ *,PasDeVoisinSup,Rj) : State(* $\alpha 1$ *)* $\leftarrow$ *3;*
receive (αk, $\alpha 1$ *,fin-de-negociation ,R*j) :
$\alpha k''\leftarrow$ *NeighbourHigh* (αi *,Rj)*;
if $\alpha k'' \neq \alpha k$ *then send* ($\alpha 1$, $\alpha k''$ *,fin-de-negociation,Rj); State(* $\alpha 1$ *)* $\leftarrow$ *3;*
Endcase;

Figure 2: Comportement de l'agent α_1

4.3.6.2 Comportement des agents des niveaux intermédiaires

Un agent de niveau intermédiaire α_i ne peut calculer le coût de la coalition qu'il occasionne et son utilité pour la requête *Rj* que s'il reçoit un message *d'invitation-de-participer*. Ce message avec les informations qu'il véhicule permet à l'agent α_i d'estimer le coût de la coalition qu'il occasionne. Si ce coût compromet le temps de réponse à la requête *Rj* (T) alors l'agent d'utilité minimale est identifié. Si celui-ci correspond à l'agent α_i lui même et s'il possède un voisin du niveau supérieur en négociation pour la requête *Rj*, il informe son voisin du niveau inférieur courant en lui communiquant l'identité de son nouveau voisin du niveau supérieur (via le message *Nouveau-Voisin*) et se retire de la négociation. Si l'agent α_i ne possède pas de voisin du niveau supérieur pour la requête *Rj* (car celui-ci s'est retiré lors des itérations précédentes) il informe son voisin de niveau inférieur courant qu'il n'a plus de voisin de niveau supérieur pour le traitement de la requête *Rj* (message *PasDeVoisinSup*). Si l'identité de l'agent d'utilité minimale n'est pas α_i alors l'agent α_i envoie un message i*nvitation-de-se-retirer* à l'agent d'utilité minimale. La figure 3 présente l'algorithme décrivant le comportement de l'agent du niveau intermédiaire lors de la réception du message i*nvitation-de-participer*.

---

receive (α_k , α_i ,*invitation-de-participer,S,* $t^{R_j}_{cum,\alpha_k}$, $U^{\alpha_k}_{\min,R_j}$, *arg(*$U^{\alpha_k}_{\min,R_j}$*),* R_j *):*

// calcul des parametres locaux à savoir

$t_{\alpha_i,R_j} \leftarrow \max\left(T_{\alpha_i,R_j} - t^{R_j}_{cum,\alpha_k}, 0\right)$; $U_{\alpha_i,R_j} \leftarrow t_{\alpha_i,R_j} + \max\delta_{\alpha_i,\alpha_k}$; $S \leftarrow S + t_{\alpha_i,R_j} + \max\delta_{\alpha_i,\alpha_k}$;

$U^{\alpha_i}_{\min,R_j} \leftarrow$ *minimum(*U_{α_i,R_j} , $U^{\alpha_k}_{\min,R_j}$*);* *// mise à jour de l'utilité minimal*

if (S<T) then // le nouveau contenu de S n'est pas dépassé

$t^{R_j}_{cum,\alpha_i} \leftarrow t^{R_j}_{cum,\alpha_k} + t_{\alpha_i,R_j}$ *; // le temps d'attente cumulé jusqu'à l'agent* α_i

UpdateNeighbourLow(α_i,α_k *);*

send(α_i, α_{i+1} *, invitation-de –participer, S,* $t^{R_j}_{cum,\alpha_i}$, $U^{\alpha_i}_{\min,R_j}$, *arg(*$U^{\alpha_i}_{\min,R_j}$*),* R_j *)*

else // S est dépassé

if arg($U^{\alpha_k}_{\min,R_j}$*)=* α_i *then //* α_i *a la plus petite utilité et va se retirer*

if neighbourHigh(α_i , R_j *)=* α_i *then //* α_i *est le dernier agent de la hiérarchie*

send(α_i, α_k *,PasDeVoisinSup,* R_j *)*

else send (α_i, α_k *,Nouveau-Voisin,* α_{i+1} , R_j *);*

state (α_i *)* $\leftarrow$ *2; //* α_i *s'est retiré;*

else send(α_i, *arg(*$U^{\alpha_i}_{\min,R_j}$ *) ,Invitation-de-se-retirer,* R_j *);*

end if;

end if;

Figure 3: Message i*nvitation-de-participer* pour l'agent du niveau intermédiaire (α_i)

Dans le cas où le coût de la coalition occasionné par l'agent α_i ne dépasse pas l'échéance de la requête *Rj*, l'agent α_i envoie à son voisin du niveau supérieur courant, s'il existe, un message i*nvitation-de-participer* en prenant soin de mettre à jour les paramètres du message pour permettre un comportement cohérent de l'agent receveur. Dans le cas où l'agent α_i ne possède pas de voisin du niveau supérieur en négociation pour la requête *Rj* (à cause de retrait dans des itérations antérieures de la négociation pour la requête *Rj*), il envoie un message de *fin-de-negociation* à l'agent obligatoire α_1.

Un autre type de message qui ne peut pas être reçu par l'agent du niveau 1 est le message i*nvitation-de-se-retirer*, ce message est envoyé par l'agent qui a identifié l'agent d'utilité minimale pour la requête R_j lors du calcul du coût de la coalition qu'il a occasionné. La figure 4 présente le comportement de l'agent α_i lors de la réception du message « *invitation-de-se-retirer* ».

receive (α_k, α_i, *Invitation-de-se-retirer*, R_j):
$\alpha_{k'}$ ← *NeighbourHigh* (α_i , R_j);
α_k"← *NeighbourLow*(α_i , R_j);
send(α_i, $\alpha_{k"}$,*Nouveau-Voisin*, R_j);
state (α_i)← 3; // α_i *se retire de la négociation*

Figure 4 : Message invitation-de-se-retirer

Un agent de niveau intermédiaire peut recevoir le message « *Nouveau-Voisin* » correspondant à la mise à jour du nouveau voisin du niveau supérieur. Cette information est envoyée par l'agent qui doit se retirer. La figure 5 présente l'algorithme du message *Nouveau-Voisin*.

receive (α_k, α_i, *Nouveau-Voisin*, α_k', R_j):
updateNeighbourHigh(α_i , α_k');
send(α_i , α_k' ,*invitation-de-participer*, *S*, $t_{cum,\alpha_i}^{R_j}$, $U_{\min,R_j}^{\alpha i}$, arg($U_{\min,R_j}^{\alpha i}$), R_j);
IIReprise

Figure 5 : Message Nouveau-Voisin

Les messages qui mettent fin à la négociation sont les messages *PasDeVoisinSup* et *fin-de-negociation.* A la réception du message *PasDeVoisinSup*, l'agent en déduit que c'est lui le dernier agent dans la coalition formée pour la requête R_j et initie le message

de *fin-de-négociation*. L'agent qui reçoit le message de *fin-de-negociation* informe son voisin du niveau supérieur, s'il existe, que la négociation pour la requête R_j est terminée et passe à l'état 3. la figure 6 présente l'algorithme des deux messages.

---

receive (α_k, α_i, *PasDeVoisinSup,* R_j):

  send (α_i, α_1 ,*fin-de-negociation*, R_j) ;

receive (α_k, α_i , *fin-de-negociation*, R_j) :

  $\alpha_{k'}$ ← *NeighbourHigh* (α_i , R_j) ;

  if $\alpha_{k'} \neq \alpha_i$ then send (α_i , $\alpha_{k'}$, *fin-de-negociation*, R_j);

 State(α_i) ← *3;*

Figure 6 : Messages PasDeVoisinSup et Fin-de-négociation

4.3.6.3 Comportement de l'agent de niveau n

Au niveau de l'agent du dernier niveau, seul un seul type de message peut être reçu correspondant au message i*nvitation-de-participer* et procèdera de la manière indiquée dans l'algorithme figure 3.

---

5. Analyse

5.1 Terminaison

La terminaison de l'algorithme de formation de coalition est garantie pour deux raisons la première est que le nombre d'agents est limité (Cas de système agent non additif), la deuxième raison est que l'agent qui se retire de la négociation pour une requête *Rj* ne peut plus rejoindre la coalition pour cette même requête.

5.2 Complexité

Notre algorithme présente des retours arrière. Il y a retour arrière quand le coût de la coalition partielle occasionné par un agent (le contenu de S) compromet le temps de réponse à la requête. Le pire des retours arrière est quand l'agent qui possède la plus petite utilité se situe au fond de la hiérarchie. Tous les changements effectués sur le contenu du sac S seront annulés et le remplissage du sac reprend à partir du premier voisin du niveau inférieur de l'agent qui vient de se retirer.

Chaque fois qu'un agent de niveau i est invité à se retirer, les agents des niveaux supérieurs doivent recalculer, chacun à son niveau, le coût de la coalition qu'il occasionne. Pour un agent de niveau i, il procède à la réévaluation du coût de la coalition *(i-1)* fois, au pire cas.

Un agent de niveau *i* procède à la mise à jour de son voisin du niveau supérieur (car son voisin du niveau supérieur courant s'est retiré) *(n-i)* fois au pire cas. La complexité au niveau de chaque agent est donc linéaire *O(n).*

La complexité de l'algorithme de formation de coalition proposé peut être mesurée en nombre de messages envoyés lors de retours arrière. Particulièrement nous distinguons le pire des retours arrière, il correspond à celui où tous les agents se sont retirés sauf l'agent obligatoire (niveau 1) d'une part ,et où le contenu du sac est dépassé au niveau du dernier agent de la hiérarchie et que l'agent qui possède la plus petite utilité se situe au fond de la hiérarchie d'autre part.

Exemple : pour n=5, le remplissage de S (l'évaluation du coût de la coalition occasionné par chaque agent) se fait de l'agent de niveau 1 à l'agent de niveau 5, au niveau duquel il y a dépassement de la capacité du sac, l'agent avec la plus petite utilité se trouve au fond de la hiérarchie c'est à dire le niveau 2.

La complexité de chaque retour arrière est *O(n),* pour *(n-1)* retours arrière au pire cas la complexité est polynomiale $O(n^2)$.

5.3 Résultats Expérimentaux

Nous avons expérimenté notre algorithme en envisageant que le pire cas en faisant varier le nombre d'agents.

Les résultats expérimentaux correspondent au résultat théorique c'est-à-dire la complexité polynomiale. La figure 7 résume le temps CPU en fonction du nombre d'agents au pire cas. C'est à dire (n-1) « pire » retours arrière.

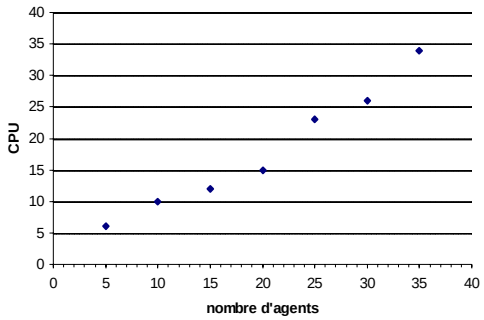


Figure 7 : expérimentation de l'algorithme de formation de coalition pour les pire cas des retours arrière

6. Conclusion

Nous avons proposé une approche agents pour la maximisation de l'utilité de la réponse à des requêtes émanant de clients. Les agents sont coopératifs pour la formation de coalition afin de répondre au mieux aux requêtes (compromis entre qualité de la réponse et ressource temps). L'avantage d'une telle approche, est qu'elle s'adapte à la charge du système. Ainsi, pour des charges différentes, les coalitions suivantes peuvent se former dans l'ensemble gamma : $\alpha_1,\alpha_3,\alpha_4,\alpha_5$ ou $\alpha_1,\alpha_2,\alpha_4$ ou $\alpha_1,\alpha_3,\alpha_4$ … Le traitement effectif des requêtes ne reflète pas l'ordre de leurs arrivées. Ceci représente un autre avantage de cette approche. En effet, une requête dont la qualité ne peut pas être améliorée (à cause d'une surcharge), se voit être exécutée avant une autre requête qui est arrivée avant et qui est en train d'attendre d'être améliorée (pour une qualité minimale, on n'a pas besoin de faire attendre la requête). Ceci est du au fait que l'agent *A*, qui exécute effectivement la requête, est indépendant des agents qui améliorent la requête.

Lors de la formation de la coalition, et si le sac S est dépassé, nous pouvons distinguer deux possibilités. La première possibilité est que l'agent qui a provoqué le dépassement se retire de lui-même [2]. Cette solution, bien que rapide car il n'y a pas de retour arrière (backtracking) mais écarte la solution finale de la solution optimale. En effet, l'agent qui a provoqué le dépassement de S n'est pas forcément celui qui a la plus petite utilité. La seconde possibilité, celle adoptée dans ce papier, est que dès qu'il y a dépassement de capacité, l'agent dont l'utilité est minimale devra se retirer. Bien que cette solution conduise à des retours arrières mais garantie une maximisation de l'utilité de la réponse.

La recherche d'une coalition qui maximise l'utilité de la réponse ne s'est pas faite sur toutes les coalitions possibles. Ceci réduit considérablement la complexité du problème. En effet le nombre de coalitions possibles pour n agents est 2^n ce qui rend le temps pour parcourir toutes les coalitions exponentiel. L'algorithme de formation de coalition que nous avons proposé, et par la nature *monotone* des traitements (les requêtes sont traitées de l'agent du niveau 1 à l'agent du niveau k tour à tour). La formation de la coalition se fait en examinant la charge de chaque agent dans le même ordre de traitement. Dès qu'il y a violation du temps de réponse à la nouvelle requête, l'agent d'utilité minimale est invité à se retirer.

Dans cette approche nous avons supposé qu'un agent du niveau i peut utiliser les résultats de n'importe quel agent du niveau inférieur. De même, il peut communiquer le résultat de sa technique à n'importe quel agent du niveau supérieur. Comme perspectives, nous pensons ajouter une autre contrainte qui est de restreindre les agents du niveau inférieur (resp. supérieur) avec qui l'agent est susceptible d'utiliser (resp. communiquer) le résultat de la technique locale. De cette manière, deux types de coalition doivent être faits. Le premier, dit *sélectif,* se fait entre les agents qui offrent

des techniques similaires (qui s'excluent mutuellement) mais différentes dans les performances et les durées d'exécution. La coalition va sélectionner un agent qui répond au mieux aux contraintes à cet instant. Le second type de coalition est *associatif*. Il est similaire à celui décrit dans ce papier et se forme entre les différents agents sélectionnés dans les coalitions du premier type.

Une autre perspective consiste à munir chaque agent de plusieurs techniques similaires mais avec des performances différentes, les agents vont entrer en négociation pour choisir la combinaison de méthodes (agent, méthode) qui maximise l'utilité de la réponse, tout en minimisant le temps pris par la négociation pour aboutir à un accord.

7. Bibliographie

[1]: Azulay-Schwartz R., Kraus S., 2002. Negotiation on Data Allocation in Multi-Agent Environments. In Autonomous Agents and Multi-agent systems Journal, 5(2): 123-172, 2002.

[2]: Belleili H., Bouzid M., Sellami M., 2004. Anytime Negotiation between Agents with Alternative Methods. In IADIS International Conference on Applied Computing, pages 73-80, April, 2004, Lisbone.

[3] : Garvey A. et Lesser V., 1993. Design-to-time real-time scheduling. IEEE transaction on systems, man, and Cybernetics, 23(6): 1491-1502, 1993.

[4]: Horvitz E. Reasoning under varying and uncertain ressource constraints. Seventh National Conference on Artificial intelligence, 111-116, 1988.

[5] : Jones Karen S. and Willett P. (eds.) Readings in Information Retrieval. Morgan Kaufmann Publishers, 1997.

[6]: Kahan J.P., Rapoport A., 1984, Théories of coalition formation, Hillsdale, LEA, 1984.

[7] : Liu J., Lin K., Shih W., Yu A., Chung J.and Zao W. Algorithms for scheduling imprecise computations. IEEE transactions on computers, 24(5):58-68, 1991.

[8] : Mouaddib AI. and Zilberstein S. Handling duration uncertainly in meta-level control of progressive reasoning. Fifteenth International Joint Conference on Artificial Intelligence, 1201-1206, 1997

[9]: Mouaddib AI., 1998. Multistage negotiation for distributed scheduling of ressource-bounded agents. In AAAI Spring Symposium On Satisfiying Models, pp 54-59, 1998.

[10]: Mouaddib AI., 1999. Anytime coordination for progressive planning agents. in AAAI-99, pp 564-569, 1999

[11]: Sahni S. 1975. Approximated algorithms for 0/1 Knapsack problem. Journal of the ACM, 22 (1975), pp. 115-124. 11

[12]: Sandholm T., Lesser V., 1995. Coalition Formation among bounded rational agents. In IJCAI-95, pp 662-669, Montréal , 1995.

[13]: Schwartz R and Kraus S., 1997. Negotiation on data allocation in multi-agents environments. In AAAI-97, pp 29-35, 1997.

[14]: Shehory O., Kraus S, 1994. Task allocation via coalition formation among autonomous agents IJCAI-95, Montréal, 1995.

[15]: Zilberstein S., Mouaddib AI., 1999. Reactive control for dynamic progressive processing. In IJCAI-99, pp 1269-1273.

[16]: Zilberstein S.and Russel S. Optimal Composition of real time systems. Artificial Intelligence 82(1-2):181-213, 1996.