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 Mokhtar
BP 12, Annaba Algérie 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
BP 12, Annaba
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 requête 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 par 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 at the new request. The goal is to minimize the number of exchanged messages between agents for a formation of a coalition in order to minimize the negotiation time.

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

KEYWORDS: Coalition Formation, Coopérative Négotiation, 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 existe un grand nombre de techniques qui ont été développées récemment [4]. 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* [12], *design to time* [2], *le calcul flexible* [3], *le calcul imprécis* [5], et *le raisonnement progressif* [6]. Le raisonnement progressif a pour motivation d'adapter la qualité de la solution en fonction du temps disponible. Ce raisonnement est fondé sur une technique multi-niveaux qui transforme une solution approximative en une solution précise. 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, où chaque agent est muni d'une méthode. 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], [10] pour l'allocation des données dans un environnement multi-agents, [8] pour la coordination de plans et [7] 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 en minimum leur temps d'attente (le délai est borné et fixé à l'avance.

Notre proposition consiste à établir un scénario de négociation entre les agents pour la formation de coalition en vue de la maximisation de l'utilité d'une réponse à des requêtes. Ce papier est 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 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 [11]. 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. Pour notre approche, nous envisageons un agent par technique.

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 Notations

– T_{α_i,R_j} : le temps nécessaire pour un agent α_i afin d'exécuter ces engagements antérieures à l'arrivée de la requête R_j .

– t_{α_i,R_j} : le temps d'attente locale de la requête R_j au niveau de l'agent α_i , $t_{\alpha_i,R_j}=\max\left(T_{\alpha_i,R_j}-t^{R_j}_{cum,\alpha_k},0\right)$ où α_k est le voisin du niveau inférieur de l'agent α_i pour le traitement de la requête R_j . Exemple si $T_{\alpha_i,R_j}=30$ et, $t_{\alpha_i,R_j}=\max(30-40,0)=\max(-10,0)=0$.

– $t^{R_j}_{cum,\alpha_i}$: le temps d'attente cumulé de la requête R_j jusqu'à l'agent α_i .

$t^{R_j}_{cum,\alpha_i}=t^{R_j}_{cum,\alpha_k}+t_{\alpha_i,R_j}$, où α_k est le voisin du niveau inférieur de l'agent α_i pour le traitement de la requête Rj.

3.4 Utilité d'un Agent pour le traitement d'une requête

Elle est calculée en fonction du temps nécessaire au bout duquel un agent fournirait une réponse à la requête Rj. Ce temps correspond *au temps d'attente locale* de Rj 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 Rj.

3.5 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 :

Receive-low/send-high*(contenu de S, temps d'attente cumulé de la requête Rj jusqu'à l'agent émetteur,Utilité minimale obtenue jusqu'à l'agent émetteur,Identité de l'agent d'utilité minimale,Agent émetteur,Agent destination,Requête)*

Receive-high/send-low*('**invitation de se retirer**',Agent émetteur,Agent destination,Requête concernée) ;*

Receive-high/send-low*('**NewNeighbour**',Agent émetteur,Agent destination,Nouveau voisin,Requête concernée) ;*

3.6 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 nouvelle requête ;
– Etat 1 : en négociation ;
– Etat 2 : n'est plus concerné ;
– Etat 3 : concerné. Dans cet état l'agent doit connaître l'agent du niveau inférieur et l'agent du niveau supérieur.

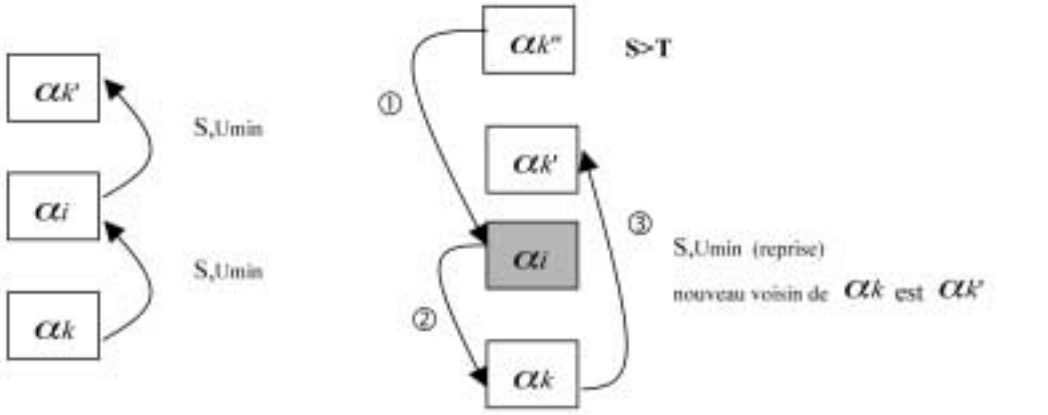
Chaque agent utilise les primitives suivantes :

– ***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***
– **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. Formation de coalition

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 qui peut être approché par le problème du sac à dos [9]. Le problème du sac à dos est un problème qui 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 et 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 aurait fourni 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 de S est dépassée, alors l'agent dont l'utilité est 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. Ci-dessus l'algorithme :



Avant dépassement de S :
Remplissage de S et mise de Umin

Après dépassement de S au niveau de $\alpha k''$ avec arg(Umin)=
①Receive_high('invitation de se retirer' …)
②Send-low('newNeighbour', .., $\alpha k'$,..); αi se retire
③Send-high(S,..,U,..... $\alpha k'$); (reprise)

Figure 1 : reprise du remplissage de S après retrait de l'agent d'utilité minimale

1. Au niveau de l'agent αn

Etat 0: ***receive*** ('nouvelle requête', **A**) :

calculer $T_{\alpha n,R_j}$; $t_{\alpha n,R_j} \leftarrow T_{\alpha n,R_j}$; $t^{R_j}_{cum,\alpha n} \leftarrow T_{\alpha n,R_j}$;

S←S+ $t_{\alpha n,R_j} + \max\delta_{\alpha n}$;

Send-high(**S**, $t^{R_j}_{cum,\alpha n}$,∞,αn,αn,$\alpha 2$,R_j); // l'utilité de αn est ∞ pour le rôle impératif de αn

State(αn) ← **1;**

Etat 1: receive-high ('NewNeighbour', αi, αl , $\alpha i'$,Rj):

UpdateNeighbourHigh(αn , αi) ; **Send-high(S,** $t^{R_j}_{cum,\alpha n}$,∞,αn,αn,αi, R_j); //**reprise**

2. Au niveau de chaque agent αi $(i \neq 1, i \neq n)$:

Etat 0 : ***receive*** ('nouvelle requête', A) ;

State (αi)← 1;

Etat 1: case event of:

receive-low (S, $t^{R_j}_{cum,\alpha i}$, $U^{\alpha i}_{\min,R_j}$, arg($U^{\alpha i}_{\min,R_j}$), αi , αi , Rj):

// αi reçoit de son voisin du niveau inférieur αk

$t_{\alpha i,R_j} \leftarrow \max\left(T_{\alpha i,R_j} - t^{R_j}_{cum,\alpha i}, 0\right)$;

$U_{\alpha i,R_j} \leftarrow t_{\alpha i,R_j} + \max\delta_{\alpha i,\alpha k}$;

S←S+ $t_{\alpha i,R_j} + \max\delta_{\alpha i,\alpha n}$;

//mise à jour de l'utilité minimal jusqu'au niveau de l'agent αi en la comparant

//avec l'utilité minimale reçue jusqu'à l'agent αk

$U^{\alpha i}_{\min,R_j}$ ← ***minimum***($U_{\alpha i,R_j}$, $U^{\alpha i}_{\min,R_j}$);

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

$t^{R_j}_{cum,\alpha i}$ ← $t^{R_j}_{cum,\alpha i}$ +$t_{\alpha i,R_j}$; *// le temps d'attente cumulé jusqu'à l'agent αi*

UpdateNeighbourLow(αi,αk);

send-high(**S,** $t^{R_j}_{cum,\alpha i}$, $U^{\alpha i}_{\min,R_j}$, **arg(** $U^{\alpha i}_{\min,R_j}$), αi, $\alpha i+1$,**Rj**);

else *//* **S est dépassé**

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

send-low (**'NewNeighbour'**, αi, αk , $\alpha i+1$, R_j);

state (αi)← **2;** *// αi* **s'est retiré;**

else *send-low*('Invitation de se retirer', αi, **arg(** $U^{\alpha i}_{\min,R_j}$), R_j);

end if;

end if;

receive-high ('Invitation de se retirer', αk, αi, R_j):

$\alpha i'$ ← ***NeighbourHigh*** (αi ,Rj); $\alpha i''$← ***NeighbourLow***($\alpha i'$,Rj);

send-low ('NewNeigbour', αi , $\alpha i''$,$\alpha i'$,**Rj**) ;

state(αi)←**2** ; *//* αi s'est retiré de la négociation

receive-high ('NewNeighbour', αi, αi , $\alpha i'$,Rj):

updateNeighbourHigh(αi , $\alpha i'$);

send-high(S, $t^{R_j}_{cum,\alpha i}$, $U^{\alpha i}_{\min,R_j}$, **arg(** $U^{\alpha i}_{\min,R_j}$), αi , $\alpha i'$, **Rj**); *//reprise*

endcase;

Au niveau de l'agent du dernier niveau, seul un seul type de message peut être reçu correspondant à *Receive-low* message et procèdera de la manière indiquée dans l'algorithme.

Le résultat de cet échange de messages est une coalition qui maximisera l'utilité de la réponse à la nouvelle requête Rj en répondant dans les délais (T). La qualité de la réponse est relative aux agents qui ont traité la requête Rj avec les méthodes locales respectives. Les agents qui se sont retirés sont les agents dont les utilités sont minimales pour la requête Rj.

5. Analyse

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 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. Cette solution, bien que simple car il n'y a pas de retours 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 arrière mais garantie une maximisation de l'utilité de la réponse. La terminaison est garantie car le nombre de niveaux est fini. De plus, un agent qui s'est retiré (à cause de son utilité minimale) ne peut plus joindre la coalition pour le traitement de la nouvelle requête.

Conclusion

Nous avons proposé un scénario de négociation coopératif entre agents ayant le même but qui est de répondre dans les meilleurs délais aux requêtes des clients tout en maximisant la qualité de la réponse. La recherche d'une coalition qui maximise l'utilité de la réponse ne s'est pas fait 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, et les faire coordonner 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.

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

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

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

[5] : 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.

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

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

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

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

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

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

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