Searching Pareto-optimal solutions for the problem of forming and restructuring coalitions in multi-agents systems

Philippe Caillou, Samir Aknine, Suzanne Pinson

SEARCHING PARETO OPTIMAL SOLUTIONS FOR THE PROBLEM OF FORMING AND RESTRUCTURING COALITIONS IN MULTI-AGENT SYSTEMS

Philippe Caillou[1] — Samir Aknine[2] — Suzanne Pinson[3]

[1]LRI, Université Paris Sud 11, Bat 490, F-91405 Orsay
caillou@lri.fr
[2]LIP6, Université Paris 6, 8, rue du Capitaine Scott, 75015 PARIS Cedex 15, France,
Samir.Aknine@lip6.fr
[3]LAMSADE, Université Paris IX Dauphine, 1 place du Maréchal de Lattre de Tassigny, F-75775 Paris
pinson@lamsade.dauphine.fr

ABSTRACT. Coordination is one of the fundamental research issues in distributed artificial intelligence and multi-agent systems. Current multi-agent coalition formation methods present two limits: First, computation must be completely restarted when a change occurs. Second, utility functions of the agents are either global or aggregated. We present a new algorithm to cope with these limits. The first part of this paper presents a coalition formation method for multi-agent systems which finds a Pareto optimal solution without aggregating the preferences of the agents. This protocol is adapted to problems requiring coordination by coalition formation, where it is undesirable, or not possible, to aggregate the preferences of the agents. The second part of this paper proposes an extension of this method enabling dynamic restructuring of coalitions when changes occur in the system.

KEY WORDS: multi-agent system, coalitions, preferences, aggregation, reorganization.

1 Introduction

The search for economic efficiency has led to the division of labor between specialists. Today, similar reasoning explains the success of agent-oriented programming and multi-agent systems. Programs are increasingly complex and have multiple functions which must sometimes be updated or improved. Using a set of specialized agents which coordinate their complex tasks gives more flexibility, efficiency and evolutivity to programs. To perform complex tasks, agents need to coordinate, either because tasks require many resources if they are to be performed by a single agent, or because certain sub-tasks can be carried out more efficiently by specialized agents (Binmore, 1999, Ossowski, 2000, Wooldridge, 1999).

Agents have specific capabilities and are programmed to carry out certain tasks. For a given agent complexity, the search for more sophisticated capabilities leads to a higher number of agents. Since agents are more and more specialized, they are not able to perform complex tasks alone and must necessarily coordinate their tasks with others. More generally, a multi-agent system is made up of several homogeneous or heterogeneous agents which communicate between themselves(Wooldridge, 2001).

How can autonomous agents be coordinated efficiently? One solution is to look for groups of agents which are able to perform the desired tasks better than one agent. This means that agents may form coalitions, a coalition being a temporary association between agents in order to carry out joint projects. The aim is a better distribution of capabilities in order to achieve a complex project, but this is not the only method of coordination. It can be imposed by a hierarchy, carried out by bilateral contracts(Smith and Davis, 1981), etc.(Sandholm, 1999)

The choice of solving a problem using the coalition model depends really on the type of problem under study. Coalitions are well adapted when there are strong externalities (when completing a task influence the utility involved in the resolution of an other task) and/or when interactions between agents are such that the contribution of an agent within a coalition depends on which agents the coalition contains, in which case a bilateral contract would be difficult to negotiate. For instance, if the payoff of an agent is the marginal utility it brings into the project, the given payoff envisaged would vary during negotiations according to the members joining the coalition or leaving it.

Once coalition formation is chosen as a coordination method, the definition of the corresponding protocol remains problematic. A coalition formation protocol is strongly dependent of the type of problem studied. The fact that the agents do or do not have the same objective, do or do not trust each others, are examples of parameters which may generate different protocols of coalition formation.

To enable the agents to form coalitions, all current protocols make the assumption that the utility functions of agents, which measure their degree of satisfaction for each suggested solution, must be comparable or identical. This means that agents must be able to agree on a common utility function, either of all the agents as in (Shehory and Kraus, 1998), or of their coalition as in (Aknine, Pinson, and Shakun, 2000) and (Vauvert and El Fallah-Seghrouchni, 2001). This assumption seems acceptable for

most multi-agent systems, in particular for productive projects where all utilities can often be calculated in terms of profit. However, in many cases, to compare the utilities of agents, and even more to aggregate them, is difficult. The numerical evaluation of an agent utility is already a strong assumption compared to the simple classification of available choices (Pareto has already shown the many advantages of ordinal utility compared to cardinal utility in economics in the XIX century). To compare the utilities of two individuals is stronger. Why should a solution weighted 8 by one agent and 6 by another be preferred to one weighted respectively 4 and 7? Our model does not suppose that the utilities of agents must be aggregated or compared(Caillou, Aknine, and Pinson, 2002a).

A second limitation of current models lies in assuming that all calculations are recomputed as one condition changes (an agent joins or leaves a coalition, a task is added or canceled, etc.). However these protocols are very complex and these changes can be very frequent. Using the information obtained in a previous execution of the protocol, i.e. a dynamic reorganization of the coalitions formed, could reduce calculations. This is the second aim of our model(Caillou, Aknine, and Pinson, 2002b).

This article is organized as follows. Section 2 describes the application used to illustrate our model. Section 3 introduces some definitions. Section 4 details our methods of coalition formation and dynamic reorganization of coalitions. Our model gives a wide choice of agent behavior. Section 5 proposes a set of behavior models in order to improve the computation time. Section 6 presents an application example of the protocol and discusses the implementation of our model. The results obtained with our models are discussed in section 7. Section 8 analyzes related work. Section 9 draws a general conclusion from this work and proposes some perspectives.

2 Application

The suggested protocol is particularly suitable for problems with complex tasks (where there is a need for several agents and for coalitions) and for dynamic problems (tasks may be added, others canceled or modified constantly) with different utility functions of agents. We assume that agents are cooperative, i.e. they trust each other in searching for and applying the solutions. Their utility functions are unknown by the other agents and do not need to be cardinal, an ordinal utility is enough. Agents just need to be able to choose between two situations (or to be indifferent), they can thus be self-interested.

A good example of this problem is a distributed teaching schedule at university. This application illustrates the dynamic evolution of the coalitions, as often a course may be added or removed, or a professor or a group of students may join the establishment. In this example we consider two types of agent: professors and students. Student agents represent homogeneous groups of students with a common utility function and with the same classes. Of course, it is possible to have an agent

for each student and thus to enable him or her to have its own utility function. However, the computational complexity will be greatly increased.

The classes correspond to the tasks to be carried out. Thus, agents form a coalition for each class. Most coalitions are formed of two agents: a professor agent and a student agent (having more agents in a coalition is also possible, for instance for lectures with several groups of students). Each (student or professor) agent defines the utility it assigns to each schedule. Since its utility function is ordinal, it just needs to be able to compare two schedules and to say which one it prefers or if it is indifferent. Agents are free to choose their parameters while computing their utility. A professor can thus prefer the morning, refuse Mondays, prefer certain classes, like a stable schedule, etc.

In a general way, the choice of an agent depends on the members of the coalitions in which it will take part, but its appreciation of a coalition may also change according to the other coalitions. This introduces externalities or an ordered processing of tasks. Thus, if a task T_i must be carried out before T_j, the utility that the agent will associate to T_j will be null if no agent takes part in the coalition which performs T_i (task T_i is then not performed, and T_j is of no interest). The agent choice may also depend on parameters which are related to its preferences and which vary in time. Thus, it can be against change. For instance, a professor may prefer one schedule to another because it is closer to the current situation. The only constraint is that these external parameters need to be stable during a negotiation step.

3 Definitions

This section presents some definitions necessary to understand our model and which we use in the rest of this article.

Coalition: a coalition is formed for each task to be performed. It contains zero, one or more agents which will carry out actions in order to achieve this task. Each action and its parameters are defined (for instance, the parameters of the action “taking a class” are: the week, the day and the time).

Coalition set: a coalition set represents a solution to the problem of coalition formation. It contains as many coalitions as tasks to be performed at a given moment (in our application, a set corresponds to one schedule).

Group of coalition sets: a group of coalition sets corresponds to several sets of coalitions brought together in order to be computed and transmitted collectively (for instance, several possible schedules). In the rest of this article, it will be referred to as a group of sets or simply a group. When an agent computes a group of equivalent sets, this means that it is indifferent regarding all the sets of coalitions in this group (for instance, it computes a group with those schedules that it prefers to others and that it cannot classify).

Context: a set of unspecified parameters which must be stable during a negotiation step. For instance, it may concern a date or any external parameter.

Utility function*:* the utility function may be ordinal or cardinal. If it is cardinal, it associates a utility with a set of coalitions within a given context. If it is ordinal, it compares two sets in a given context. In this case, measuring the utility of a set means comparing it with a reference situation which will be the same one throughout the negotiation.

Reference situation*:* In order for the agents to know if they have to accept a set of coalitions as a solution, they need to be able to compare it with what they are sure to obtain during the negotiation. This minimum is the reference situation. If no coalition has yet been formed, the reference situation is the situation where nobody does anything. If there are already coalitions, it is the current situation, with possibly some changes in order to take into account new information (cf. section 4.3.2). To be sure to find a solution after a negotiation, the reference situation needs to be feasible and to be the same for all the agents (a demonstration is proposed in section 4.3.3).

Acceptable set*:* we consider that a set is acceptable for an agent if it is preferred or equivalent to the current reference situation.

Pareto optimum*:* a Pareto optimum is a situation where it is not possible to improve the situation of an agent without deteriorating that of at least one other. Graphically, for two agents a situation is optimal if no other situation exists at the top right position of the situations considered.

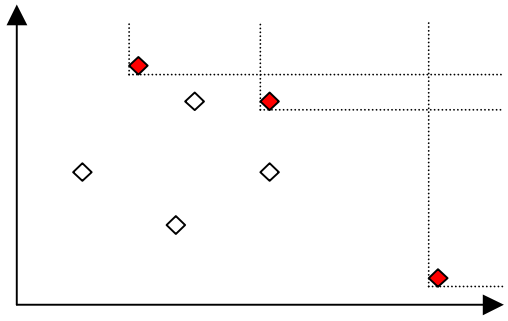


Figure 1. Example of Pareto optimal solutions

4 Coordination methods

Our first aim in defining this protocol is to solve the agent coalition formation problem without having to aggregate the preferences of the agent . Then, we extend this protocol to allow a dynamic and fast reorganization of these coalitions according to new changes in the multi-agent system.

4.1 Presentation

As we do not intend to aggregate the utilities of the agents, we seek a solution which is "objectively good", i.e. which may not be contested by any agent. A logical criterion likely to be accepted by all the agents is that we cannot increase the utility of an individual without deteriorating that of at least one other. If this does not happen, i.e. there is a situation such that we can increase the utility of an individual without deteriorating that of another, there is no reason not to prefer this situation. The solution we seek must thus be a Pareto optimum.

Which Pareto optimum should we choose? Now the problem is to compare the utilities of different agents. How should we choose between a schedule which is the first choice of a professor and another which is the first choice of a student? One solution is to avoid making a choice but to find a Pareto optimum. This offers the advantage of reducing computations as agents do not have to compute all possible schedule. The only constraint is that it should be in the interest of each agent to accept this solution, therefore to prefer this solution to the initial situation. The first aim of our protocol is thus to find a Pareto optimum likely to be accepted by all the agents and as early as possible. To find a Pareto optimal situation is a first step. Extensions of this protocol to find a more equitable solution are under study (see for example(Aknine and Caillou, 2004)). But this is a necessary step, because finding a distributed way to compute a Pareto optimum without transferring or aggregating preferences is not trivial.

4.1.1 Principle

How is a Pareto optimum obtained?

- The agent which initiates a negotiation seeks one or more sets of coalitions it prefers (cf. section 5) and chooses an agent to which it sends them (cf. section 4.2.1). Then it seeks the set(s) that it would choose as a second choice and sends them to that agent, and so on, until there are no more sets at least equivalent to the current situation.
- When an agent receives a group of sets, it evaluates them and sends them to the next agent sorted in decreasing order of preference.
- When an agent receives a group of sets, if there is at least one set which is preferable or equivalent to the current situation and if all the agents have already taken part in the negotiation, the set of this group that it prefers is a Pareto optimum and may be used as a solution set for the negotiation.

For instance, let us consider two agents and seven sets of possible coalitions. Let $E(U_1;U_2)$ be the relative utilities of agents a_1 and a_2 for the set E. Having E_0 as the initial situation, the seven possible sets are: $E_0(0;0)$; $E_1(0;10)$; $E_2(2;8)$; $E_3(4;8)$; $E_4(4;5)$; $E_5(-2;2)$; $E_6(10;-1)$ (cf. figure 1). Of these seven sets, three are Pareto optima (E_1, E_3 and E_6). We represent these solutions on a plan according to the utility that they bring to each agent (cf. figure 2).

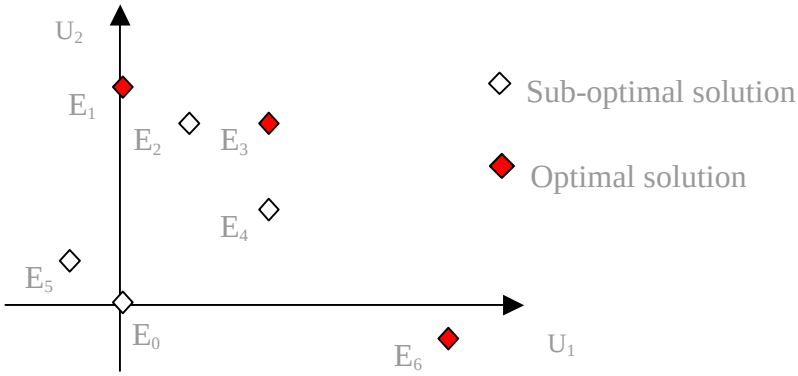


Figure 2. Describing all possible solutions in a utility space

Agent a_1 initiates the negotiation. It sorts all the acceptable sets for it into equivalent groups of sets (cf. figure 3): $G_1(E_6)$; $G_2(E_4;E_3)$; $G_3(E_2)$; $G_4(E_0;E_1)$. E_5 is not sorted as the reference situation (E_0) is better.

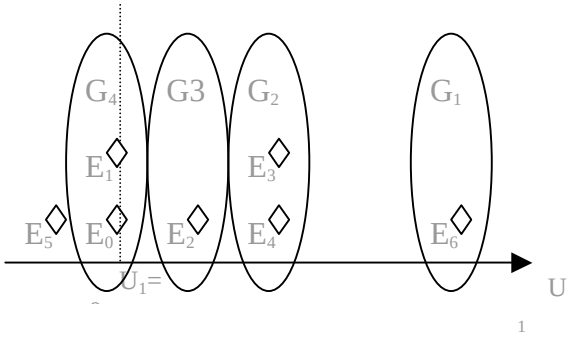


Figure 3. Group of sets of agent 1

Groups G_1, G_2, G_3 et G_4 are acceptable for agent a_1 as they correspond to a situation which is as satisfactory as the initial reference situation, or better. G_1, G_2, G_3 et G_4 are sent in this order to the next agent. Thus, agent a_2 starts by receiving G_1 and evaluates it (cf. figure 4). Set E_6 is unacceptable for the agent because it would bring a less satisfactory situation than the initial situation (figure 4). The agent does not send this set and waits for the rest.

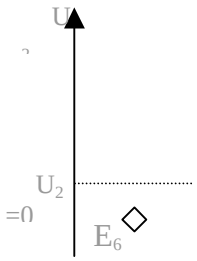


Figure 4. First group for agent a2

It receives G_2 and sorts it into two sets (figure 5) in two groups $G_{21}(E_3)$ and $G_{22}(E_4)$. G_{21} is acceptable. As all the other agents have already participated in the negotiation, agent a_2 cannot send it. All the sets of G_{21} can thus be a solution. The agent must choose E_3, which is Pareto optimal. It sends this set to agent a_1 in order to inform it of the result of the negotiation.

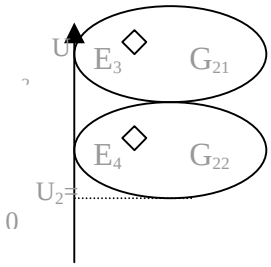


Figure 5. Second group for agent a2

4.1.2 Algorithm

The negotiation process is based on three phases: initialization of the negotiation and transfer of tasks, negotiation, transmission of the solution. We can distinguish the behavior of the agent which initiates the negotiation from the intermediate and final agents which take part in the negotiation. The order of the agents can differ from one negotiation to another and each agent can be in any position. However, the order must be stable during a given negotiation. The importance and influence of this order will be discussed in section 4.2. In short, this protocol can be seen as a distributed lexicographic search in a virtual common preference space (this problem is not trivial

since no agent has a complete knowledge of this space, because the preferences are not transmitted between agents).

4.1.2.1 Phase 1. Initialization of the negotiation and transfer of tasks

Any agent can initiate the negotiation. This action can be initiated when a new task appears or when an agent modifies its preferences. The initiator agent informs the others that it begins a new negotiation and any agent which wants to begin another negotiation must wait until the end of the negotiation in progress. To avoid conflicts between two simultaneous requests, each agent sends a confirmation. Each agent asks the other agents to send it their tasks, deduces the set of tasks to be performed and associates each one with a coalition. The initiator agent computes all possible sets of coalitions (cf. sections 4.1.3. and 5. regarding complexity), gathers them in a group of sets and sends this group to the agent which would initiate the negotiation.

4.1.2.2 Phase 2 : Negotiation

When an agent receives a group of sets, it sorts the sets in order of preference into homogeneous new groups of sets. In these groups, all sets are equivalent in terms of agent utility. The agent sorts only those sets that are at least equivalent to the reference situation and the others sets are not considered.

If the agent is not the last agent, it sends its new groups to the next agent in decreasing order of preference. If it is the last agent, and if this agent has created new groups because it has found acceptable sets, it considers that all the sets of the best new group are Pareto optima. It can thus choose one of them randomly and this will be the solution for the negotiation

4.1.2.3 Phase 3: Transmission of the solution.

Once the last agent has identified a Pareto optimal solution, it sends this set to the other agents which accept it as the solution of the negotiation (Remind that the goal of the distributed negotiation was to find a Pareto optimal solution).

4.1.3 Importance of the choice of the next agent

The order in which agents negotiate influences the result. The first agent is the one which has the strongest impact on the final solution as it is the first to choose the sets it prefers from among all the possible sets, and to send them to the following agents. The choice of the next negotiator agent is also very important.

The first solution is to choose randomly among those which have not yet participated in the negotiation. However, to improve the computation time of the protocol, it is preferable to take the agent which appears the most often in the

computed sets. We assume that, since it takes part in many coalitions, this agent will be more interested in the alternatives which will be proposed than an agent which is less involved. It will thus sort the sets into several groups (it will possibly reject some of them if it considers them as unacceptable). The next negotiator agent will receive smaller groups which means that it will have less computing to do.

The second solution consists in choosing the agents in a predefined order. This makes it possible to favor agents with high priority. This solution is very practical in many real applications, such as in drawing up schedules, where professors have priority over their students.

A third solution is to let the agent choose the next agent which will maximize its utility. This optimization may be complex (each agent ignores preferences of others) but it gives more freedom to the agent who has to make a strategic decision.

4.1.4 Parallel computation

One advantage of the protocol is that agents evaluate and rank coalition sets in parallel (once first group of coalition sets is transferred). They do not rank the same group at the same time, but they work on different groups at the same time.

For example, consider four agents A_1, A_2, A_3 and A_4. A_1 computes groups in preference order G_1 to G_n and sends them in this order to A_2. A_2 first receives G_1. It evaluates acceptable sets in G_1 and rank them in G_{11}.. G_{1m}. It sends them in this order to A_3. While A_3 is evaluating G_{11} (it builds groups G_{111} to G_{11k} with acceptable sets of G_{11} and sends them to A_4), A_2 can work in parallel on G_2 (and build G_{21} to G_{2m}).

Let us say that no coalition set in G_{11} is acceptable for A_3. It will consider G_{12} and compute G_{121} to G_{12p}. Note that the groups are received by each agent in a lexicographic order (for A_3: $G_{11,}$ $G_{12, ...,}$ $G_{1m,}$ $G_{21,}$ $G_{22,}$ …). As mentioned in section 4.1.2, the selected Pareto optimum is chosen thru a distributed lexicographic search in a virtual common preference space.

This example also illustrates the importance of cooperative agents: if an agent makes the strategic choice to “lie”, which here would mean to consider and evaluate a group G_i while a group G_j has been received before, there is no assurance to obtain a Pareto optimum anymore. Agents may have opposite objective, but they have to respect the order required by the protocol.

4.1.5 Using undeveloped coalitions to improve the computation time of the algorithm

As the first agent starts by computing all the possible combinations for all the tasks, this process implies a huge the computation time of and volume of data sent to the following agents. A way to improve the computation time without decreasing information quality, and thus the result and the properties of the algorithm, is to use and transmit undeveloped coalitions, i.e. the tasks for which all possible coalitions have not yet been computed. If an agent receives an undeveloped coalition in a set and this coalition does not affect its utility (if it joins the coalition or not), it leaves it aside and does not compute it. If it does affect the utility, it computes all possible

combinations for the corresponding task. Considering our assumption, the result of this computation is the same whatever the agent which does it.

For instance, in drawing up schedules, a professor agent which begins the negotiation will only develop coalitions related to classes it is likely to give, because they are the only ones which can modify its utility. More precisely, let us assume that there are two classes c_1 and c_2 (c_1 can be only given by professor p_1 and c_2 can only be given by professor p_2); two possible time slots; a group of students (s_1) and let us also consider that the utility of each professor depends only on their own classes. If p_1 begins the negotiation without using the undeveloped coalitions, it must compute and evaluate all 9 possible sets (3 possible coalitions for c_1 (s_1 and p_1 with two possible time slots, plus the course not given) multiplied by 3 possible coalitions for c_2). It classifies them in groups and then sends them to the following agent (let us say p_2). By using the undeveloped coalitions, p_1 has only 3 sets to evaluate (made up each time of one of the three possibilities for c_1 and not specifying anything for c_2), that it classifies in groups and sends to p_2 which is asked to develop c_2 (see another example in section 5.2.1)

This method produces better results if the agent's utility depends only of few tasks. However, if an agent's utility depends on all the tasks, it will be asked to develop all possible sets when it takes part in a negotiation.

In our protocol, only a few changes are necessary in order to use undeveloped coalitions. At the end of the first phase, the initiator agent sends to the agent which will begin the negotiation a group of sets containing one set of undeveloped coalitions. In the second phase, when the agent receives a group, for each set of this group and for each undeveloped coalition of this set, the agent checks if the corresponding task can influence its utility. If so, it computes all possible coalitions corresponding to this task, adds the new sets of coalitions to the sets it must evaluate and removes the set which contained the undeveloped coalition.

4.2 Formal analysis of our model

4.2.1 Why is the solution Pareto optimal?

How can we be sure that the first set received by the last negotiator agent is Pareto optimal, as all possible sets have not yet been evaluated by all the agents? A demonstration is necessary.

Proposition 1.

When an agent receives a group of sets, if:
- all other agents have already participated in the negotiation,

- at least one of the received sets is acceptable, i.e. it is at least equivalent to the reference situation,
- none of the sets previously received during the negotiation satisfies the two conditions below,

the acceptable set(s) S that it prefers in the received group is/are Pareto optimal and can be used as a solution for the negotiation.

Demonstration.

If S is not Pareto optimal, this would mean that there is a set S' which is preferable for one of the agents that shall be called (a_i). S' is at least equivalent to S for all the other agents. In this case, all the agents which were before a_i in the negotiation have transmitted S' either in the same group as S (if they are indifferent), or in a previous group (if there is at least one agent which prefers S' to S).

If a_i receives S' before S, it had necessarily sent it (since S is a solution, S is acceptable for all agents, therefore S', which is at least equivalent to S, is also acceptable for all agents). As the groups are computed completely before starting with the next group, a_i returns S' before computing S. The following agents should thus receive S' before S. Since S' is also acceptable, they send it to the following agent and so on until the last one which will therefore find it acceptable and thus select it as a solution, which is impossible since S has been selected.

If a_i receives S and S' in the same group (all previous agent have considered S equivalent to S'), a_i should send S' before S as it prefers S' to S. As in the previous case, agents following a_i should receive S' before S. Since S' is acceptable, they should then send it to the next negotiator agent, until the last agent which should also find it acceptable and should therefore select it as a solution for the negotiation, which is also impossible since S has been selected.

Consequently, it is impossible for a set S' to exist such that an agent prefers S' to S and that all the other agents find it at least equivalent to S. Therefore S is Pareto optimal.

4.2.2 Why are agents sure to find a solution?

The first optimum S found is the first set which is received by the last negotiator agent and that this agent considers acceptable. Why is there always an optimum? A demonstration is thus necessary.

Proposition 2.
The protocol always provides at least one solution to the problem.

Demonstration.

For each agent, the acceptability criterion is that the set is at least as satisfactory as the reference situation. However this reference situation is the same for all and belongs to the possible sets. Therefore all the agents necessarily find this situation acceptable and will forward it. Thus, there will always be at least one acceptable set which will reach the last negotiator agent. If the reference situation is the first set to arrive, it is an optimum and also the solution for the negotiation. If another acceptable set had arrived first, this one would provide the solution.

We can note that reference situation is here considered as “acceptable” for all the agents. If it is not (i.e. if the initial situation is not acceptable for at least one agent), the protocol is still working: the agents just have to consider the reference situation as the “worse acceptable situation”, and if the final solution of the negotiation is this reference situation, this means that there is no solution to the problem.

4.3 Dynamic restructuring of coalitions

4.3.1 Principle

Our protocol provides a solution, i.e. a set of coalitions with the initial conditions (utility functions, a set of tasks and a context). What happens if a change occurs in one of these conditions, for instance if a task is added or removed, or if an agent modifies its utility function? In current protocols (Aknine, Pinson, and Shakun, 2004b, Sandholm and others, 1999, Shehory and Kraus, 1998), all the computation must be redone to find a new solution to the problem. We propose a more efficient solution which use the results obtained in the current situation. It adds new information to the previous conditions, instead of completely replace them.

A simple means to use earlier computation is to start from the current solution. Instead of evaluating the different sets compared to the initial situation where no agent does anything, the agents will evaluate the new solutions compared to the current solution. As this solution is at least equivalent to the initial situation for all the agents (since it is Pareto optimal), it is difficult to find a similar or better one. Thus, fewer sets and groups of sets will be forwarded and evaluated. This will accelerate the problem-solving process.

The change which has initiated the renegotiation may of course affect the utility of the agents, this is why those agents must reevaluate the sets that they have computed. Computation time is lower because the new reference situation has a higher utility level, which implies less acceptable sets to compute..

The new reference situation must remain feasible and identical for all agents in spite of the new information. Thus it is not the current situation which is used as the reference situation but the modified current situation, in which all the changes have been taken into account. For instance, for an agent which leaves, the reference situation will be the current set of coalitions without this agent. For a removed task, it

will be the current set of coalitions minus the coalition corresponding to the task. If it is impossible to obtain a new reference situation that is feasible, the initial reference situation (no one does anything) is used (and agents come back in the non-dynamic configuration).

4.3.2 Why is the solution Pareto optimal?

The demonstration of the first proposition (cf. section 4.4.1) is still valid: when the last agent receives a group of sets, if it has not yet received an acceptable set and the best set S of the received group is acceptable, set S is a solution of the negotiation. Moreover, there is no other set S' which is at least equivalent to S for all the agents, and preferable for at least one of them, otherwise the last agent would have received it before receiving S and this set would have been selected as a solution.

4.3.3 Why do the agents always find a solution?

The demonstration of the second proposition (cf. section 4.4.2) is still valid: the reference situation is the same for all agents, acceptable by all agents (as it is compared with itself) and also feasible. Thus, there is at least one set (the reference situation) which will be sent by each agent to the next agent and which will be the solution if it is the first to be received by the last agent.

5 Behavior models of the agents

How do the agents process to find the sets of coalitions to send to the other agents? The answer to this question has a great impact on the computational computation time of the solution. It is appropriate to analyze in detail the various possible methods in order to select the most appropriate one with the minimum computation.

5.1 Objective

The aim of each agent is to build new groups of homogeneous coalitions from a group of sets received from the previous negotiator agent and to classify these new groups in order of preference. This means that the agent must be indifferent to all the sets of the group, it must prefer these sets to all the sets of the lower groups and prefer all the sets of the higher groups to them. Heuristics can be used to find the best group according to the context and the application. This improves the computation time of the algorithms.

The simplest solution is that the first negotiator agent computes all the possible sets and then each agent makes an exhaustive classification of all the possible sets. The advantage of this solution is that it is simple, but it leads to a high computation time, especially for the first agent. Other search methods can serve to improve the computation time and to distribute the calculations among the various agents. Even if the theoretical worst-case complexity remains the same, experimentations shows a much better average case (see section 7).

5.2 Using heuristics to reduce the computation time of the behavior of the agents

To illustrate these heuristics, we will use a simple example with one teacher-agent (P1), two student-agents (E1, E2), two classes for each student (four coalitions in each coalition set), and four time slots (H1, H2, H3, H4). The total number of possible coalition sets is 5^4=625 (5 for the four time slots and the not-given case).

5.2.1 First heuristic: Using undeveloped coalitions

The method proposed using undeveloped coalitions (presented in section 4.1.3) reduces the calculations and the volume of the information transferred while preserving the ease of calculation by the agents.

Applied to our example, the first student begins the negotiation. It simply evaluates coalitions sets by adding the hours of its classes (for example if it has a class at H2 and one at H4, it evaluate each set with these parameters with a value of 2+4=6) and 0 if a course is not given. Using undeveloped coalitions, agent E1 has only to develop coalitions corresponding to its classes. It will have a total of 5*5=25 sets to send to the next agent. Figure 6 represents these sets and their evaluation. It first develops coalition corresponding to its first class and obtain the 5 partially developed sets EnsPart1 to EnsPart5. Then, it develops coalition 2 and obtains the 25 coalitions sets Ens1 to Ens25. It evaluates them and places them into groups. The first group G1 contains its preferred sets, Ens14 and Ens18, both with a utility of 7. Next group contains Ens9 and Ens17. Last group will contains all sets with a utility of 0, equivalent to the situation where no course is given.

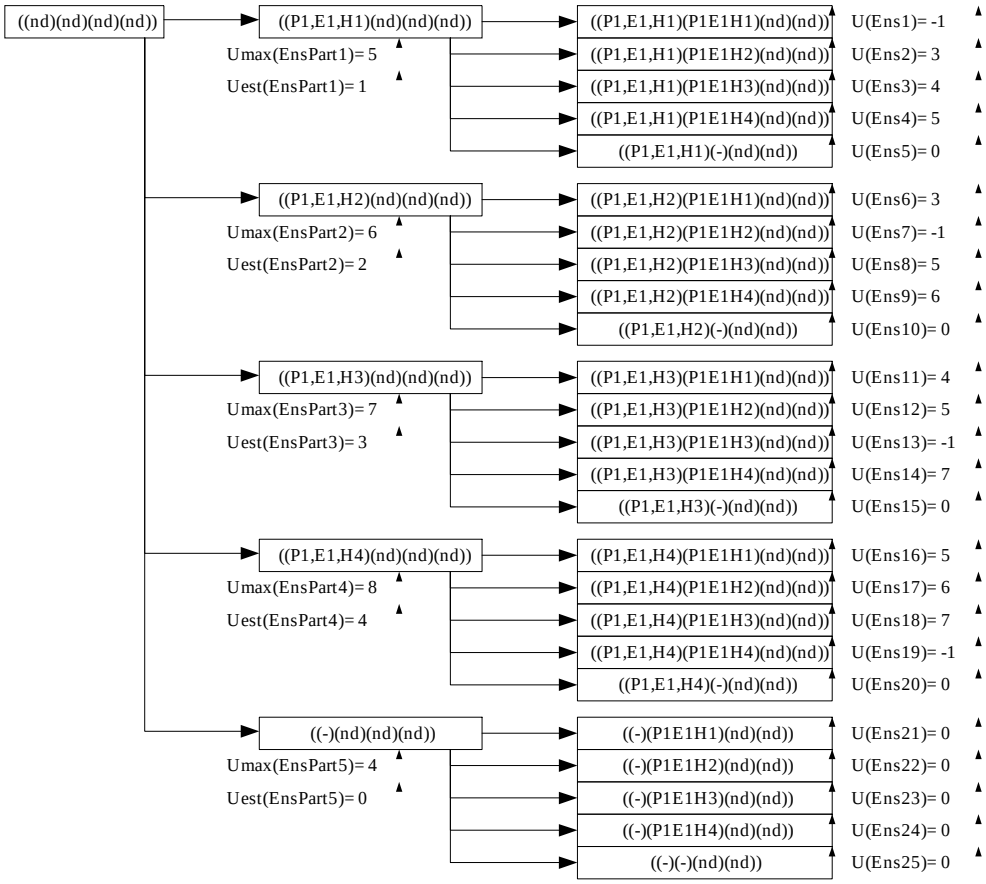


Figure 6. Development of all coalition sets for student-agent E1 using undeveloped coalitions (nd) for the two student-agent E2 classes

5.2.2 Second heuristic: Tests of intermediate acceptability

In order to reduce the number of iterations, a complementary solution would be to test if an (incompletely developed) set can be potentially preferred to the reference situation. If this is not the case, it will not be necessary to develop it and this branch of the exploration tree can be pruned. These tests are especially useful during the restructuring of coalitions. The reference situation would then be the current situation that is likely to be very acceptable for the agent. This agent can easily set aside many sets which will not give a better solution, especially if the agent prefers not to change its situation. All the solutions which begin to move away from the current solution are thus quickly dropped because the agent will necessarily prefer the reference situation to them.

In the example, the student agent will use the intermediate evaluation Umax(EnsPart) which evaluates the maximum utility reachable by the set. In this

case, if the reference situation is evaluated 7, the agent will know it is useless to develop EnsPart1, EnsPart2 and EnsPart5 and it will just concentrate on the two other situations, computing only 5 intermediate evaluations instead of 15 final evaluations.

5.2.3 Third heuristic: Search limited to the best group

The aim of an agent is to send to the next negotiator agent groups of sets sorted in decreasing order of satisfaction. If the solution is in group Gi, all groups Gj with j>i have been evaluated, classified and probably developed unnecessarily. It would be useful to only evaluate the sets of G1, then those of G2, and so on. The problem is that agents do not know in advance what will be the degree of satisfaction associated with the best group. However, in order to evaluate only the members of G1, it is necessary to know the satisfaction associated to them, and therefore to have already evaluated them! Even if it is impossible to compute only the sets of the group G1, we can try to gradually limit computations to the useful sets. To do so, the agent needs a lower limit, which is the best evaluated set at the current computation time, and it will only develop the sets which are at least equal to this limit. Each time a set, even an incompletely developed one, is evaluated and the evaluation is higher than the limit, it becomes the new limit. On the contrary, when an evaluated set does not reach the limit but is nevertheless acceptable in a weaker group, it is kept and added to a group which will be used as a starting group to compute the following groups.

In the example, the agent first develops the first coalition and obtains the 5 partially developed sets EnsPart1 to EnsPart5. It chooses one randomly, say EnsPart3 and develops it. It obtains Ens11 to Ens15. The best set is Ens14, rated 7. The best group has so a minimum rating of 7 and the agent searches only sets with a minimum rating of 7. It uses the intermediate evaluation Umax for each remaining partial set and consider only those which have a potential rating of 7, in this case only EnsPart4 (Umax(EnsPart4)=8). It develops EnsPart4, adds Ens18 to the actual best group G1 and has terminated to create this group. It can thus send it to the next agent who can begin its evaluation. In parallel, first agent continues its search with remaining sets to find the second best group.

5.2.4 Fourth heuristic: limited search using intermediate evaluation

In the previous case, the order in which the coalitions are developed is of great importance. The faster the best set is reached, the faster it becomes the reference situation and the less the other sets are developed (because the reference situation becomes rarely reached). This is thus useful to set up an intermediate evaluation procedure of the sets to be developed in order to compute first of all the set which seems most likely to generate sets bringing great satisfaction.

In the example, This means that instead of choosing randomely between partially developed sets, the agent uses an evaluation function (Uest) to choose the best set considering its information at this time. Here, it will choose to develop in first EnsPart4, which has the best evaluation because the only developed class is in H4, which is its preferred slot. It will then find the best group rating quicker than having chosen randomly for example the EnsPart5 set.

5.2.5 Prospective search.

In order to better use the utility function, instead of starting from an empty set and developing it, an agent may immediately use the knowledge of its utility function and the tasks to be achieved in order to deduce the best sets. If the number of possible sets is high, this solution can be advantageous since in this case the complexity does not depend of the number of possible sets but depends of the type of utility function of the agent. This method can give far more effective results but the procedure for each type of utility function needs to be rewritten.

5.2.6 Non-exhaustive methods

Using the utility function or developing the possible coalitions, the agent can make approximations in order to have much faster results, even if it is not certain to obtain the best possible results. Using this method to develop the coalitions, the agent can decide not to explore the undeveloped sets - which do not offer very interesting prospects –even if one of their developments may theoretically give the best solution. The final solution will thus be obtained quicker, but this solution would not be proved to be Pareto optimal.

6 Implementation and tests

To illustrate our model, we have implemented a teaching scheduling application system using the utility function of the professors and the students. The utility function has different variables: for each day, the time of the first class, the time of the last class, the number of hours per day, the number of classes not given, the number of compulsory classes not given for each agent, the number of changes compared to the current schedule, the total number of hours per week. Given the number of parameters, three profiles have been defined to simplify the choices: morning, afternoon and grouped (it prefers to group its classes on a minimum number of days). These profiles correspond to values of arbitrary parameters used for the tests. Because it is easier for implementation purposes, we have chosen to compute a utility function in order to valuate the preferences. Even so we get cardinal values, our proposed protocol only needs ordinal utility functions.

6.1 Description of the multi-agent system

6.1.1 Architecture of the system

The multi-agent system is composed of two principal object-oriented classes: the agent and the environment. The environment aims at identifying the newcomers, at

carrying out a total follow-up of the system and at enabling the user to intervene if needed. During their creation, the agents obtain the IP addresses of the other agents which they then use to communicate directly with them. Our aim is to build a generic multi-agent system that can be used easily to redefine an environment and agents adapted to the field to which we want to apply them, and to define tasks, actions or behaviors of the agents (maximization of collective utility, individual utility, etc.). The system is made up of several executable programs. The goal is to make the environment and each individual agent really autonomous entities.

6.1.2 Communications

The agents and the environment communicate by TCP/IP. The various agents can thus be distributed on different computers and communicate in a local area network or by Internet. This distribution of the resources allows a greater speed and a real parallelism between actions and deliberations. Agents subscribe to the environment, which IP address must be known by all. At the time of their subscription, the environment sends the agents the addresses of the other agents in the multi-agent system and informs the others of the presence of the newcomers. All the agents thus have all the addresses and can communicate with each other. To communicate, the agents and the environment use object messages, which are structured vectors so that they can be created, sent and received. All the agents and the environment have a message processing program permanently on standby so that these messages can be received and processed immediately.

6.1.3 Description of the environment

At the beginning, the environment agent is activated and then the agents start the negotiation process. This allows:

- The user to intervene in the system as a whole.
- To inform the user of the global state of the system.
- To register the entrance of a new agent, to provide it with information concerning the current state of the multi-agent system (addresses of the other agents) and to inform the other agents of its arrival by providing them with its address.

6.2 Description of the agents

We assume that an agent is an autonomous program which behaves for its own interest. It is mainly characterized by its action and reasoning mechanisms and by its knowledge structure. The agent has knowledge about the various types of task that exist in the domain considered and about the various types of action. In addition to

this knowledge about itself, the agent has knowledge about the other agents. This knowledge is provided to it by the other agents when they join the system (or when this agent itself joins the group, if the others were in the system before it) or when they modify one of their characteristics (like adding a course to or withdrawing a course from the list of capabilities of a professor, in the case of our application). The action mechanisms (the behavior) are free. Just like the environment, the agent uses a process which allows it to wait for possible communications from the other agents or from the environment.

6.2.1 Negotiation process of the agent

Although this part of the agent is always reusable for other applications, this module knows the coalition formation protocol and can thus negotiate so as to reach a Pareto-optimal situation.

6.2.1.1 Structure

The agent is made up of three processes, all of which are activated during initialization:

- An action process, which carries out the actions that the agent planned.
- A negotiation process, which is activated as soon as a new negotiation starts.
- A communication process, which waits to receive messages from the other agents or the environment, and consequently communicates with the other processes.

6.2.1.2 Managing communications

The negotiation process of the agent manages the negotiations in order to form coalitions. For that, it knows a certain number of specific messages: "*Initialization*" and "*Update*" are messages sent by the user or by an agent. "*Initialization"* message starts a new coalition formation process, "*Update*" enables to modify current coalitions. The agent starts by warning the other agents of the new negotiation (with the message "*New Negotiation*", to which they answer by "*Confirmation New Negotiation*"). Then it begins the negotiation and gradually sends the groups of sets of coalitions to the following agent (message "*Negotiation in Progress*" to which the agent answers "*Evaluation Ended*" each time that it has finished processing a group).

6.2.1.3 Initializing a new negotiation

When an agent receives a group of sets from another agent, it saves these sets in a vector of sets which will be analyzed with the process used for analyzing the group. When the agent initializes a new negotiation, it places, in the same vector a single initial set in which no coalition is developed and it fires the same analysis procedure. In order to initialize a negotiation it is sufficient to receive an empty set from the others. In all cases, the agent initializes the groups where it will classify the sets according to the satisfaction that these sets provide it.

6.2.1.4 Searching for coalitions

To classify the sets in groups, the agent has several methods which are described in section 5.2. The first, the basic method, has been improved thanks to the addition of intermediate tests and by developing only necessary coalitions. The second, the search method limited to the best group, means major modifications to the algorithm while moving from breadth search to an in-depth search. This is because the agents must obtain completely developed sets as quickly as possible in order to be able to evaluate them. However, in the basic method such sets are only obtained at the end, when the agent simultaneously develops the last coalition, for all the intermediate sets. The third method, which limits itself to the best group with intermediate tests, improves the preceding method by carrying out intermediate tests more quickly to find a set of the best group.

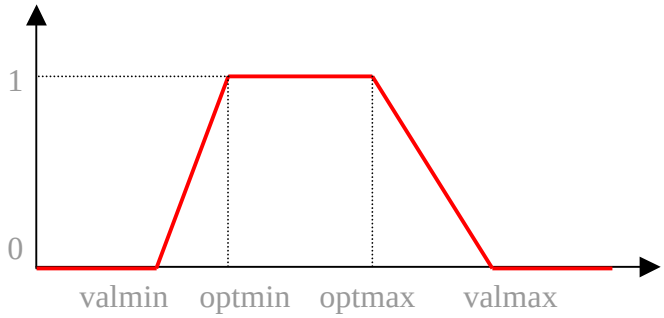
6.2.2 Student and professor agents

The utility function returns a complete result which relates, however, to the reference situation, the utility of which (absolute, measured by the *Ucomp* variable) is computed at the beginning of each negotiation.

The various parameters of the utility function are as follows:

- The time the day starts: for each day, the agent allots a utility to the time of the first class.
- The time the day ends: for each day, the agent allots a utility to the time the last class ends.
- Numbers of hours of the day: for each day, the agent allots a utility to the number of classes are given.

Each of these partial utility functions is of the following form:



Valmin: value below which the utility is null
Optmin and optmax: limits between which the utility is maximum
Valmax: value above which the utility is null
Criterion: time the day starts, time the day ends, Numbers of hours of the day

Figure 6. Partial utility according to the value of the selected criterion

For each day, the agent computes the weight of these three functions (*d(j,s), f(j,s), n(j,s)*) according to its own coefficients (p_d, p_f, p_n) based on the criteria it considers more important. The agent computes the average of these daily utilities. The professor agent has a fourth criterion: the total number of hours of classes in a week. It computes the weekly average of this criterion *(t(s))*, which is then weighted using *(Pt)* with the preceding daily average. The agent then seeks the number of classes *nbtt* which are not given in the evaluated sets. It uses an aversion coefficient c_t for the classes which are not given. It multiplies the intermediate utility by this coefficient to the power *nbtt*. In the same way, the agent seeks the number of classes it must attend (tasks which are allocated to it at the beginning) but which are not given *nbtp*. It uses a second coefficient c_p to the power *nbtp* and that it multiplies by the preceding result. Lastly, the agent seeks the number of classes *nbch* which have moved compared to the initial situation in its timetable. It uses a third coefficient c_c to the power *nbch* and that it multiplies by the preceding result. It then uses an absolute result from which it subtracts the utility of the current reference situation *Ucomp* in order to obtain a relative result.

For the student agent, the formula corresponding to these calculations is:

$$U(student) = c_t^{nbtt} c_p^{nbtp} c_c^{nbch} \frac{\sum_{s=1}^{nbs}\sum_{j=1}^{nbj}(p_d d(j,s) + p_f f(j,s) + p_n n(j,s))}{nbj \times nbs} - U_{comp}$$

Equation 1. The relative utility of a student agent associated with a set of coalitions

The utility function of the professor agent is defined as:

$$U(professor) = c_t^{nbtt} c_p^{nbtp} c_c^{nbch} \frac{\sum_{s=1}^{nbs}\sum_{j=1}^{nbj}(p_d d(j,s) + p_f f(j,s) + p_n n(j,s) + p_t t(s))}{nbj \times nbs} - U_{comp}$$

Equation 2. The relative utility of a professor agent associated with a set of coalitions

At the end, the agent checks if there are incompatibilities in the timetable (for instance, two classes programmed at the same time) or some violated constraints (a class starting before the minimum value or after the maximum value, etc.). If it is the case, it returns the value –1. If not, it returns the computed utility. All these parameters can be modified individually by each agent. Given their number, three profiles have been defined to simplify the choices by default: morning, afternoon and grouped (the agent prefers to group its classes on a minimum number of days).

6.3 Operational process

The first step consists in executing the environment agent. Professor and student agents can then be added, either directly with the assistance of the environment, or by a separate program, possibly on another computer on the network.

The various functions available for each agent are:

- Redefinition of capabilities and utility function. For both professors and students, all the parameters of the utility function can be modified. For the professors, the courses in which they are qualified to teach can also be defined.
- Modification of the search method. For each agent, its model can be chosen by defining the method that it will use to search for the sets of coalitions.
- Addition or deletion of classes. For the students, new classes can be added and the existing ones can be removed.

Once all the agents are created and utility functions are defined, an agent can starts the initialization phase of the negotiation in order to obtain a set of timetables, i.e. a solution. Other agents or other tasks may then be added and utility functions may be modified. An agent may then start the update step will seek a new solution starting from the current situation.

7 Evaluation results

The proposed solutions were tested through simulation. To understand these tests, remind the objectives of this work:

- To propose a protocol in order to form Pareto optimal coalitions.
- To propose a protocol which enables a dynamic reorganization of coalitions.
- To propose heuristics to search for coalition groups so as to accelerate the negotiation.

7.1 Coalition formation and optimality of the result

How should such a protocol be evaluated? We cannot check if the utility function is maximal, as we assume that the multi-agent system has several utility functions that are incomparable. We have checked that during the tests we always obtain a result and that this result is a Pareto optimum (as proved in sections 4.4.3 and 4.4.2). We have analyzed the performance of the protocol by observing several parameters: the number of messages exchanged, the size of these messages (the number of coalition sets they contain) and the number of coalition sets that have been evaluated. The number of messages exchanged between the agents is independent of the search method strategy. However, their size depends on the use of undeveloped coalitions. As for the number of evaluated sets, it depends very much on the search method used. The basic method systematically evaluates all the possible sets whereas the heuristics proposed seeks to reduce the number of these sets in order to obtain the result quicker. Four of these heuristics have been implemented. The heuristics which gave the best results and which are used in the experiments described consists in seeking only the best group by doing intermediate tests as soon as possible in order to identify the value of the best group.

In the following, we will analyze these factors on a simple example using 4 agents (2 professors, 2 groups of students) and 2 classes. Each group attends two classes, i.e. there are four tasks in the system). Several experiments have been done with more agents (for more details see (Caillou, 2000)). In this example, we consider the schedule for two days, with eight possible time slots per day. We vary the profile of each agent (morning, afternoon, grouped) to obtain the average, maximum and minimum of the results. Students attends two classes, each one must be placed in one of the 16 time slots (or not to be placed if it is not given, so we have 16+1 cases to consider) with only one possible professor for each class. Thus, there are 17^4, i.e. 83,521 possible schedules. The order in which the negotiations proceed is as follows: student s_1 starts the negotiation, then student s_2, professor p_1 and professor p_2 ends the negotiation. The last agent does not send any message as it just waits to receive a set which is appropriate for it and then sends it to the other agents as the solution for the negotiation.

Only messages related to groups are counted (they are more frequent and especially voluminous as they contain the sets which will be evaluated). The number of messages sent depends very much on the precision of the utility function of the agents. If agents have very precise preferences, they will distribute the sets among many small-sized groups and will send many messages. We choose agent utilities with one hundred levels, therefore there is a maximum of 100 messages sent by the first agent (the second can thus send a maximum of 10,000 because it can divide each group received into 100 other groups). The number of messages sent is summarized in figure 7.

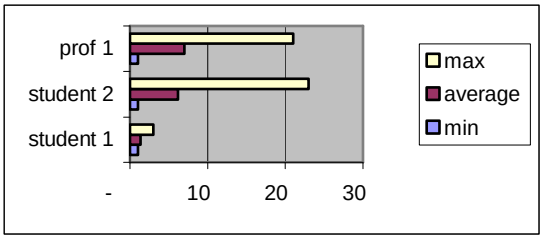


Figure 7. Number of messages exchanged during a negotiation with 4 agents and 4 tasks

The number of messages sent varies considerably according to the incompatibility of the preferences of the agents. For instance, a morning profile student will seek morning classes in priority. If the professor has an afternoon profile, it will consider these schedules unacceptable. Consequently, the student will have to send other propositions which it finds less appropriate. On the contrary, if all agents accept the first propositions, only one message per agent is necessary (minimal case). The total size of the messages sent (figure 8) makes it possible to measure network obstruction. This size, measured with the number of sets, must be compared with the 83,521 possible sets.

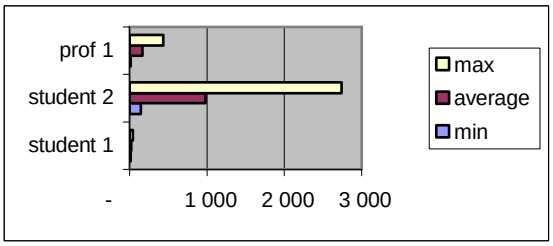


Figure 8. Total size of messages sent (measured by number of sets) during a negotiation with 4 agents and 4 tasks

The agent which sends most sets and messages is student s_2 for two reasons: (1) student s_1 sends it the sets corresponding to its two classes according to its preferences; (2) student s_2 computes all the possible combinations of its own classes in each of these sets. Then it sends these combinations in decreasing order of preference to professor a_1 until there are no more acceptable sets to send and if no solution has been found. At this moment, student s_1 sends it a second message and the negotiation continues.

The number of evaluated sets makes it possible to measure the effectiveness of the heuristic search of the best group. If the basic method is used, the first agent would simply evaluate all 83521 possible sets and would send them classified to the

following agent. Figure 9 shows that the maximum number evaluated set il less than 15000 and the average number is less than 5000.

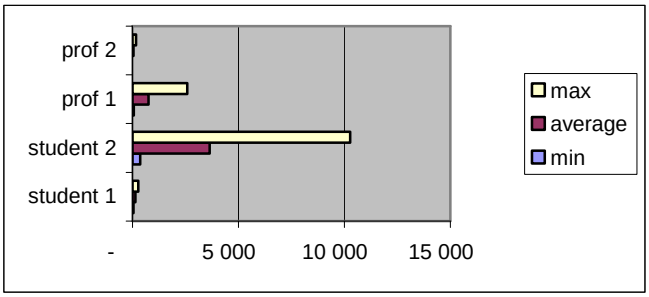


Figure 9. Number of evaluated sets of coalitions during a negotiation with 4 agents and 4 tasks

7.2 Dynamic restructuring of coalitions

The purpose of dynamic restructuring of the coalitions is to give a result that is as satisfactory as the basic protocol but faster, which is possible because the algorithm uses information drawn from the preceding negotiation by taking the previous solution as a new reference situation. The result will probably not be the same than the results obtained, if the initial protocol had been applied, but the result is always a Pareto optimum.

We have studied the effect of adding new classes to the previous situation in terms of the number of sets evaluated and transmitted. We gradually added 4 classes to students 1 and 2. The size of the messages sent during a negotiation is indicated in figure 10.

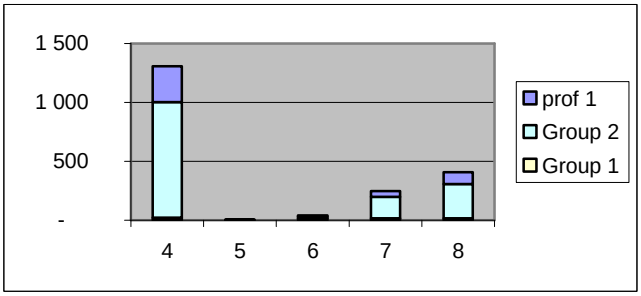


Figure 10. Number of evaluated sets (i.e. size of message sent) during a negotiation with 4 agents and classes varying from 4 to 8

The first negotiation (4 classes) used the basic protocol, whereas the other four are restructurings from the previous situation. The number of sets sent and evaluated must be compared with the total number of possible sets. It varies between 80,000 for 4 classes to 7.10^9 for 8 classes. The average size of the messages sent during these additions is indicated in figure 12.

During the formation of the coalitions corresponding to the four initial tasks, we showed that the total number of sets sent was on average 1,308 which implies that the total number of evaluated sets corresponds to 5.5% of the total number of sets of possible coalitions (83,521). During the dynamic restructuring of coalitions, due to the addition of the 6^{th} class, we observed that the total number of evaluated sets is 40 which gives the total number of evaluated sets corresponds to 0.00085% of the total number of possible coalitions.

The number of sets sent and the number of sets evaluated is related not to the total number of tasks carried out but to two parameters: the number of tasks the agents fail to perform (because of incompatible preferences) and the number of new tasks. The effect is cumulative, which explains why the number of sets sent gradually increases. For instance, if the 6^{th} class has not been assigned, this affects the number of sets sent after the addition of the 7^{th} and the 8^{th} classes because agent 1 tries again each time to assign the 6^{th} class (which is useful, as it may happen that in future negotiations a new class may modify the utility of the agents).

7.3 Comparison of the agent heuristics

The number of sets evaluated (figure 10) is here very low compared to the number of possible sets (on average 1% of the basic protocol and between 4.10^{-4}% and 6.10^{-6}% for the restructurings (figure 11 and 12)). However, the use of the basic protocol would have led to the computation and evaluation of all the possible sets before sending the groups of acceptable sets. The choice of a good heuristics to search for the best group is thus fundamental so that the search time is acceptable. The heuristics which gave the best results are described in section 5.2 and consist in seeking only the best group by carrying out intermediate tests as soon as possible in order to identify the value of the best group. Using this heuristics enables to obtain the following number of evaluations (cf. figure 11) during the reorganizations carried out previously by the agents:

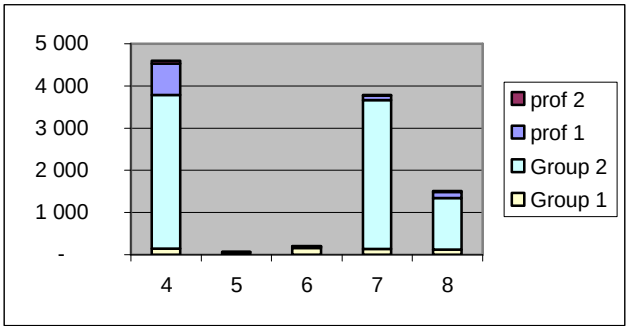


Figure 11. Number of evaluated sets of coalitions during a negotiation with 4 agents and classes varying from 4 to 8

Related to the total number of possible sets, we obtained the results presented in figure 12.

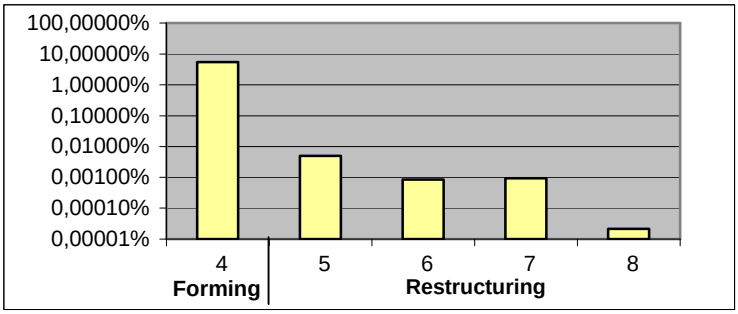


Figure 12. Percentage of valuated sets of coalitions for 4 agents and classes varying from 4 to 8 (logarithmic scale)

8 Related work

8.1 Theoretical origins

The use of coalition formation models is highly adapted to complex projects requiring the intervention of several agents selected from among a set of available ones. All the characteristics, even those related to the concept of an agent (an autonomous system with needs which can be represented in the form of a utility function), are such that scientific research in many fields deals with problems referring more or less directly to the coalition formation problem. Some of these

fields, such game theory, are already being used intensively in recent developments of multi-agent systems. Considering these different fields gives an overview of the problem and the various analytical approaches, which leads to original methods for coalition formation adapted to software and hardware agents.

Game theory has already addressed the question of coalition formation. It has provided the concepts used in MAS for the analysis of this problem (typology of the problems, solutions, equilibrium, utility functions). Through power indices, it is possible to compute the real influence of an agent in a coalition. Game theory provides methods of calculation to define the best coalitions in various types of problem. A good synthesis of the analysis of coalitions in game theory can be found in (Kahan and Rapoport, 1984). Its application to multi-agent systems has been originally studied by Sandholm (Sandholm, 1996). Game theory has been at the origin of the majority of recent developments (Sandholm and others, 1999, Sen and Dutta, 2000). The limits of its use are related both to the underlying assumptions (the agents are generally considered as perfectly rational) and to its aim (game theory focuses generally on the value of the optimal solution and not on the most efficient method to reach that solution, never on the most efficient distributed method).

Economic theory, especially microeconomic theory, is concerned with a research topic that is very similar to that of multi-agent systems: autonomous agents, considered as imperfectly rational and having needs modeled using a utility function. Many economic concepts can be used to address the coalition formation problem in MAS: the Pareto optimum, the maximization of the individual utility as a means. Our proposed protocol is grounded on these economical concepts (Caillou, Aknine, and Pinson, 2002a).

Sociological theory can also be useful to understand the coalition formation problem. Coalitions are groups of agents and these groups are not formed independently of their environment. The “Society”, in which they move plays a significant role in their choices: the presence of a third party (gendarme, state or program) may guarantee confidence. In the same way, the presence of social standards can accelerate or improve the formation of coalitions. Work like that of Shoham and Tennenholz (Shoham and Tennenholz, 1998) studied the influence of social standards and their emergence in a multi-agent system.

8.2 Multi-agent coordination models

The coalition formation problem has been studied in multi-agents domain since 1994 (Ketchpel, 1994). This model does not solve however the problem in all cases. Even in the cases where this algorithm finds coalitions, it does not always find coalitions of the desired size (Aknine, Pinson, and Shakun, 2004b). (Shehory and Kraus, 1998) proposed a model which is now considered by the community as the “basic model”.

Since then, most current protocols address the same problem while making improvements from this basic model at different levels of their solution. The main distinction is between improvements made to the methods, by preserving agents with limited rationality but by changing the objectives, and improvements made to the agents using more realistic or effective agents. In the following, we present the most relevant protocol of coalition formation, keeping in mind that unlike our protocol, these current protocols are based on aggregation of agent preferences or on a common utility function.

8.2.1 Forming overlapping coalitions with multiple tasks

The basic model proposed by (Shehory and Kraus, 1998) has the following characteristics:

- The game is not necessarily super-additive or sub-additive.
- The agents can take part in several coalitions simultaneously.
- Each agent has a list of capabilities and each task requires a list of capabilities.
- The tasks can be partially ordered.
- The agents cooperate and seek to maximize a single and total utility function. The problem of distribution thus does not arise.

The protocol is based on two phases. Initially, the values of all the possible coalitions are computed in a distributed way. The value of a coalition corresponds to the value of the task minus the cost of coordination related to the coalition and minus the cost of the capabilities used. In the second phase, the coalitions are formed gradually. In each sub-stage, the coalition which has the lowest cost per participant is formed and the value of all the remaining coalitions is recomputed so as to take into account the modifications generated by the use of capabilities related to the formation of coalitions.

The algorithm ends when there are no more tasks or agents. This algorithm remains very greedy in terms of time. In order to decrease its complexity, the authors recommend limiting the coalitions available, for instance, computing only thoses coalitions with less than k agents. The number of possible coalitions decreases then from 2^n to $O(n^k)$. The complexity of the protocol per agent is around $O(n^k \times |T|)$ (with $|T|$ the number of tasks).

This algorithm is promising especially because it is the first to propose a functional method for coalition formation within a general framework: several coalitions by agent. The protocol is simple and the simulations show that the results obtained are close to the optimal results. However, this protocol has several limits. (1) It focuses on the problem of coalition formation and the problems of distribution of the profits and of optimizations are not addressed. (2) A global utility function is supposed to be known and shared by all the agents. (3) The value of the set of all coalitions considered as acceptable is calculated. This calculation can become too complex to be done in reasonable time if the number of agents is too high as it could be the case in a

real application. (4) Each time that a condition changes (a new coalition, but also the addition of a task or of an agent, etc.), all the calculations for evaluating the values of the coalitions must be computed again. This constraint prevents the algorithm from being useful in open or dynamic environments. If agents constantly enter and leave of the system, if they build temporal preferences or if tasks are added progressively, the method cannot be used.

8.2.2 Search for a minimum solution

Sandholm, Larson, Andersson, Shehory and Tohmé (Sandholm and others, 1999) have proposed a protocol that provides a final solution which is at least equal to a certain proportion of the optimal solution, in term of value of the common objective function. This protocol can be applied under the following usual conditions. There are no externality between tasks, tasks are not necessarily super-additive or sub-additive, each agent belongs to only one coalition at a time. It is thus a problem of distributing the set of agents. The agents cooperate and seek to maximize a common and social utility function. This protocol reasons on the different levels of possible sets of coalitions represented in a lattice. For instance, for 4 agents there are 14 possible coalitions and 15 possible sets. This protocol shows that it is possible to obtain a result that is relatively close to the optimum by calculating a small proportion of the total number of nodes. It has been extended in (Sen and Dutta, 2000). We can, however, observe that when the number of agents become relatively high, the minimum guaranteed is very low compared to the optimum whereas the number of nodes to be computed and the calculation complexity increase rapidly (because of the number of agents).

8.2.3 Heterogeneous agents

8.2.3.1 Constraints on agents computation

The limits of the rationality of the agents can be addressed at several levels. Sandholm and V. Lesser (Sandholm and others, 1999) propose a protocol based on agents which are limited in their computation capabilities. Calculating the optimization necessary for coordination and negotiation is expensive. Compared to the calculating time, the agents minimize the total cost of the coalition, which is the difference between the cost of negotiation and the profit resulting from the negotiation for a given calculating time. This cost serves to compute the value of the coalition. The negotiation is expensive. However, the agents are supposed to be perfectly rational when they evaluate this cost. This means that they perfectly consider the profits resulting from a negotiation for a certain period of time and that each coalition is really optimized and at no cost. The principal interest of this article lies in the formalism which is presented. This formalism is adapted to agents limited in computing time. They have also proposed a relevant classification of the problems related to this case.

8.2.3.2 Agents with multi-criteria preferences

All the current models are based on the same assumption: the agents try to maximize a global utility function which was defined, either directly in the agents, or

by an agreement between the agents which will then distribute their payoffs. The coalition formation problem is considered as a distributed optimization problem. Though limited by the constraints of a decentralized multi-agent system (nevertheless the agents seek their personal interest and, unless all the agents seek the collective interest, there will be some constraints on distributing the payoff in order to ensure the stability of the solution), these methods propose a centralization of the problem (handling a collective utility function, often calculating the value of all the possible or desirable coalitions). This involves a strong complexity of the algorithms (even per agent), and coordination problems and social negotiation on the utility functions and on the final distribution.

A decentralized approach to the coalition formation problem would seem to comply both with the agent-oriented approach (independence) and with economic reality (a large number of agents). The self-interested case can also be considered. This approach was chosen by (Aknine, Pinson, and Shakun, 2004a, Aknine, Pinson, and Shakun, 2004b). They consider the preferences of the agents and not a global utility function. When a coalition is formed, agents consider the aggregated preferences of the coalition, but never a global aggregate such as the global utility which is neither necessary nor calculated.

The model presented here is based on two fundamental concepts: the Choquet integral as an aggregation operator and the ESD (Evolutionary System Design) methodology for coalition formation. The Choquet integral makes it possible to carry out multi-criteria aggregations by taking into account the collective weights of several criteria which are different from their sum. The agents are characterized by a multi-criteria utility function which assigns to each agent a vector representing its preferences with respect to each criterion. The Choquet integral is used to:

- aggregate the preferences of an agent with respect to another agent,
- aggregate the preferences of an agent with respect to several other agents and obtain its preference for a coalition,
- aggregate the preferences of several agents with respect to another agent in order to obtain the preference of a coalition for an agent.

To form the coalitions, agents use the ESD methodology to restructure a problem. This method has been developed by Shakun (Shakun, 1998). Two types of protocol are proposed, depending on how the agents do or do not share their preferences. Sharing the preferences is realistic in a cooperative situation; it provides a more comfortable situation and so a more effective solution. Forbidding the sharing of preferences makes it possible to handle non-cooperative situations with self-interested agents. The two protocols can be viewed as protocols for making contracts (such as the Contract-Net Protocol) with broader coalitions. When preferences are not shared, each agent concerned announces that it is seeking to form a coalition, receives alternative proposals, studies these proposals on its own behalf or as a representative of its coalition. If the preferences are shared, the agent which wants to form a coalition saves time by looking in its knowledge base the preference structures which are compatible with its own structure. (Vauvert and El Fallah-Seghrouchni, 2001) studies the case of overlapping coalitions which are formed gradually through alliances and progressive adaptation of the preferences of the agents (the agent interest it is to adapt so as not to be excluded from the coalitions).

8.2.3.3 Other models

Several coalition formation models have been suggested to date, for instance (Rahwan and Jennings, 2005) which extend earlier work of (Shehory and Kraus, 1998) and which provides some heuristics to improve the complexity of their mecanism.

(Tsevovat and others, 2000) proposed an algorithm based on the principle of electing a leader for coalition formation. This algorithm has been applied to electronic commerce processes. This approach is similar to the one proposed in (Aknine, Pinson, and Shakun, 2000). (Lerman and Shehory, 2000) have proposed an alternative, physics-motivated mechanism for coalition formation that treats agents as randomly moving, locally interacting entities. They consider that a new coalition may form when two agents meet randomly, and it may grow when a single agent randomly meets the coalition. The aim of this work was to define a mathematical model, formalized as a series of differential equations. These equations have steady state solutions that describe the equilibrium distribution of coalitions, but the authors have not given any details of the autonomous agent behaviors and how they concretely use this mathematical model. No algorithmic specifications have been proposed and the convergence of this model has not been addressed.

(Zlotkin and Rosenschein, 1994, Zlotkin and Rosenschein, 1996) have proposed a mechanism for coalition formation that uses cryptography techniques for sub-additive task-oriented domains. This mechanism is based on a Shapley value. A Shapley value for an agent is a weighted average of all the utilities of the agent which contributes to all possible coalitions. The weight of each coalition is the probability that this coalition will be formed in a random process that starts with the first agent, and in which this coalition grows by one agent at a time such that each agent that joins the coalition is credited with its contribution to the coalition. The Shapley value is the expected utility that each agent will have from such a random process . However, this mechanism can only be applied to small-sized multi-agent systems because of its combinatorial complexity due to the calculation of all possible coalitions.

Recently, a solution that suggests that agents compromise their gains to promote coalition formation was suggested (Kraus, Shehory, and Taase, 2003a, Kraus, Shehory, and Taase, 2003b). However, this work assumes that the value at which compromise is beneficial is known, or can be derived experimentally. In many real applications where coalitions are necessary, this assumption does not hold. In our solution to the coalition formation problem, we do not assume that the optimal solution points are known in advance. Rather, we provide agents with means to gradually arrive at an agreed solution via a series of discussions.

9 Conclusion and future work

In this paper, we have proposed a distributed protocol adapted to problems requiring coordination through the formation of coalitions where it is not desirable, or possible, to aggregate, or share, the preferences of the agents. The protocol provides

optimal Pareto-type solutions. One of the advantages of this protocol is that, if changes occur in the multi-agent system, it enables agents to compute a new solution, which is always Pareto-optimal, dynamically and quickly, on the basis of the current solution.

It is difficult to compare our protocol to current protocols since it does not have the same objectives. In current protocols, utility functions of the agents are either global for all agent or systematically aggregated. On the contrary, the utilities here are neither aggregated nor transmitted. The results cannot thus be compared because they relate to different problems. However, if all the agents have an identical utility function at the beginning, our suggested protocol should obtain the same result as that of (Shehory and Kraus, 1998) or (Aknine, Pinson, and Shakun, 2000).

For the considered problem, which is formation and restructuring of coalitions without aggregation of agent preferences, we have shown in this paper that the protocol allow to obtain a solution which is a Pareto optimum. Moreover, the tests have shown that the average complexity remained low compared to the total number of possible cases. In spite of these encouraging results, many improvements are still possible and are currently being addressed.

Regarding the protocol, a logical extension would be to send sets with constraints on the coalitions instead of sending several independent sets of coalitions. For instance, in our application of drawing up schedules, instead of transmitting three sets of coalitions with the three alternatives time 1, time 2, time 3, one agent could send: "time ranging between 1 and 3". This would reduce the number of sets of coalitions to be computed and would enable the agent which receives them to make an intelligent search instead of having to evaluate all the sets without seeking links between them.

10 References

Aknine, S., and Caillou, P., 2004, Agreements without disagreements: a coalition formation method, ECAI04: Valencia, IOS Press, p. 3-7.

Aknine, S., Pinson, S., and Shakun, M. F., 2000, Coalition Formation Methods for Multi-Agent Coordination Problems, International Conference on Group Decision and Negotiation.

-, 2004a, An Extended Multi-Agent Negotiation Protocol: International Journal on Autonomous Agents and Multi-Agent Systems, v. 8, p. 5-45.

-, 2004b, A Multi-Agent Coalition Formation Method Based on Preference Models: Group Decision and Negotiation, v. 13, p. 513-538.

Binmore, K., 1999, Jeux et théorie des jeux: Bruxelles, De Boeck.

Caillou, P., 2000, Pareto Optimality Method for Coalition Formation and Dynamic Restructuring of Agent Coalitions: Paris, Université Paris 9.

Caillou, P., Aknine, S., and Pinson, S., 2002a, A Multi-Agent Method for Forming and Dynamic Restructuring of Pareto Optimal Coalitions, *in* Castelfranchi, C., and Johnson, W. L., editors, AAMAS 02: Bologna, Italy, ACM Press, p. 1074-1081.

-, 2002b, Multi-Agent Models for Searching Pareto Optimal Solutions to the Problem of Forming and Dynamic Restructuring of Coalitions, *in* Harmelen, F. v., editor, ECAI 2002: Lyon, France, IOS Press, p. 13-17.

Kahan, J. P., and Rapoport, A., 1984, Theories of coalition formation: Hillsdale, LEA.

Ketchpel, S., 1994, Forming coalitions in the face of uncertain rewards, National Conference on Artificial Intelligence (AAAI): Seattle, p. 414-419.

Kraus, S., Shehory, O., and Taase, G., 2003a, The Advantages of Compromising in Coalition Formation with Incomplete Information, AAMAS 2003: Melbourne, ACM Press, p. 588-595.

-, 2003b, Coalition formation with uncertain heterogeneous information, AAMAS 2003: Melbourne, ACM Press, p. 1-8.

Lerman, K., and Shehory, O., 2000, Coalition Formation for Large-Scale Electronic Markets, ICMAS.

Ossowski, S., 2000, Co-ordination in Artificial Agent Societies: Berlin, Springer.

Rahwan, T., and Jennings, N. R., 2005, Distributing Coalitional Value Calculations Among Cooperating Agents, AAAI 2005: Pittsburgh, USA, AAAI Press, p. 152-157.

Sandholm, T. W., 1996, Negotiation among Self-Interested Computationally Limited Agents: Amherst, University of Massachusetts.

-, 1999, Distributed Rational Desicion Making, *in* Weiss, G., editor, Multiagent Systems, MIT Press, p. 121-164.

Sandholm, T. W., Larson, K., Andersson, M., Shehory, O., and Tohmé, F., 1999, Coalition Structure Generation with Worst Case Guarantees: Artificial Intelligence, v. 111, p. 209-238.

Sen, S., and Dutta, P. S., 2000, Searching for Optimal Coalition Structures, ICMAS, IEEE Press.

Shakun, M. F., 1998, Evolutionary Systems Design: Policy Making Under Complexity and Group Decision Support Systems: Oakland, Holden-Day.

Shehory, O., and Kraus, S., 1998, Methods for task allocation via agent coalition formation: Artificial Intelligence, v. 1998, p. 165-200.

Shoham, Y., and Tennenholz, M., 1998, On the Emergence of Social Conventions: Modelling, Analysis and Simulation: Artificial Intelligence, v. 97, p. 139-166.

Smith, R. G., and Davis, R., 1981, Frameworks for co-operation in distributed problem solving: IEEE Transaction on System, Man and Cybernetics, v. 11.

Tsevovat, M., Sycara, K., Chen, Y., and Ying, J., 2000, Customer Coalitions in the Electronic Marketplace, Agents: Barcelona.

Vauvert, G., and El Fallah-Seghrouchni, A., 2001, Coalition Formation among Strong Autonomous and Weak Rational Agents, MAAMAW 2001: Annecy, France.

Wooldridge, M. J., 1999, Intelligent Agents, *in* Weiss, G., editor, Multiagent Systems, MIT Press, p. 27-78.

-, 2001, An Introduction to Multiagent Systems, Wiley & Sons.

Zlotkin, G., and Rosenschein, J., 1994, Coalition, Cryptography and Stability: Mechanisms for Coalition Formation Task Oriented Domains, AAAI: Seattle.

-, 1996, Mechanisms for Automated Negotiation in State Oriented Domains: Journal of Artificial Intelligence Research, v. 5.