How to solve a scheduling problem by negotiation

Marie-Hélène Verrons[1] and Philippe Mathieu[2]

[1] Équipe MAGMA, Laboratoire Leibniz-Imag – CNRS UMR 5522
INPG, 38031 GRENOBLE, France
verrons@imag.fr
[2] Équipe SMAC, LIFL – CNRS UMR 8022
Cité Scientifique, 59655 VILLENEUVE D'ASCQ, France
mathieu@lifl.fr

Abstract. Timetable creation has usually been tackled as a constraints satisfaction problem (CSP). In this article, we propose a new, original and promising way to create timetables: negotiation. Using negotiation allow users to establish their timetable according to their preferences and without revealing their constraints. The problem we tackle here is to create a timetable involving teachers and groups of students. Multi-agent systems seem particularly of interest for this problem as different entities have to negotiate together.

1 Introduction

Problems involving resource allocation such as appointment taking or timetable creation have usually been tackled as a constraint satisfaction problems. For the two aforementioned applications, this forces the users to reveal their agenda in order to give their constraints. But one doesn't always want to reveal his private events. Thus a solution is for the users to keep their agenda private and to use negotiation to take appointments or courses.

In this article, we present a negotiation based approach for appointment taking and timetable creation. Our aim here is not to present an approach that gives the best results but an original one that is more flexible than others which must restart from scratch at each change in the environment. Using agent based negotiation enables us to dynamically add or remove constraints or agents. We show how to transform the problem into a negotiation problem between agents. Note that an agent can represent a person as well as a thing such as a room.

To illustrate our purpose, we use two applications: a meeting scheduling one and a university timetable creation one. The meeting scheduling problem has been studied by many researchers, like Sen and Durfee [6], Ito and Shintani [2], and Jennings [3]. They all propose a meeting scheduler that keeps private users agendas. The problem is to find an appointment that satisfies users preferences. Moreover, when an appointment has to be cancelled, the application automatically tries to reschedule it.

For the university timetabling problem, a benchmark has been proposed by the group Asa GDR-I3 [1]. This is a complex problem, and typically a constraints satisfaction problem which is NP-complete, where the solution isn't a priori known in the general case. This problem needs, in order to provide a solution, to be able to adapt itself in response to dynamic changes in the environment. This problem needs a collective search of the solution, and isn't a simulation problem: the aim isn't to recreate virtually the behaviour of an existing organism, but to furnish an expertise.

This kind of problem can lead us to think about some essential problematics such as how to bring all steps of a multi-agent oriented methodology into operation, how to identify agents to use in a problem resolution, how to take into account an open environment (how to manage the addition or retraction of constraints in real time), how to manage the fact that the environment is dynamic and that the system to be used has to adapt itself as a result and how to judge of the quality of a solution.

In order to develop these applications, we use a generic negotiation API: GeNCA. This API provides the whole management of negotiation processes and only needs communication configuration and strategy definition. For the applications we want to develop, there'll be no work for communication configuration as we'll use two of the provided communication ways: e-mail communication for appointment taking and a particular MAS communication (either Magique or Madkit) for timetable creation. Default strategies are also provided with GeNCA which are well-suited for our applications.

In this paper, we first present the negotiation approach for resource allocation problems and two applications that serve our purpose: a meeting scheduling application and a university timetabling creation one. Then, we present the negotiation toolkit used to build these applications, the way to develop them and the results of this approach.

2 How negotiation is helpful

2.1 The scheduling problems solved

Meeting scheduling. This problem is a negotiation application for appointment taking. Each agent must be able to negotiate rendezvous for the user. Each user defines a diary with time slots free or not and keeps this diary secret. In addition, he gives preferences on slots and on persons with who he prefers to take appointments. Each user can initiate an ask for a rendezvous with one or more participants on one or more time slots.

This problem is a full featured one because it needs preferences over persons, for example, my boss has a greater priority than my colleague, but also priorities over resources (here time slots), eg. if I don't want to have appointments at lunch time or before 8 am, then the corresponding time slots will have a lower priority. Moreover, appointment taking is an application where there are typically many renegotiations and retractions, because it is difficult to find time slots that fit everyone.

University timetable creation. The problem here is to create the timetables of students and teachers in a University. We present here the benchmark that has been proposed by the group Asa GDR-I3. Actors (in a UML sense) involved are: teachers, students groups and rooms. Each one of these actors (individually) has constraints to be satisfied (at best). A teacher has constraints over his availabilities (day of week, time slot), his skills (particular teaching), and his need of particular teaching equipment such as an overhead projector.

A group of students has to follow a particular teaching composed of a set of several courses of several teaching subjects. For example, x courses of subject 1, y courses of subject 2, and so on.

A room is equipped or not of particular equipments (overhead projector, ...) and can be occupied or not during a time slot, a special day.

We assume that for each actor, constraints are given in a list. The order in the list gives the importance of the constraint comparing to the others (the first one can be relaxed easier than the last one). The problem to solve consists of conciliating these constraints in order to propose a time table for a specified duration.

In order to compare the different solutions used to solve this problem, we proposed a typical scenario. Four versions are proposed, which increase the complexity of the timetable to produce. For each of these versions, a possible solution is given, which is obviously not unique but clarifies the problem.

First scenario. In a first time, the problem is simplified by supposing that time slots are fixed to two hours (8-10am, 10-12am, 2-4pm and 4-6pm), and that the problem has to be solved for two days ($d1$ and $d2$). Three teachers ($t1$, $t2$ and $t3$) teach a specific subject, and their constraints are: $t1$ cannot teach on $d1$ between 4pm and 6pm, and on $d2$ between 2pm and 4pm; $t2$ cannot teach on $d2$ between 10am and 12am and on $d1$ between 4pm and 6pm; $t3$ cannot teach on $d1$ between 2pm and 4pm and on $d2$ between 8am and 10am. Three students groups are considered ($g1$, $g2$ and $g3$), each one has to follow, on these two days, two 2hour-courses done by each teacher (that is to say 12 hours of courses in total).

For the moment, we assume that the system doesn't have to manage rooms availability: each group has a room. We also assume that actors cannot relax any constraint. In this case, a solution could be:

day and time	group 1	group 2	group 3
d1 8-10am	t1	t3	t2
d1 10-12am	t3	t2	t1
d1 2-4pm	t2	t1	
d1 4-6pm			t3
d2 8-10am	t1	t2	
d2 10-12am		t1	t3
d2 2-4pm	t2	t3	t1
d2 4-6pm	t3		t2

Second scenario. We add constraints over rooms. Three rooms ($r1$, $r2$ and $r3$) are free. Only rooms 1 and 2 have a overhead projector. Room 1 is occupied on $d1$ between 10am and 12am; room 2 is occupied on $d2$ between 8am and 10am and between 4pm and 6pm; room 3 is occupied on $d1$ between 2pm and 4pm and on $d2$ between 4pm and 6pm. Each teacher wants to use a overhead projector at least one time for each students group during the two days. Constraints can be relaxed to propose a solution. In this case, a solution could be:

day and time	group 1	group 2	group 3
d1 8-10am	t1/r1	t3/r2	t2/r3
d1 10-12am	t3/r2	t2/r3	t1/r1
d1 2-4pm	t2/r1	t1/r2	
d1 4-6pm			t3/r1
d2 8-10am	t1/r3	t2/r1	
d2 10-12am		t1/r1	t3/r2
d2 2-4pm	t2/r3	t3/r1	t1/r2
d2 4-6pm	t3/r2		t2/r1

On condition that the constraint over $r1$ unavailable on $d1$ between 10am and 12am can be relaxed in order to place the course of $t1$ for $g3$, and that the constraint over $r2$ unavailable on $d2$ between 4pm and 6pm can be relaxed in order to place the course of $t3$ for $g1$ (we could have tried to relax the constraint $t3$ cannot teach on $d2$ between 8am and 10am).

Third scenario. We add the possibility to change (modify, add or remove) constraints in real time. So, a teacher will be able to notify that he cannot teach on a special day and time, rooms will be able to become free or occupied,... As constraints can change dynamically, we must be able to design a system able to adapt itself to changes, without completely being reinitialised. For example, during a search for a solution to the previous example, the teacher $t1$ can notify that he can't teach on day $d1$ between 10am and 12am but instead he can teach between 4pm and 6pm. How the system can solve the problem without interrupting the search and starting from scratch? A solution could be to ask teacher $t3$ to teach to group $g3$ on day $d1$ between 2pm and 4pm in order to move $t1$'s course to the same day between 4pm and 6pm.

Fourth scenario. A last scenario could be to totally open the system, with actors that may appear or disappear in real time.

2.2 Which negotiation system are we using?

Many kinds of negotiation systems exist, such as the Contract-Net Protocol (CNP), auctions, multi-step negotiations or else combined negotiations. The CNP was proposed by Smith [7] in the early 80s to enable a manager to delegate a task to a contractor. The manager issues a call for proposals where he submits the task, contractors bid for achieving it and the manager collects the bids and

chooses which contractor will achieve it. The negotiation protocol we use here is an extension of the CNP which adds rounds of counter proposals from the contractors and the manager.

The negotiation system we use allows different actors to negotiate contracts over resources. One actor (the initiator) proposes a contract over several resources to a set of actors (the participants). Each participant answers either by accepting the contract or by rejecting it. If the contract has been accepted by a sufficient number of participants, the initiator confirms it. Otherwise, the initiator asks participants which resources they prefer for the contract. He then chooses another set of resources according to participants preferences and proposes a new contract to the participants. This cycle is done until a solution is found or a predefined number of rounds is reached.

2.3 How to transform a scheduling problem in a negotiation problem

The first thing to do is to determine which are the resources to be negotiated and who are the actors in the negotiation (those who will negotiate together over the resources). In the applications we use here, resources that will be negotiated are naturally time slots. Actors for the meeting scheduling problem are all possible participants in the meeting and for the timetable creation problem actors are the teachers, the students groups and the rooms.

Then, you have to define the negotiation protocol that fits best your application. For the scheduling problem, the extension of the CNP presented in the previous section is the best one. You also specify some features of the negotiation as the possibility to retract yourself from a contract previously taken and to renegotiate this contract. These features are typically needed for scheduling problems.

Finally, you need to define the constraints of each actor and assign them a negotiation strategy that takes these constraints into account.

2.4 Advantages of this approach

The advantages of this approach are twofold. On the one hand, the use of negotiation allows users to find a timetable that respect their constraints without having to revealing them to the others. It also enables them to manage themselves their constraints and so they choose which one to relax if needed. On the other hand, the multi-agent approach facilitates dynamic changes such as the arrival (or removal) of an actor (agent) and negotiation facilitates the changes of constraints. These dynamic changes are taken into account in real time and don't affect the whole process of finding a solution. That is to say that the process has not to be restarted from the beginning but adapts itself to the changes. The resulting system is thus more flexible facing dynamic changes during the resolution process.

3 GeNCA

The toolkit used to build a negotiation application is GeNCA (Generic Negotiation of Contracts API) [5, 4]. We'll give here a short description of GeNCA and detail its most important features for these applications.

3.1 General description

We present here our generic model for contract-based negotiations. This model is built on three levels, provides automatic renegotiation of contracts, allows to negotiate contracts either sequentially or simultaneously, and can be used as a negotiation help tool for users thanks to its user interface. GeNCA also takes the user preferences into account while negotiating.

GeNCA architecture. GeNCA is built on 3 levels: a communication level, a negotiation level and a strategic level. Each one has its specificity and doesn't affect the other. The communication level defines how agents communicate together: they can use e-mail communication or being part of a multi-agent system, for example.

The negotiation level is the core of the framework: it contains the management of negotiations and the protocol used. The contracts negotiated involve resources. The negotiation process of each contract is assigned to a micro-agent, thus enabling parallel negotiations. When there are conflicts on resources involved in several contracts, the user can choose either to negotiate these contracts sequentially, either in parallel. When there is no conflict, all negotiations take place simultaneously.

The strategic level allows a user to define its own negotiation strategy. As a matter of fact, the negotiation strategy is different according to the application: auctions and appointments are not negotiated the same way.

GeNCA features.

User preferences. Two priority lists are defined in GeNCA in order to store user preferences. One is to define priorities between resources, the other is for participants. These lists are used in strategies during negotiations in order to choose which contract to take in case of conflicts over resources for example.

Use ways. GeNCA can be used in 2 ways, whether the agent is responsible for the whole negotiation process or the user wants to answer proposals himself. GeNCA can thus be used as a decision help tool for human users that are assisted by agents.

End of negotiation. In GeNCA, negotiation ends either successfully or not. There is no obligation of success.

We have conceived default strategies for initiators and participants that takes into account the preferences defined by the user. These strategies can thus be used by any application that doesn't need more attributes in its negotiation activity. We detail here these strategies.

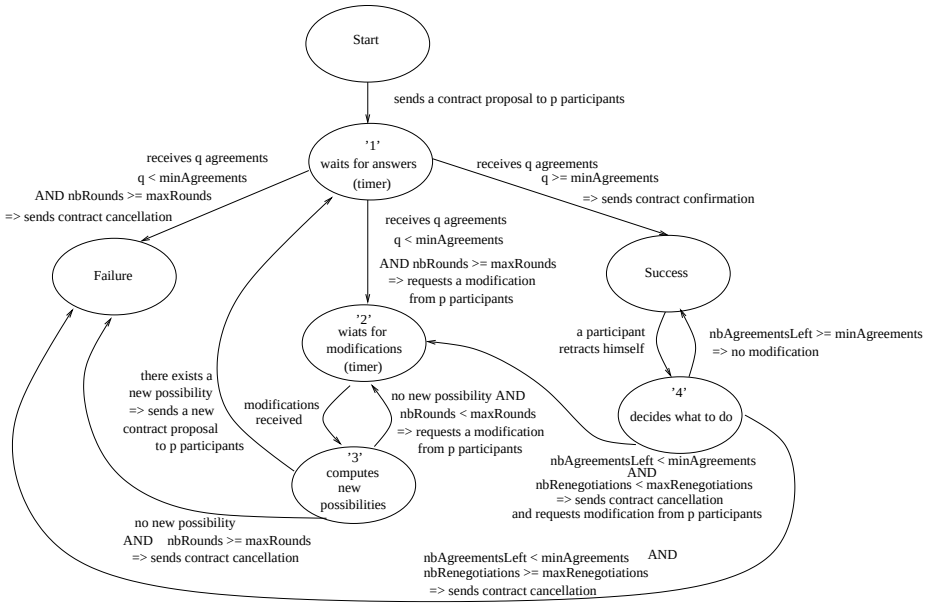

Fig. 1. Initiator behaviour

Initiator Strategy. The initiator's behaviour can be represented by a graph (Figure 1). It starts by sending a contract proposal to a set of `p` participants. The initiator goes in `state '1'` where he waits for participants' answer. Thanks to the answer delay defined at the contract creation, the initiator doesn't wait indefinitely for participants' answer. If a participant doesn't answer before the delay, the initiator considers a default answer defined at the contract creation. Each time an answer is received, the initiator takes it into account and updates his information such as number of answers received and number of agreements received, for example. When all answers have been received (or answer delay has expired), the initiator faces three possibilities:

1. If he has received `p` rejections, then he cancels the contract and goes to a `'Failure' state`.
2. If he has received `q` agreements with `q` greater or equals to the minimum number of agreements needed to confirm the contract, then he confirms the contract and goes to a `'Success' state`.
3. If he has received a number of agreements lower that the minimal number of agreements needed to confirm the contract, then he goes to `state '2'` where he decides what to do for the following of the negotiation.

When he is in `state '2'`, the initiator chooses his next action according to the number of rounds already done in the negotiation. If this number of rounds is greater or equals the maximum number of rounds in the negotiation defined by

him at the contract creation, he cancels the contract and goes to a `'Failure' state`. Otherwise, he requests a contract modification from the `p` participants and goes to `state '3'`. Once again, an answer delay prevents the initiator to wait indefinitely for participants' proposition. When the initiator has received all modification propositions, or the delay has expired, he goes to `state '4'`. When in `state '4'`, the initiator takes into account each modification he received in order to find a new possibility for the contract. To do so, he gives a note for each resource according to the following formula:

$$note(r_i) = priority(r_i, init) * priority(init, init) + \sum_{j=1}^{p} priority(r_i, part_j) * priority(part_j, init)$$

Where *priority(r_i,init/$part_j$)* stands for the priority of resource i for the initiator/participant (if it is not in the received list, it is given a discriminatory note), *priority($part_j$,init)* is the priority of participant j for the initiator and *priority(init,init)* stands for the importance the initiator gives to himself (in order to be compared with the other participants).

This note is updated at each reception of modification: the discriminatory note given to the time slot at a precedent round because of its non proposition by the participant is subtracted and the note is then increased by the calculation done by the second part of the formula.

These notes enable the initiator to find a new resource possibility according to his preferences and those of participants. Moreover, unproposed time slots are marked discriminatorily for the initiator to avoid propose them in the new contract.

Of course, other formulas could have been used, but we choose to use this one because we think that it takes the best into account everyone opinion and importance given to this opinion by the initiator.

If there exists a possibility, the initiator sends a new contract proposal to `p` participants and goes again in `state '1'`. If there is no new possibility, the initiator has two solutions. If the number of rounds in the negotiation is lower than the maximum number of rounds in the negotiation, then the initiator requests again a contract modification to the `p` participants and goes in `state '3'`. If the maximum number of rounds is reached, then the initiator cancels the contract and goes to a `'Failure' state`.

When the initiator is in a `'Success' state`, he may receives a retraction from a participant. That is to say that the participant can't meet the contract anymore. The initiator then goes to `state '5'` where he has to decide what to do face to this retraction. If the minimum number of agreements needed to take the contract is still reached, the initiator doesn't do anything and goes again in the `'Success' state`. If this number isn't reached anymore (or if the retraction comes from the initiator), the following depends on the number of renegotiations already done. If this number is greater or equals the maximum number of

renegotiations defined at the contract creation by the initiator, then the initiator cancels the contract and goes to a `'Failure' state`. Otherwise, the initiator cancels the contract and requests a contract modification from participants and goes to `state '3'`. Renegotiation is thus automatic, provided it is authorised.

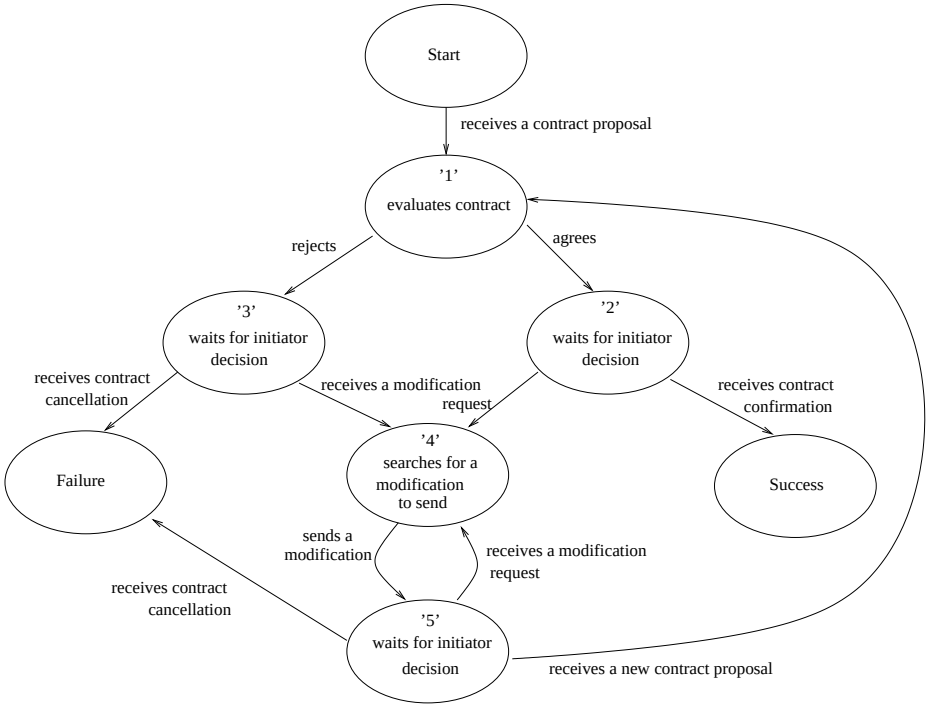

Fig. 2. Participant behaviour

Participant Strategy. The participant's behaviour can also be represented by a graph (Figure 2). It starts at the reception of a contract proposal. The participant then goes to `state '1'`. This state is the one where the participant examines the contract, evaluates it and decides whether to accept it and go to `state '2'` or to reject it and go to `state '3'`. When in `state '2'`, the participant waits for the answer of the initiator, which will be either a contract confirmation, either a modification request for the contract. If the contract is confirmed, the participant goes to a `'Success' state`. Else, he goes to `state '4'`. When in `state '3'`, the participant waits for the answer of the initiator, which will be either a contract cancellation, either a modification request for the contract. If the contract is cancelled, the participant goes to a `'Failure' state`. Else, he goes to `state '4'`.

When in `state '4'`, the participant searches for a possible contract modification and sends it to the initiator. The participant then goes to `state '5'`. In this state, the participant waits for the answer of the initiator, which will be either a new contract proposal, either a request for another modification, either a contract cancellation. If the contract is cancelled, the participant goes to a `'Failure' state`. If the initiator request another modification, then the participant goes to `state '4'`. If a new contract proposal is received, then the participant goes to `state '1'`.

3.2 Coding the problem with GeNCA

The meeting scheduling problem. For this application, e-mail communication is the most obvious communication way. Agents communicating by e-mail are provided in GeNCA, so only configuration files for e-mail have to be created for each user, no modification of the communication level is needed. Each user is assigned to an agent. This agent is responsible for negotiating meetings according to its user preferences. The resources are naturally time slots and contracts exchanged between agents contain only time slots proposed for the appointment. Thus no modification of the negotiation level is needed. The user can easily define his preferences over persons and time slots thanks to our priority lists for persons and resources. This can be done via the graphical user interface. This interface also lets the user create an appointment proposal by defining the time slots, participants and negotiation parameters such as the minimum number of agreements needed to confirm the appointment and the number of rounds in the negotiation. The user can choose whether to reply himself to appointment proposals or to let its agent do it. The default strategies proposed in GeNCA are well suited for this application. We give here some details about these strategies for their use in appointment taking.

Initiator strategy. When the initiator requests a modification from participants, they have to propose new time slots which better fits them.

Once all modifications have been received (or when the waiting delay has expired), the initiator chooses the time slots which best fit everyone thanks to the notes computed for each resource. He then proposes a new appointment for the time slots that have the best notes.

When an initiator receives a retraction, he looks if there are enough participants left for the appointment, otherwise he tries to move it.

Participant strategy. When a participant receives an appointment proposal, he first checks if the time slots are still free. If yes, he accepts the contract. Otherwise, he looks if the initiator of the contract has a greater priority for him than the initiator of the contract already taken. If yes, he accepts the contract, if not, he refuses it.

When a confirmation of appointment is received, the participant adds it to his diary, and sends, if necessary, a retraction message to the initiator of the less important rendezvous needing the same time slots.

When an modification request is received, the participant checks his/her diary and sends a predefined number of free time slots sorted by priority to the initiator.

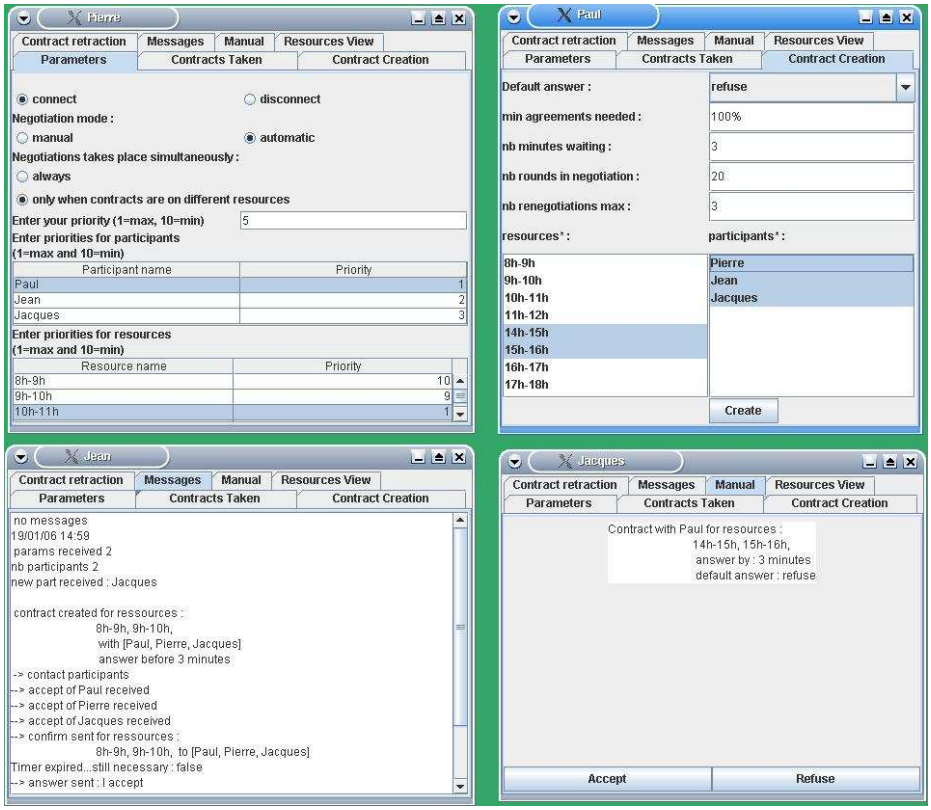

Fig. 3. Four agents in a Rendezvous application

Figure 3 shows interface of 4 agents negotiating rendezvous with our API.

The top left screen is the configuration tab of Pierre's agent. Pierre can defines his preferences over persons and time slots there, and also defines that he uses the agent for negotiating (automatic mode) and that he negotiates appointments that have common time slots sequentially.

The top right screen shows the appointment creation tab of Paul's agent. Paul can there defines an appointment that he wishes to take. He indicates time slots and participants, number of agreements needed, default answer, answer delay and number of rounds of negotiation and number of renegotiations. Then he launches the negotiation by clicking on the `create` button.

The bottom left screen shows the window for visualising messages sent and received by Jean's agent. It allows him to see the different proposals received and advancement of negotiations (answer sent, confirm, cancel, ask for modification, ...).

The bottom right screen shows the display of an appointment proposal to Jacques who uses its agent in manual mode. Jacques has to answer the proposals by clicking either on the `accept` button or the `reject` one.

The timetable creation problem. To solve this timetable problem, we propose to use negotiating agents in a multi-agent system. We use the MAS platform Madkit and the GeNCA API which provides a framework for building a negotiation application.

First scenario. The resources that will be negotiated are the time slots. To solve this problem, we decide to assign an agent to each actor, thus, 6 agents are defined: $t1$,$t2$, $t3$, $g1$,$g2$ and $g3$. Agents work on an asynchronous mode. Each teacher inputs his timetable in real time (they use GeNCA as a negotiation help tool), students groups are in an automatic mode, that is to say the agents work as background tasks. Users give priority to resources and to the other actors.

When all teachers have the same priority for students groups, the systems runs in a first arrived, first served way. When teachers have different priorities, courses taken by a less prior teacher can be moved automatically.

Here's an example of a timetable found when all teachers have the same priorities and don't have constraints:

	group 1	group 2	group 3
J1 8-10am	e1	e3	e2
J1 10-12am	e2	e1	e3
J1 2-4pm		e2	e1
J1 4-6pm	e3		
J2 8-10am	e1		e2
J2 10-12am		e1	e3
J2 2-4pm	e3	e2	
J2 4-6pm	e2	e3	e1

Adding constraints to teachers can prevent them to find a timetable. Let's take the following example: there are 2 teachers, 1 group and 5 time slots and teachers must see the group twice. Here are the teachers constraints:

t1	
8-9am	
9-10am	
10-11am	////////
2-3pm	////////
3-4pm	

t2	
8-9am	
9-10am	////////
10-11am	////////
2-3pm	
3-4pm	

Teacher $t1$ chooses its slots first, suppose he chooses $8-9am$ and $3-4pm$. Then, the only slot for teacher $t2$ is $2-3pm$, and thus he won't see the group twice. So a solution isn't found, whereas there was this one:

	group 1
8-9am	t1
9-10am	t1
10-11am	
2-3pm	t2
3-4pm	t2

The problem is solved if teacher $t2$ has a greater priority than teacher $t1$. As a matter of fact, $t2$ will propose either $8-9am$ or $3-4pm$ and the group will accept it and ask $t1$ to move its course.

Fixing priorities between teachers doesn't automatically solve the problem, even when there is a solution. As a matter of fact, GeNCA doesn't force to take the contract, so with no other features in the agent, if it doesn't succeed in its negotiation, it won't try to move another contract in order to be able to take the one he failed to take. The teacher has to monitor its agent to check that all courses have been taken and otherwise he must cancel courses and move them in order to take the missing ones.

Second scenario. We introduce in the system 3 new agents $r1$, $r2$ and $r3$ that represent the rooms. Now, teachers have to choose the group and the room when they create their contracts. A teacher can select all rooms when he creates the contract so that he can see if there's a place for his course, choose the room he wants if several are free and search another time slot if no room is free. Selecting all rooms isn't a problem as you can choose to confirm the contract for some of the participants and cancel it for the others. To do so, initiator strategy might be slightly modified in order to keep only one 'room participant'.

Third and fourth scenarii. Relaxing constraints and adding new ones are already provided in GeNCA, as you can cancel previous contracts and add new ones. If a contract for a course is cancelled, it will be automatically renegotiated.

Having a multi-agent system enables us to add or remove agent in real time, without perturbing the whole application and having to restart from scratch. Multi-agent systems properties make it possible to adapt to the third and fourth scenarii without any adaptation effort.

3.3 Concrete results

The meeting scheduling problem. The meeting scheduling problem has been studied by many researchers, like Sen and Durfee [6], Ito and Shintani [2], and Jennings [3]. They all propose a meeting scheduler that keeps private users agendas. The problem is to find an appointment that satisfies users preferences.

Moreover, when an appointment has to be cancelled, the application automatically tries to reschedule it. Their works are closed to ours. Some of them take into account more user preferences than us, but it is because their application is only devoted to meeting scheduling whereas GeNCA is a generic negotiation API that we have here used for meeting scheduling. Thus specific aspects of meeting scheduling are not included in GeNCA.

Different commercial softwares exist such as Microsoft's Schedule+ 1.0, ON Technology's Meeting Maker 1.5, Now Software's Now Up-to-Date 2.0 and WordPerfect Office 3.0's Calendar module, but all of them are based on agenda sharing. So none of them negotiates meetings but solves a CSP.

Our application allows agents to negotiate meetings for their user. Contrary to other commercial systems, users agendas are private and the problem is not simply to find common time slots free for all participants knowing their agendas, but to negotiate time slots for a meeting taking into account users preferences. Moreover, our system automatically negotiates again a meeting that has to be moved due to participants retractions.

System convergence is linked to negotiation parameters. As a matter of fact, our strategy consists of proposing all time slots one after another by order of preference according to the notes computed for each one. Each time slot is proposed only once, so the negotiation ends as soon as a time slot fits or when all time slots have been proposed. So negotiation always ends as we do not use an infinite calendar.

The timetable creation problem. In the previous section, we showed that using a multi-agent system to create timetables was a solution that enables to dynamically add or remove agents (teachers, groups or rooms) or constraints, without having to stop the process and restart the search from scratch. Using negotiation allows teachers to create themselves their timetables but it may be possible that a suitable timetable for all courses is not found because of constraints that are not shared between agents.

In our experiments, we have used agents that negotiate contracts that have been created by the user. We haven't added to the agent skills to store a list of meetings or courses that must be scheduled and to check that they have been scheduled. As GeNCA doesn't specify that the contract must be taken at the end, it is possible that all meetings or courses the user wanted aren't scheduled. The strategy used looks at each possibility of free resources (or resources taken by a less prior initiator) before cancelling the contract but it doesn't cancel another contract in order to take the new one. Each teacher must then check that he has all his courses. To face this problem, we should use an agent having those skills. On the contrary, GeNCA is adapted to dynamic changes: if a course is cancelled, it is automatically renegotiated.

4 Conclusion

In this paper we have presented a novel approach to solve scheduling problems such as appointment taking or timetable creation. Using negotiation to establish timetables is a new and promising research field. It enables teachers to find a timetable taking into account their constraints without giving them to the others. Moreover, it is a flexible approach that facilitates the integration of dynamic changes such as modification of constraints or arrival of an actor. It is important to keep in mind that this approach is interesting because of its flexibility and not for the results it gives. For the moment, we don't try to give the best results for scheduling problems but we want to show that negotiation is an approach that is worth to be explored. More experiments are needed for the timetable creation problem in order automate the whole process. We are at the moment designing more specialised agents that will have the list of courses they must schedule in order to make experiments without human intervention for contract creation.

References

1. ASA group members. Emploi du temps. Technical report, groupe ASA du GDR-I3, 2003. http://www-poleia.lip6.fr/∼guessoum/asa.html.
2. Takayuki Ito and Toramatsu Shintani. An Approach to a Multi-agent Based Scheduling System Using a Coalition Formation.
3. Nick R. Jennings and A.J. Jackson. Agent based meeting scheduling: A Design and Implementation. *Electronics Letters, The Institution of Electrical Engineering*, 31(5):350–352, March 1995.
4. Philippe Mathieu and Marie-Hélène Verrons. Three different kinds of negotiation applications achieved with GeNCA. In *Proceedings of the International Conference on Advances in Intelligent Systems - Theory and Applications (AISTA) In cooperation with the IEEE Computer Society*, Centre de Recherche Public Henri Tudor, Luxembourg-Kirchberg, Luxembourg, 15-18 novembre 2004.
5. Philippe Mathieu and Marie-Hélène Verrons. A General Negotiation Model using XML. *Artificial Intelligence and Simulation of Behaviour Journal (AISBJ)*, 1(6):523–542, August 2005.
6. Sandip Sen and Edmund H. Durfee. A Formal Study of Distributed Meeting Scheduling. *Group Decision and Negotiation*, 7:265–289, 1998.
7. R. G. Smith. The Contract Net Protocol : high-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, C-29(12):1104–1113, December 1980.