A MultiAgent Architecture for Distributed Course Timetabling

Luca Di Gaspero[1], Stefano Mizzaro[2], and Andrea Schaerf[1]

[1] Dip. di Ingegneria Elettrica, Gestionale e Meccanica – Università di Udine
via delle Scienze 208, I-33100, Udine, Italy
Phone: +39-0432-558242 | +39-0432-558280
e-mail: {l.digaspero|schaerf}@uniud.it
[2] Dip. di Matematica e Informatica – Università di Udine
via delle Scienze 206, I-33100, Udine, Italy
Phone: +39-0432-558456, e-mail: mizzaro@dimi.uniud.it

Abstract. The course timetabling problem consists in scheduling a set of lectures in a cyclic fixed period of time, typically a week. We consider the course timetabling problem for a set of university departments, in which each department prepares the schedule for its curricula according to private rules, constraints, and objectives, and relying on own resources. Resources are not common property, but departments could negotiate in order to share and/or exchange them for mutual benefits.
For this problem, we propose a multiagent scheduling system based on a marketplace and an artificial currency. In this framework, each department has a team of three cooperating agents, which are responsible for different tasks: searching for a local solution, negotiating resources with other departments, and managing relevant information.
To prove the effectiveness of the architecture, we present an experimental analysis that shows the benefits in the real situation of our institution.

1 Introduction

The course timetabling problem consists in scheduling a set of lectures between teachers and students in a fixed period of time, typically a week. We consider the timetabling problem for a set of university departments (or schools, or faculties) that have to schedule the courses of their curricula in a given term. Each department prepares its weekly schedule based on its endowment of rooms, and according to its own constraints, rules, and objectives. In general, a department is not willing to share its information with the other departments; therefore we assume that all input data are private for each department and thus inaccessible to the others.

On the other hand, whenever resources are usable for more departments, e.g., they are located in the same site, departments could benefit from sharing and/or exchanging their resources. Indeed, the resource endowment for each

term is not always optimally suited to the needs of the departments, but rather based on political and historical matters. Moreover, there are no global objectives to be satisfied; therefore all departments exchange resources for their own selfish interest, although they have a moral impulse to be helpful with the other departments, whenever possible without loss.

In our university, the Faculty of Engineering uses an automatic solver (described in a previous work, not cited for anonymity) that schedules the courses in a quite satisfactory way. Unfortunately though, it is not able to negotiate automatically with the other departments that are located in the same campus. At present, the negotiation takes place verbally among the deans of the departments, the administrative staffs, and/or the persons that operate the timetabling system. It requires good "diplomatic skills", it is quite time consuming, and in general not effective enough.

Due to privacy of information, different objectives, and selfish behavior, the use of a single centralized timetabling system (or at least a centralized room assignment system as discussed in [9]) is not a viable option within our framework. Furthermore, the presence of different objective and the genuine hardness of the timetabling instances make a distributed (but complete) solution [16] impractical as well. For these reasons, we propose an automatic scheduling system based on a multiagent architecture. Each department has three cooperating agents, called Solver, Negotiator, and Manager, which are responsible for searching a solution, negotiating with other departments, and managing and updating relevant information, respectively.

We propose a general architecture for the system and we describe the tasks and the functionalities of each of the three agents. Our current efforts are mostly focused on Solver, and we have developed a version of Solver that is based on the local search paradigm [5]. Our local search algorithms exploit different cost functions that take into account expenses related to possible trades. The algorithms rely on information coming from Manager about the probability of finding certain resources and the expected prices for them.

We also provide an experimental analysis of the system in a specific setting and on real data. In this analysis, we evaluate the overall performance of the system, based on a small number of parameters that control the behavior of the agents.

2 Course Timetabling

We first define the internal problem that each individual department has to solve based only on its own resources. Secondly, we introduce the general problem with several departments, and we discuss the global assumptions.

2.1 Centralized Course Timetabling

Among different formulations proposed in the literature [13], we consider the one that applies to our institution, though slightly simplified for the purpose of generalization.

There are q courses $c_1, \ldots, c_q$, p periods $1, \ldots, p$, and m rooms $r_1, \ldots, r_m$. Each course c_i consists of l_i lectures to be scheduled in distinct time periods, and it is attended by s_i students. Each room r_j has a capacity cap_j, in terms of number of seats. Since teachers are not available for all periods, we define a $q \times p$ availability matrix A, such that $a_{ik} = 1$ if lectures of course c_i can be scheduled at period k, $a_{ik} = 0$ otherwise. The input includes also a $q \times q$ conflict matrix *CM*, such that cm_{ij} is equal to the number of students that attend both courses c_i and c_j. An element cm_{ij} has the conventional value -1 if the courses i and j have the same teacher (in this case the number of common students is irrelevant).

The output of the problem is an integer-valued $q \times p$ matrix T, such that $T_{ik} = j$ means that course c_i has a lecture in room r_j at period k, and $T_{ik} = 0$ means that course c_i has no class in period k.

We search for a T such that the following *hard* (H) constraints are satisfied, and the violations of the *soft* (S) ones are minimized.

H1. Lectures: the number of lectures of course c_i must be exactly l_i.

H2. Room Occupancy: two distinct lectures cannot take place in the same room in the same period.

H3. Teacher conflicts: lectures of courses with common teacher must be all scheduled at different times.

H4. Availabilities: a course cannot be scheduled in a period in which the teacher is not available.

S1. Student Conflicts: lectures of courses with common students should be all scheduled at different times.

S2. Room Capacity: the number of students that attend a lecture should be less or equal than the number of seats of the room that host the lectures.

In order to fit the soft constraint types in a single objective function, we define the notion of *unit of penalty*: a unit of penalty is *a student that is forced to miss a lecture*. According to this definition, for S1 the weight of a violation is the number of common students cm_{ij}. In fact, these are the students that are forced to choose one out of the two lectures, and consequently miss the other one. For S2 the weight is the number of students in excess w.r.t. the capacity of the room (we assume that it is not allowed to have students standing in the room, and students in excess are forced to skip the lecture).

We also assume implicitly that all rooms are considered identical, and they differ only for the capacity. Therefore we neglect the possibility that some specific courses might require special features (e.g., projector, audio system, accessibility for handicapped persons).

Let us remark that with this formulation it is relatively easy to find a feasible solution in practice, even though the underlying problem is still NP-complete. In fact, the constraint types which are normally the most difficult to be solved, namely the student conflicts, here are considered as soft.

2.2 Distributed Course Timetabling

The distributed course timetabling problem consists in the simultaneous solution of a set of instances of the centralized course timetabling problem, each instance representing a single department. Courses and rooms are distinct for each department, so that each department has a preassigned endowment of rooms that it can use freely for its own courses. Timeslots are the same for all departments: they represent the same time intervals, and all intervals have the same length.[1]

What makes the instances interconnected is the possibility for a lecture to be allocated in a room owned by another department (if it is free in the given timeslot). Rooms are the resources that can be exchanged, although they are never actually "given away", but simply "rented" for some timeslots. The unit of good to be traded is thus a pair room/timeslot, that we call *roomslot*.

Each department however maintains its own constraints and objectives, and there is no notion of global objective. In general, different departments may also use different constraint types, and this indeed happens in practice. In this work, for simplicity we assume that they all use the same constraint and objective types defined in the previous section.

The problem thus consists in searching a general solution in which departments might also use external roomslots, but they have no complete information about available external roomslots. The information instead comes only from explicit bids by the departments that accept to give away some of their roomslots.

In order to solve this problem we have to define bidding protocols, synchronization, and trading mechanisms that allow the departments to negotiate. The system we have to design should lead to virtuous behaviors, in respect of the autonomy (and the selfishness) of the departments.

[1] This hypothesis of equalized timeslots might seem too restrictive, but it is indeed natural for departments that want to exchange resources profitably, and in fact it is already agreed in our campus.

3 The Trading Framework

To devise a negotiation mechanism capable of fostering roomslot exchanges, we design an electronic marketplace that we call RSMP (for RoomSlot Market-Place).

There are different types of architectures and formalizations for marketplaces proposed in the literature (see, e.g., [1, 8, 10]). Following the usual dimensions (see, e.g., [15, chap. 7] or [7]), the negotiation environment in RSMP is: *single issue*, the only issue is the price; *many-to-many*, several agents take part in it, assuming both buyers and sellers roles; and *with correlated value*, the value of a roomslot depends partly on public factors (capacity, temporal location in the week) and partly on private factors (specific necessity for a department).

The aim of RSMP is different from a classical market/auction [15]: in the latter, the aim is that buyers offer the true valuation of the good on sale (this is why, e.g., Vickrey auctions are adequate); conversely, we aim at designing a mechanism that maximizes the number of roomslot exchanges, thus leading to win-win situations and to better timetables.

Therefore, RSMP needs to encourage, in some respect, a more cooperative behavior than classical economic marketplaces and auctions, with the aim of maximizing the social welfare. Notice that it is difficult to place precisely our agents in a welfare classification [2], because of the uncertainty of their utility function. That is, an agent that has to decide whether to sell a roomslot or not cannot foresee whether that roomslot will be useful in its final plan. This is due to the complexity of the search space and the non-determinism of the search procedure.

To deal with this scenario, and to avoid an overwhelming complexity, we need to make some working assumptions, underlined in the following description of RSMP.

First we assume that roomslots are only sold and bought, rather than bartered between departments. In addition, we assume that agents always trade single roomslots and not multiple units as a whole (leading to a combinatorial auction [12]).

These assumptions might look counterintuitive because it is easy to imagine that a department might be willing neither to sell a currently-unused roomslot without assurance of the acquisition of the necessary one, nor to buy only some of the roomslots necessary to fulfil its perspective timetable.

However, the current practice shows that a more complex trading mechanism would make the offer and demand matching rarely successful, thus blocking the market. Conversely, it is agreed that the departments accept a limited amount of risk so as to make the trading system work profitably.

To implement this mechanism, we define a notion of *artificial currency* that the agents spend/receive for buying/selling roomslots. This is a common practice; for example, a similar approach is used in [14, 3] for scheduling problems. We define a *unit of currency* to be nominally equivalent to one point in the objective function, i.e., to a student seat in a room for a timeslot. We call our currency SPT (seat per timeslot). Consequently we assume that each department is given an initial amount of available SPTs, and it can spend its SPTs as long as its budget is above zero.

Based on the above considerations and assumptions, in its current implementation the RSMP follows a simple synchronous mechanism: at the beginning of the negotiation phase, all the agents enter the marketplace. Then, all agents post their purchase and sale offers as sealed bids to the marketplace. Purchase bids are a pair, made up by wanted roomslot and offered price; sell bids, instead, do not have an associated price, and the roomslot will be sold to the best offer. This means that there is an implicit total *concession* by the seller. This greatly simplifies the trading mechanism because it does not require concession protocols [11] that take place in iterated rounds.

At the end of the bidding stage, the RSMP makes the *clear*, and it calculates a matching of bids by assigning each roomslot for sale to the bidder that offers the highest price (breaking ties randomly). In this matching, RSMP maximizes the number of deals made, so that a request for a roomslot of a given capacity is matched also by a roomslot with a bigger room (if no precise match is available).

The overall process, starting from bids posting to clearing, is iterated until either a maximum number of iterations has been reached or the RSMP is empty because all agents exited it. An agent exits the RSMP when it is satisfied, meaning that it has achieved all its deliberated trades. Between two consecutive clears (i.e., after a new solving round), bids are posted again from scratch.

An important point of our setting is that agents keep their SPTs over subsequent teaching terms. Therefore, an agent can be willing to sell its roomslots not only to buy others, but also to accumulate money to be spent later. This mechanism, that already exists in the current verbal negotiation in our institution (in a less formal way, though), allows a department to save money for future harder instances (i.e., more dense terms), and also creates more room for negotiations.

Since hard constraints are easily satisfied with internal resources (see Sect. *Centralized Course Timetabling*), we can assume that agents buy roomslots only to improve the objective function, and never to satisfy a hard constraint. Under this assumption, the buy price of a roomslot is a value that ranges from 0 SPTs to the capacity of the room: in a timeslot in which there are plenty of rooms available for all departments (typically, Friday afternoon) the value is normally

0; in a very crowded timeslot (e.g., Wednesday morning) a roomslot can give an improvement of the objective function up to its capacity.

Actually, is it worth remarking that the value of SPTs is only nominal, because in principle a department that is rich enough of SPTs could be willing to spend more than the value of a roomslot in order to provide against an actual penalty. Nevertheless, we assume that the maximum bid for a roomslot is its nominal value so as to prevent the possibility of generating *inflationary* phenomena in the marketplace. Such possibility will be investigated in a future work.

4 The Agents

In our framework each department has a team of three cooperating agents, with different tasks and objectives. In this section, we describe briefly the agents and their interaction.

The agent Solver generates the solutions to the timetabling problem for the department, using an optimization technique. However, in our framework the search strategy of the individual Solver must be adapted so as to take into account the deliberation of potential trades [6]. Solver will be discussed in detail in the following section.

The agent Negotiator participates in the RSMP, and its task is to deliberate the list of sell/buy bids. The buying bids are directly inferred from the solution received from Solver. The selling bids are autonomously decided by Negotiator, which can have different deliberation strategies: it might be totally *bold* by selling everything that is not used in the potential solution, or totally *conservative* by selling only resources that are useless also in best local solution, or have an intermediate behavior. Notice that due to the structure of RSMP the Negotiator does not need to have any pricing strategy.

The agent Manager has the task of maintaining the price quotations for needed resources (see, e.g., [14]). These quotations are necessary for Negotiator to make profitable bids and for Solver to estimate the cost of the missing resources, so as to evaluate whether a given solution is promising or not.

The information is stored into a two-dimensional map Q (for quotation), such that the indexes are the timeslots $1, \ldots, p$ and the available room capacities. The value of each cell is a list of *options*. Each option is a pair composed by a price and the estimated probability to find a roomslot of the given capacity at that price. For example if $Q(3, 40) = \{(10, 0.02), (20, 0.07), (30, 0.50), (40, 0.80)\}$, it means that Manager believes that the probabilities of buying a room of capacity 40 in timeslot 3 at prices 10, 20, 30, and 40 are 2%, 7%, 50%, and 80%, respectively.

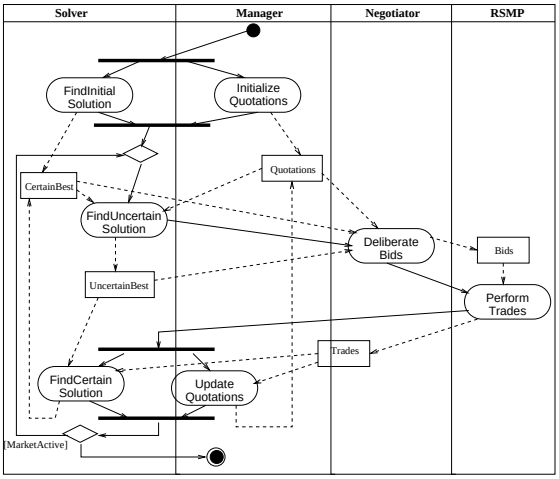

Fig. 1. Agents Activity Diagram

Manager updates the map after each trading session using the actual information coming from RSMP. In the current implementation, it simply takes as probability the frequency of the postings in the marketplace in all past runs. In the first run, the values are created from data coming from random training runs. A more sophisticated learning algorithm for the update will be developed in the future.

The behavior of the agents' team is shown by the UML Activity Diagram in Fig. 1. Initially Solver finds the best solution based on its own resources (called *certain best*) and Manager initializes the price quotations. In the next stage, Solver searches for the *uncertain best solution*, that is the best solution considering also potential trades. The uncertain best goes to Negotiator that first deliberates which sell/buy offers to post, and then enters the RSMP to trade with the other Negotiators. The outcomes of the trading are passed to Solver that updates the available resources and searches for the best (certain) solution in the actual situation, and to Manager that updates the quotations.

5 Search Strategy for Solver

All activities of Solver, shown in Fig. 1, are carried out using local search. Local search is a family of general-purpose techniques for search and optimization problems. These techniques are *non-exhaustive* in the sense that they do not assure to find a feasible (or optimal) solution, but they explore a search space until a specific stop criterion is satisfied.

Local search methods rely on the definition of the *neighborhood relation* and the *cost function*, which are the core features for the exploration of the

search space. The neighborhood of a solution is the set of solutions which are obtained applying a set of local perturbations, called *moves*. The cost function estimates the quality of each state, and is related to the hard and soft constraint violations.

Several search techniques can be defined upon this framework, depending on the criteria used for move selection and search stopping. The most common local search techniques are *hill climbing*, *simulated annealing*, and *tabu search* [4], not described here for the sake of brevity.

The activities FindInitialSolution and FindCertainSolution use the same algorithm, except that the former starts from a random solution, whereas the latter starts from the uncertain best solution that comes from the previous search. Their cost function is simply the sum of the objective function and the number of hard constraint violations, the latter being multiplied by a large constant.

The algorithm used is a "tandem" of a hill climbing and tabu search runs, such that they are repeated cyclically and each one starts from the final solution of the other. At each iteration, the hill climbing procedure draws a random move and executes it if the move is improving or sideways (same cost). Otherwise the hill climbing remains in the same state. The tabu search instead explores the full neighborhood and executes the best non-tabu move, being it improving or not. The loop continues as long as one of the two procedures provides some improvement.

The rationale for the tandem is that the hill climbing is fast and provides some diversification, whereas tabu search intensifies search in the promising areas of the search space. Due to their complementary roles in the search the two components use two different neighborhood relations: hill climbing moves a lecture in a different timeslot and a different room, tabu search moves either the timeslot or the room.

For FindUncertainSolution the search space comprises also all rooms that are stated as potentially available by Manager. In addition, the cost function is augmented by two new components:

- Cost of the roomslot options to be found in the marketplace; this is a soft constraint and its weight is simply 1 for each SPT to be spent.
- Total risk of the necessary purchases; this is also soft and weighted by multiplying it by a small constant. In addition, there is a *maximum acceptable risk*, called α, that behaves as a threshold, so that a total risk above α is considered a hard constraint violation.

Notice that there is no component that takes into account the gains coming from sales. This is a deliberate choice based on the intuition that if a department needs one of its roomslots, it should not try to sell it only because the possible

gain from the market is higher than the contribution to the internal objectives. In other words, we want to refrain departments from behaving like *speculators* that search more for the best deal rather than solving their actual problem, making consequently the market very unstable and unpredictable.

6 Experimental Analysis

We perform an experimental evaluation of the proposed multiagent system on two real timetabling instances from our university. Each instance comprises five departments and has a timetabling horizon of one week divided in twenty periods (four 2-hour slots per five days).

The first set of experiments aims at validating Solver in terms of the capability of deliberating successful trades, depending on the maximum acceptable risk α. For this purpose, all departments have the same configuration, based on a Negotiator that simply acts as a bold trader and a Manager that is fed with the bidding frequencies coming from a set of trial runs. Initial SPTs are assigned so as to balance inequalities in the room endowment (the sum of room seats and SPT is proportional to the total number of students in the lectures).

We run this experiment for α ranging from 0.0 (no trading) to 0.95 (max trading). For each risk level $\alpha \in \{0.0, 0.2, 0.4, 0.6, 0.8, 0.9, 0.95\}$ we perform 20 runs of the system and we record the total solution cost (i.e., the sum for all departments).

The results are reported in Fig. 2. For each instance, the graph plots the distributions of the normalized cost (by the median cost at $\alpha = 0$) against the risk level α. Due to the nature of Solver, the cost distributions are highly stochastic, and therefore we describe them by means of box-and-whiskers plots.[2] As intuitively expected, the trading mechanism significantly improves the quality of the timetables obtained by the isolated solvers (corresponding to the leftmost boxes, $\alpha = 0$).

More interestingly, Fig. 2 shows that there is a tendency of the cost values to decrease up to an optimal value of risk and to increase thereafter. This shows how an excessive confidence on the possibilities offered by the marketplace might result in an uncertain solution that eventually does not turn into a certain one of comparable quality. The optimal risk value can vary for different instances, depending on the general room occupancy in the given term.

[2] The dashed vertical lines represent the range of variation, the frequency is expressed through a boxed area featuring the range between the 1st and the 3rd quartile of the distribution, the horizontal line within the box denotes the median of the distribution and the notches around the median indicate the range for which the difference of medians is significant at a probability level of $p < 0.05$.

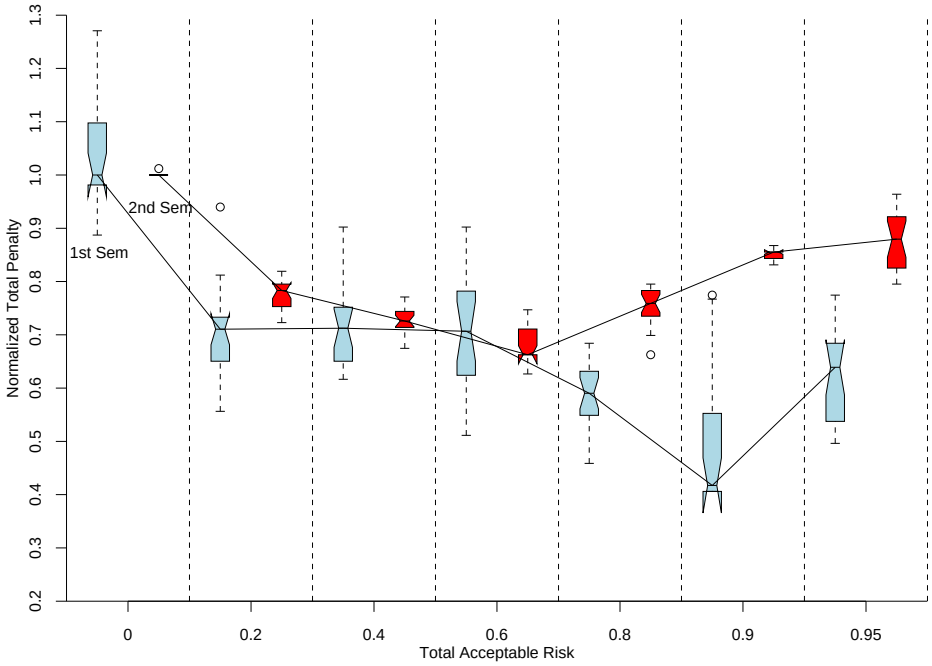

Fig. 2. Normalized total cost for different risk levels α

Fig. 2 only shows the aggregate values for all departments. Normalized costs for single agents are shown in Tab. 1. Data show that some departments (Civil Eng. Dept) have a very high gain whereas other (Environmental Eng. Dept) in some configurations have no gain at all, which reflects the unfairness of the room endowment. The table also shows that intermediate values of α lead not only to better results, but also to a more uniform distribution of the gain among all departments; this is due to the fact that all departments participate effectively in the market.

Dept.	0.2	0.4	0.6	0.8	0.9	0.95
Environmental	0.74	0.76	0.78	1.00	1.00	1.00
Civil	0.06	0.13	0.20	0.13	0.14	0.16
Electrical	0.39	0.46	0.54	0.49	0.38	0.32
Management	0.89	0.84	0.79	0.86	0.97	0.93
Mechanical	0.80	0.71	0.62	0.65	0.60	0.94

Table 1. Normalized costs for all agents and risk levels α, averaged on the two instances

Our second set of experiments investigates the behavior of a system configuration in which agents have different risk attitudes. We select two extreme risk levels 0.2 (risk adverse) and 0.95 (risk taker) and we run the system for a set

of configurations in which four agents have a normal risk level (0.6), and the remaining one is somewhat "unconventional" and has the extreme value.

Table 2 shows the normalized performance for only the unconventional agent. The left column shows that the risk adverse agent obtains reasonably good results, although not for all departments. This is intuitively explained by the fact that such an agent tends to make fewer buy bids at high price than the others (due to their higher risk level), resulting in a bigger number of successful purchases. For the same reason, the risk-taking agent obtains poorer results, as shown by the right column of the table.

Dept.	0.2 (risk adverse)	0.95 (risk taker)
Environmental	0.76	0.94
Civil	0.32	0.26
Electrical	0.49	0.52
Management	0.69	1.00
Mechanical	0.60	0.98

Table 2. Normalized costs for the unconventional agent averaged on the two instances

7 Conclusions and Future Work

We have shown a working multiagent architecture for distributed course timetabling that features interesting experimental results. The system would have an unquestionable practical usefulness, given the time currently spent in meetings and phone calls to obtain a quite unsatisfactory result; the main potential obstacles that we foresee to its use are the consensus on the marketplace rules and the acceptance of its outcomes by deans, professors, and students.

This is an ongoing work, and many features need to be further developed and investigated. First of all, recall that the formulation of the timetabling problem used in this paper is simplified w.r.t. the actual situation. In the future, we plan to experiment with the complete problem formulation accepted by the actual solver of our university (which has 15 constraint types, counting both hard and soft ones altogether). In addition, the use of external roomslots generally has some organizational costs that cannot be completely neglected. For example, the rooms might be distant to each other, or simply there is a cost for the confusion related to students wandering around searching for their room. The cost of the negotiation itself (time spent on computing, number of messages exchanged by the agents,...) should also be included in some way in order to obtain a fair comparison of the results.

The agent Manager must be redesigned in such a way that it updates its information by using some form of reinforcement learning mechanism, rather than the simple frequency values. Furthermore, we plan to develop a more informed

strategy for the deliberation of sell bids. For example, Solver could provide to Negotiator some objective information about the usefulness of roomslots based on statistics about its runs. Based on such information, Negotiator could use a less bold strategy and sell only roomslots that are clearly useless. Moreover, in the experiments we have assumed that SPT are assigned fairly (i.e., balancing the unfair room endowment), so that departments with less seats can bid higher and have more chance to get the rooms they need. However, in practice departments with more rooms might not accept such a distribution (they do not accept room rebalancing!). Therefore, we plan to investigate what happens if the initial SPT allocation is not fair, but based on other distributions.

The marketplace mechanism could be improved to allow also for different kinds of trades. For example, agents could be interested in swapping rooms of different capacities in the same timeslot (with some compensation in SPTs). Furthermore, some limited form of multi-roomslot trading could also help in some contexts.

Finally, we are interested in studying the interaction of different kinds of negotiators in the same marketplace. In particular, we want to investigate whether smart deceitful agents could take advantage of cooperative naive ones. Consequently, we plan to redesign the marketplace rules to prevent and/or neutralize such a behavior.

References

1. Anthony Chavez and Pattie Maes. KASBAH: An agent marketplace for buying and selling goods. In *Proc. of the 1st Int'l Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, 1996.
2. Ulrich Endriss and Nicolas Maudet. Welfare engineering for multiagent systems. In A. Omicini, P. Petta, and J. Pitt, editors, *Proc. of the 4th Int'l Workshop Engineering Societies in the Agent World (ESAW-2003)*, volume 3071 of *LNAI*, pages 93–106. Springer-Verlag, 2004.
3. K. Fischer, J. P. Müller, M. Pischel, and D. Schier. A model for cooperative transportation scheduling. In *Proc. of the 1st Int'l Conference on Multiagent Systems (ICMAS'95)*, San Francisco, USA, 1995.
4. Fred Glover and Manuel Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.
5. Holger Hoos and Thomas Stützle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann, 2004.
6. Kate Larson and Tuomas Sandholm. Bargaining with limited computation: Deliberation equilibrium. *Artificial Intelligence*, 132(2):183–217, 2001.
7. Alessio Lomuscio, Michael Wooldridge, and Nicholas Jennings. A classification scheme for negotiation in electronic commerce. *Int'l J. of Group Decision and Negotiation*, 12(1):31–56, 2003.
8. Peter McBurney and Simon Parsons. The Posit spaces protocol for multi-agent negotiation. In F. Dignum, editor, *ACL 2003*, volume 2922 of *LNAI*, pages 364–382. Springer-Verlag, Berlin Heidelberg, 2004.

9. Amnon Meisels and Eliezer Kaplansky. Distributed timetabling problems (DisTTP). In Edmund Burke and Patrick De Causmaecker, editors, *Proc. of the 4th Int. Conf. on the Practice and Theory of Automated Timetabling*, volume 2740 of *Lecture Notes in Computer Science*, pages 166–177, Berlin-Heidelberg, 2003. Springer-Verlag.
10. J. A. Rodríguez, F. J. Martin, P. Noriega, P. Garcia, and C. Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11(1):5–19, 1998.
11. J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing conventions for automated negotiation among computers*. The MIT press, Cambridge MA, 1994.
12. Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1–2):1–54, 2002.
13. Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.
14. Michael Wellman, William Walsh, Peter Wurman, and Jeffrey MacKie-Mason. Auction protocols for decentralized scheduling. *Games and Economic Behavior*, 35:271–303, 2001.
15. M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, 2002.
16. Makoto Yokoo, Edmund Durfee, Toru Ishida, and Kazuhiro Kuwabara. Distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Trans. on Knowledge and Data Engineering*, 10(5):673–685, 1998.